Preprint of article published as ACM Computing Surveys 57(10): Article 247:1--36, 2025.

# Hybrids of Reinforcement Learning and Evolutionary Computation in Finance: A Survey

SANDARBH YADAV, Indian Institute of Technology Bombay, India
VADLAMANI RAVI, Institute for Development and Research in Banking Technology, India
SHIVARAM KALYANAKRISHNAN, Indian Institute of Technology Bombay, India

Many sequential decision-making problems in finance like trading, portfolio optimisation, etc. have been modelled using reinforcement learning (RL) and evolutionary computation (EC). Recent studies on problems from various domains have shown that EC can be used to improve the performance of RL and vice versa. Over the years, researchers have proposed different ways of hybridising RL and EC for trading and portfolio optimisation. However, there is a lack of a thorough survey in this research area, which lies at the intersection of RL, EC, and finance. This paper surveys hybrid techniques combining EC and RL for financial applications and presents a novel taxonomy. Research gaps have been discovered in existing works and some open problems have been identified for future works. A detailed discussion about different design choices made in the existing literature is also included.

CCS Concepts: • **Computing methodologies** → **Reinforcement learning**; **Bio-inspired approaches**.

Additional Key Words and Phrases: Evolutionary computation, metaheuristics, finance, trading, portfolio optimisation

**ACM Reference Format:**
Sandarbh Yadav, Vadlamani Ravi, and Shivaram Kalyanakrishnan. 2023. Hybrids of Reinforcement Learning and Evolutionary Computation in Finance: A Survey. 1, 1 (November 2023), 43 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

Trading and portfolio optimisation (PO) are two of the most prominent activities in finance. While PO deals with managing a portfolio of financial assets, trading focuses on a single financial asset. Most practitioners use domain knowledge to formulate manual strategies for trading and PO. This process is time-consuming and prone to behavioural biases like overreaction [46], overconfidence [63], regret [14], loss aversion [92], etc. The dynamic nature of financial markets makes it difficult even for specialists to formulate profitable strategies consistently [82]. Recently, algorithmic trading has gained popularity by reducing the manual workload of traders. It also leads to stable trading as it suppresses the impact of human emotions. Contemporary automated systems in finance are mainly used for drawing inferences from the vast amounts of available data and executing trades, based on the rules fed by humans, at faster speeds [181]. However, the crucial decision-making aspect is still predominantly handled by humans. Intelligent systems are needed which can discover profitable strategies for trading and PO.

Researchers have proposed systems for trading and PO using supervised learning [39, 62]. However, some limitations of such systems include poor availability of accurate labels and the dynamic nature of ever-changing financial markets.

Authors' addresses: Sandarbh Yadav, Indian Institute of Technology Bombay, Mumbai, India, sandarbhyadav@cse.iitb.ac.in; Vadlamani Ravi, Institute for Development and Research in Banking Technology, Hyderabad, India, vravi@idrbt.ac.in; Shivaram Kalyanakrishnan, Indian Institute of Technology Bombay, Mumbai, India, shivaram@cse.iitb.ac.in.

RL [175] is a learning approach in which an agent continuously interacts with its environment and learns through trial and error. RL is one of the most natural approaches for sequential decision-making problems like trading and PO due to its ability to handle delayed feedback and dynamic environments. Additionally, the agent-environment framework of RL allows easier incorporation of practical considerations like transaction costs, risk preferences, etc. EC [54] is a class of population-based search metaheuristics, inspired by biological evolution, that work even when the objective function is non-linear, non-convex, non-differentiable, and non-unimodal. EC is quite helpful in finance as the objective functions become non-differentiable after incorporating practical considerations like transaction costs, risk preferences, etc.

Learning and evolution have contributed significantly to the development of intellectual capabilities in living organisms. Learning assists individuals in adapting to new situations while evolution leads to genetically superior populations across generations. Hinton et al. [75] have made one of the initial attempts to study interactions between learning and evolution. Ackley et al. [4] hybridise genetic evolution and reinforcement learning. Whiteson et al. [190] investigate how EC can improve RL and vice versa. EC has been used to improve RL in terms of better function approximation [190], selecting better state features [50], etc. Also, RL has been shown to assist EC in hyperparameter adaptation [178], balancing exploration-exploitation trade-off [190], etc. We use the term **EC+RL** for referring to techniques that hybridise EC and RL. EC+RL hybrids have been successfully used to solve various problems like feature selection [19], nuclear fuel assembly [148], satellite scheduling [166], sorting [189], spam bot detection [104], etc. Drawing inspiration from the success of EC+RL hybrids in other domains, researchers have applied EC+RL hybrids for financial applications like trading [50, 82, 201] and portfolio optimisation [7]. This paper presents a detailed survey of EC+RL hybrids proposed for financial applications like trading and PO.

**Motivation for the survey.** In literature, there exist many recent survey papers [57, 71, 125, 133, 174] focusing on applications of RL in finance. However, none of them specifically focuses on EC+RL techniques in finance. On the other hand, there also exist survey papers [11, 53, 167, 191, 205] which focus on EC+RL techniques but none of them specifically focus on financial applications. The finance domain has its intricacies in terms of the non-stationary, noisy, and chaotic nature of financial data. Although EC and RL have been independently used to tackle trading and PO in various research studies [103, 130, 135, 141, 172], there are multiple limitations. Incorporating practical considerations like transaction costs, risk preferences, etc. leads to a non-differentiable objective function that cannot be optimised using gradient-based RL approaches. EC can help RL in developing realistic solutions to financial problems due to its ability to handle non-differentiable objective functions. EC has also been used to evolve a population of RL-based trading agents that provide more robust judgement compared to a single trading agent. In PO, rules for asset selection have also been dynamically evolved. Financial applications like trading and PO are sequential decision-making tasks in which a decision is made at every time step. The decision at the next step is often influenced by the choice of actions in the previous time steps. However, most works using EC for finance utilise single-period formulation and neglect the sequential component. This leads to temporally incoherent decisions, resulting in higher transaction costs. RL can assist EC in this regard through its sequential decision-making framework. Thus, it is evident that EC and RL possess complementary strengths and their hybrids can significantly alleviate their limitations while amplifying their strengths. Using EC+RL hybrids can prove to be highly beneficial in the finance domain and this survey precisely captures that aspect. To the best of our knowledge, the literature does not contain a survey article at the intersection of RL, EC and finance (Fig. 1). This paper aims to fill the existing void by providing an exhaustive survey of EC+RL techniques proposed for two important financial applications: trading and PO.

**Intended audience.** The main intended audience of this survey paper includes practitioners who intend to use intelligent systems for executing trades and managing portfolios. Secondly, researchers working on EC+RL hybrids

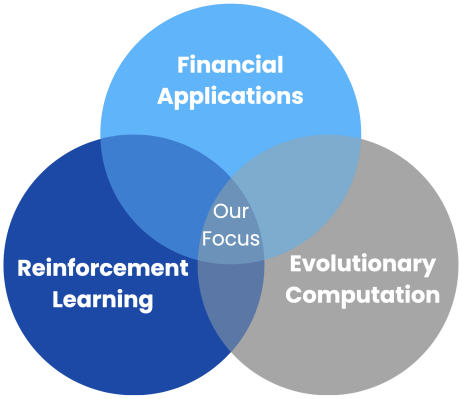
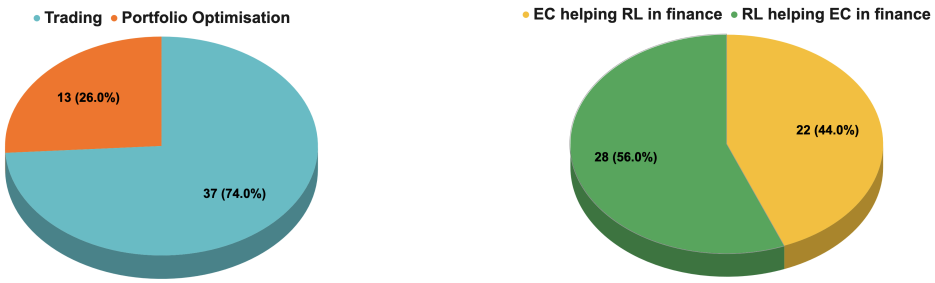



Fig. 1. This paper surveys EC+RL hybrids proposed for financial applications. We also present brief reviews of RL in finance (Section 2.4), EC in finance (Section 2.5) and EC+RL hybrids in general (Section 2.6).

can also find this survey paper useful. Lastly, the survey paper might be of general interest to researchers, educators, engineers, technologists, etc.

**Theme of the survey.** This paper focuses on two important financial applications: trading and PO (Fig. 2a). We classify EC+RL techniques in finance into two major categories (Fig. 2b). In the first category, RL is the main driver technique which provides the final decision while EC assists RL in arriving at that decision. In the second category, EC is the main driver technique that provides the final decision while RL assists EC in arriving at that decision. Both these categories are further classified on the basis of whether the hybrids are tightly coupled or loosely coupled. In tightly coupled approaches, RL and EC are closely bound and can not function without each other. On the other hand, in loosely coupled approaches, the binding is not tight and both EC and RL can operate independently of each other. Loosely coupled approaches simply focus on using one technique to improve some particular aspect of the other technique. The theme of the survey is succinctly captured by Fig. 3. EC has been shown to improve RL in a tightly coupled manner for gradient-free policy search (Section 3.1), state feature selection (Section 3.2), evolving population of trading agents (Section 3.3), evolving rules for asset allocation (Section 3.4), and controlling overfitting (Section 3.5). In terms of loose coupling, EC improves RL in selecting state features (Section 4.1), searching hyperparameters (Section 4.2), tuning technical indicators (Section 4.3), and generating labels (Section 4.4). The tightly coupled ways of improving EC using RL include dynamic indicator selection (Section 5.1) and generating signals for EC algorithms (Section 5.2). RL improves the performance of EC in a loosely coupled manner by adapting hyperparameters (Section 6).

**Organisation of the paper.** The remainder of the paper is organised as follows: Section 2 provides the required background in RL, EC, and finance. Section 3 surveys the approaches where EC helps RL in a tightly coupled manner for financial applications. Section 4 focuses on loosely coupled approaches that utilise EC to improve RL for financial applications. Section 5 and 6 present techniques where RL assists EC for financial applications in tightly and loosely coupled manner respectively. Section 7 discusses the research gaps in the existing literature and presents some open problems. Section 8 concludes the paper. We describe the paper collection methodology in Appendix A. The table of abbreviations is presented in Appendix B while the table of notations is presented in Appendix C.




(a) (b)

Fig. 2. Distribution of the surveyed papers (a) based on whether the financial application is trading or portfolio optimisation (b) based on whether EC helps RL or RL helps EC in finance.

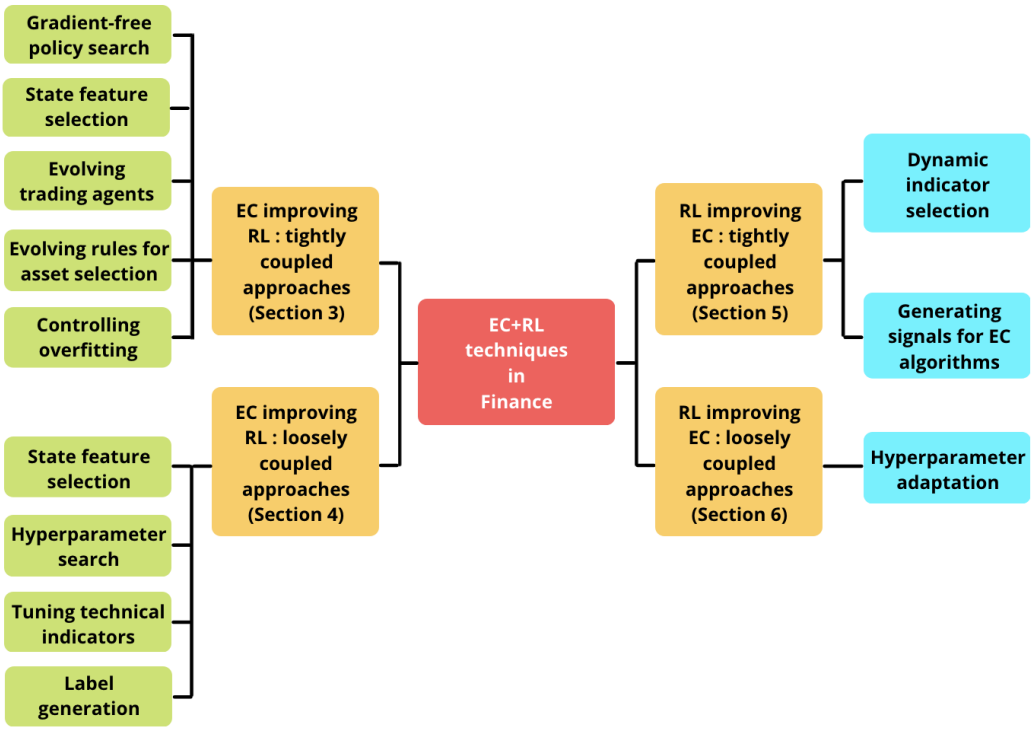



Fig. 3. Classification of EC+RL techniques in finance. Each classification category (in yellow) has its dedicated section with the leaf nodes discussed in corresponding subsections.

## 2 BACKGROUND

This section provides the required background in RL, EC, and finance. The finance domain comprises many sequential decision-making problems that have been modelled using RL and EC. We discuss two of them in detail: trading and portfolio optimisation. We also present brief surveys on RL in finance, EC in finance and EC+RL hybrids in general.

### 2.1 Reinforcement learning

Reinforcement learning [175] is a learning approach in which an agent learns by interacting with its environment through trial and error. Based on the current state, an agent takes a suitable action and consequently, the environment provides it with a reward and the next state. Fig. 4 depicts the typical agent-environment interaction for sequential decision-making. In finance, decision-making algorithms act as agents while the financial markets are treated as environments. RL models the environment as a Markov decision process (MDP) represented by the 5-tuple $(S, A, T, R, \gamma)$, where

- $S$ denotes the set of states that the agent can be present in.
- $A$ denotes the set of actions that the agent can take.
- $T : S \times A \times S \rightarrow [0, 1]$ denotes the transition function. For $s,\ s' \in S,\ a \in A,\ T(s, a, s')$ represents the probability of moving to state $s'$ from state $s$ if the agent picks action $a$.
- $R : S \times A \times S \rightarrow [-R_{\max}, R_{\max}]$, for some $R_{\max} > 0$, denotes the reward function. For $s,\ s' \in S,\ a \in A,\ R(s, a, s')$ represents the reward obtained by the agent if it moved to state $s'$ from state $s$ by choosing action $a$.
- $\gamma \in [0, 1)$ denotes the discount factor. Smaller $\gamma$ means lesser importance to future rewards.

The agent follows a policy $\pi : S \rightarrow A$. The value function of policy $\pi$ is denoted by $V^\pi : S \rightarrow \mathbb{R}$. $V^\pi(s)$ denotes the value of state $s$ under policy $\pi$. The value of a state under policy $\pi$ means the expected long-term discounted cumulative reward obtained by the agent if it starts in that state and chooses actions according to policy $\pi$. For $s \in S$,

$$V^\pi(s) \stackrel{\text{def}}{=} \mathbb{E}_\pi \left[ r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots \mid s_0 = s \right], \tag{1}$$

where $r_t$ denotes the reward obtained at step $t \in \{0, 1, 2, 3, \ldots\}$ and $s_0$ denotes the state at $t = 0$. Larger $V^\pi(s)$ is better. The goal of RL is to maximise the long-term discounted cumulative reward. $Q^\pi : S \times A \rightarrow \mathbb{R}$ denotes the action-value function under policy $\pi$. $Q^\pi(s, a)$ denotes the expected long-term discounted cumulative reward obtained by the agent if it starts in state $s$, taking action $a$ at $t = 0$, and following policy $\pi$ afterwards. For $s \in S, a \in A$,

$$Q^\pi(s, a) \stackrel{\text{def}}{=} \mathbb{E} \left[ r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \mid s_0 = s; a_0 = a; a_t = \pi(s_t) \ \text{ for } t \geq 1 \right], \tag{2}$$

where $r_t, s_t$ and $a_t$ denote the reward, state, and action respectively at time step $t \in \{0, 1, 2, 3, \ldots\}$.

The agent does not have complete information about the environment dynamics. It can learn by interacting with the environment using algorithms based on value function, policy, or both. Value-based approaches, also called critic-only approaches, focus on learning the optimal action-value function and use it to obtain the policy. Some popular examples of value-based approaches include Q-learning (QL) [186] and state–action–reward–state–action (SARSA) [154]. Policy-based approaches, also called actor-only approaches, focus on learning the optimal policy directly without involving the action-value function. If the policy $\pi$ is parameterised by parameter vector $\boldsymbol{\theta}$, various algorithms can be used to optimise for $\boldsymbol{\theta}$ using the objective function $J(\boldsymbol{\theta}) = V^{\pi_\theta}(s_0)$, where $s_0$ denotes the start state. $J(\boldsymbol{\theta})$ can be expressed as follows (see Sutton and Barto [175], Chapter 13):

$$J(\boldsymbol{\theta}) = \sum_{s \in S} \mu^{\pi_\theta}(s, s_0) \sum_{a \in A} \pi_{\boldsymbol{\theta}}(a|s) Q^{\pi_\theta}(s, a), \tag{3}$$

where $\mu^{\pi_\theta}$ denotes the on-policy distribution under $\pi_\theta$ with the start state $s_0$. Here, $\pi_{\boldsymbol{\theta}}$ is a stochastic policy that assigns a probability distribution over the set of actions $A$ as opposed to a deterministic policy that specifies a single

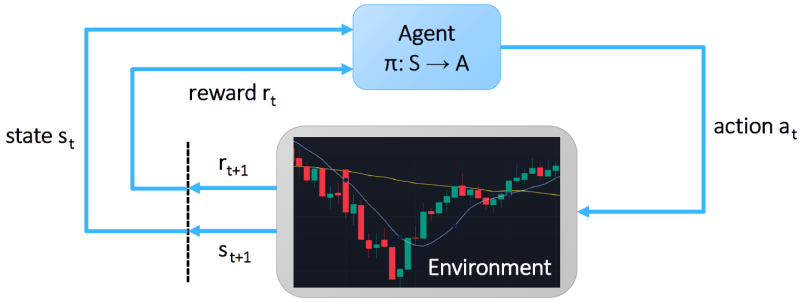



Fig. 4. Reinforcement learning framework (adapted from Sutton and Barto [175]).

action for each state. If $J(\boldsymbol{\theta})$ is differentiable with respect to $\boldsymbol{\theta}$, good values of $\boldsymbol{\theta}$ can be searched using the following stochastic gradient ascent updates:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t), \tag{4}$$

where $\alpha$ denotes the learning rate. The policy gradient (PG) theorem states that:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \sum_{s \in S} \mu^{\pi_\theta}(s, s_0) \sum_{a \in A} Q^{\pi_\theta}(s, a) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s). \tag{5}$$

Various PG algorithms [176] have been proposed using the foundation laid by the PG theorem. Suppose $\pi_\theta$ is used to generate a trajectory $\langle s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T \rangle$, where $s_T$ denotes the terminal state of the episode. The vanilla PG method estimates the gradient using:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{T-1} Q^{\pi_\theta}(s_t, a_t) \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(a_t|s_t) \right]. \tag{6}$$

Although there is no bias in vanilla PG updates, they introduce a high variance. REINFORCE [192] uses the returns estimated by Monte-Carlo methods to compute $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. For reducing the variance of PG methods, a baseline like $V^{\pi_\theta}(s)$ is subtracted from $Q^{\pi_\theta}(s, a)$ in Eq. (6), giving rise to the advantage function $A^{\pi_\theta}(s, a)$ defined as:

$$A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s). \tag{7}$$

Trust region policy optimisation (TRPO) [157] focuses on improving training stability by imposing a KL-divergence constraint on the size of the PG parameter update, which ensures that policy does not change significantly in a single step. Proximal policy optimisation (PPO) [158] simplifies the TRPO constraint by using a clipped surrogate function. Actor-critic approaches [97] combine value-based approaches and policy-based approaches to get the best of both worlds. Some popular examples of actor-critic methods include advantage actor-critic (A2C) [38], asynchronous advantage actor-critic (A3C) [126] and soft actor-critic (SAC) [70].

Traditional RL approaches maintain value function and policy using tables that are updated as the agent accumulates more experience by interacting with the environment. However, when dealing with large and continuous state or action spaces, tabular approaches become infeasible. In such complex scenarios, function approximators are needed. Neural networks are one of the most popular function approximators. The use of neural networks in reinforcement learning gave rise to deep reinforcement learning (DRL) [58]. DRL uses neural networks for maintaining value function and

policy. Recently, the usage of DRL has resulted in many successful breakthroughs in various domains [127, 161–163]. A taxonomy of RL algorithms is given in Appendix D while Appendix E presents a summary of the various design choices made in the literature for state features, actions, reward functions, etc. to tackle trading and PO.

### 2.2 Evolutionary computation

Evolutionary computation [54] approaches are population-based search metaheuristics [149] that work even when the objective function is non-linear, non-convex, non-differentiable and non-unimodal. EC algorithms take inspiration from biological evolution to guide the search process. We depict the typical flow of an EC algorithm in Fig. 5. EC approaches begin with a population of $P$ randomly generated candidate solutions $\{\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^P}\}$ where $\mathbf{x^i} = \langle x_1^i, x_2^i, \ldots, x_D^i \rangle$ denotes the $i^{th}$ candidate solution in the $D-$dimensional search space. In evolutionary terminology, a candidate solution $\mathbf{x^i}$ is referred as a chromosome with $D$ genes $\{x_1^i, x_2^i, \ldots, x_D^i\}$. The quality of the candidate solutions is evaluated using a fitness function $f$ that maps good solutions to higher fitness values while bad solutions are assigned lower fitness values. Suppose an EC algorithm is used to minimise the sphere objective function, $g(\mathbf{x}) = \sum_{j=1}^{D}(x_j)^2$, whose optimal value lies at the origin. In this case, the fitness function can be formulated as $f(\mathbf{x}) = -||\mathbf{x}||_2$ where $||\mathbf{x}||_2$ denotes the $l_2$-norm. In EC algorithms, the next generation of candidate solutions is derived from the current generation of candidate solutions by applying two evolutionary operators: crossover and mutation. The crossover and mutation operators are inspired by biological crossover and mutation. The crossover operator, also called the recombination operator, takes two candidate solutions and combines their genes to produce two new candidate solutions. The mutation operator takes one candidate solution and perturbs some of its genes to produce a new candidate solution. Suppose the candidate solutions are represented as binary vectors. In this case, the crossover operator splits two candidate solutions at the same point and exchanges their tails while the mutation operator simply flips some of the bits of the candidate solution. We refer the reader to Eiben and Smith [54] for further details. Over the generations, the fitness of candidate solutions keeps improving and eventually, the candidate solution with the best fitness in the last generation is chosen as the final solution. Some common stopping criteria include specifying the maximum number of generations, terminating when the fitness improvement saturates, etc. The biggest advantages of EC algorithms include gradient-free optimisation, inherent parallelisation and excellent empirical performance.

There are different types of EC algorithms based on the data structure used for the representation of candidate solutions. Genetic algorithm (GA) [81] is inspired by the biological evolution of chromosomes and is used to solve optimisation problems by evolving a population of vectors. Genetic programming (GP) [98] extends the expressibility of GA by replacing vectors with tree structures. However, GP faces scalability issues as the search space grows exponentially with an increase in the depth of the tree. Genetic network programming (GNP) [76] is an improvement over GP which uses directed graphs to represent candidate solutions instead of trees. The directed graph structures used by GNP are quite compact as nodes can be reused and connections between nodes exist only if necessary. GNP node transitions begin from a designated start node and continue on the basis of directed edges. Thus, the directed graph structure of GNP acts as an implicit memory function and gives it the capability to deal with repetitive processes. There are two types of nodes in GNP: judgement nodes and processing nodes. Judgement nodes are meant just for determining the next node while processing nodes execute the actual actions like buying or selling stocks. There are no terminal nodes in GNP so time delays are used to implement termination criteria.

Differential evolution (DE) [171] is a popular EC algorithm that uses distance and direction information from the current population to guide the search process. Evolutionary strategies (ES) [16] form another popular class of EC techniques that perform self-adaptation of hyperparameters by incorporating hyperparameters alongside parameters

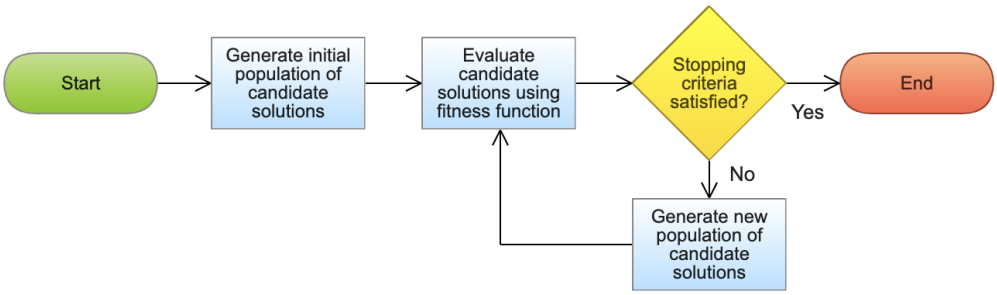



Fig. 5. Typical flow of an EC algorithm (adapted from Kachitvichyanukul [90]).

in the solution representation. Apart from biological evolution, researchers have also taken inspiration from natural processes like group behaviour of living organisms to propose swarm intelligence algorithms like particle swarm optimisation (PSO) [94]. We provide a taxonomy of nature-inspired algorithms in Appendix D. Additionally, artificial processes like annealing from metallurgy have been used as inspiration to propose algorithms like simulated annealing (SA) [96]. It is worth noting that SA is a point-based search algorithm, unlike EC algorithms which use a population of candidate solutions.

### 2.3 Finance

Traders and investors are two important elements of the finance domain. Traders indulge in short-term trades while investors focus on long-term investments. Investors base their decisions on fundamental analysis while the decision-making of traders involves technical analysis. Fundamental analysis [67, 100] deals with estimating the intrinsic value of financial assets. This involves the study of the economy, industry conditions, business, financial reports, etc. Technical analysis [3, 134] involves studying price charts of financial assets to identify patterns and trends using a variety of technical indicators like moving average, momentum, etc. Technical indicators are functions of prices of financial assets or other market data like trading volumes, open interest, etc. Technical indicators help in discovering trends in prices, reversal of trends, and various fluctuations in prices. Some indicators work better in trending markets while others work better in neutral markets. Fundamental analysis and technical analysis aid in decision-making by predicting the rise or fall of prices in the upcoming future. Both fundamental analysis and technical analysis are tedious tasks and require a lot of time and effort. Many people hire a financial advisor to assist them in trading or investment-related decisions. Traders and investors consider a wide variety of financial assets like currencies, stocks (also called equities), indices, exchange-traded funds (ETF), bonds, commodities, and derivatives (Fig. 6). Researchers have shown that the risk associated with a portfolio of assets is lower than that of an individual asset provided that the assets are not perfectly correlated [206]. This is an intuitive implication of Jensen's inequality which claims that the mean of the payoffs will always be larger than or equal to the payoff of the mean outcome. Due to this, portfolios are constructed in a manner that leads to maximum diversification of chosen financial assets.

Portfolio optimisation (PO) is a challenging task of capital allocation among a portfolio of assets to maximise risk-adjusted returns. Suppose the portfolio contains $N$ assets which are to be managed for a horizon of $T_H$ time-periods. The duration of the period can vary from a day to a year based on the preferences of the fund manager. A cash reserve is also maintained and is typically considered part of the portfolio. Each asset $1 \leq i \leq N$ has a price $p_i^t$ corresponding to

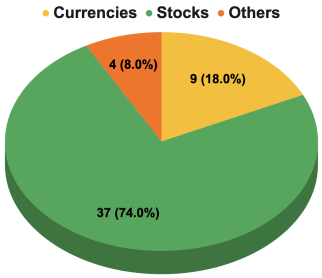



Fig. 6. Choice of financial assets used for testing performance in the surveyed papers.

each period $1 \leq t \leq T_H$. For each period $1 \leq t \leq T_H$, the decision-making task in PO is to determine $n_i^t$, the number of units of asset $i$ to have in the portfolio for the period $t$. If $n_i^t > n_i^{t-1}$, then $(n_i^t - n_i^{t-1})$ units of asset $i$ are bought by spending $(n_i^t - n_i^{t-1}) \cdot p_i^t$. If $n_i^t < n_i^{t-1}$, then $(n_i^{t-1} - n_i^t)$ units of asset $i$ are sold resulting in earnings of $(n_i^{t-1} - n_i^t) \cdot p_i^t$. In both scenarios, transaction cost $c_i^t = c \cdot |n_i^t - n_i^{t-1}| \cdot p_i^t$ is incurred, where $c$ is a market-dependent constant. Transaction costs are typically expressed in basis points (1 bp = 0.01%). The need for rebalancing portfolios stems from the varying rates at which asset prices grow. After period $t$, the portfolio's total value is

$$R^t = \sum_{i=1}^{N} n_i^t \cdot p_i^t - \sum_{i=1}^{N} \sum_{t'=1}^{t} c_i^{t'}. \tag{8}$$

Conventionally, the portfolio for period $t$ is represented as a vector of non-negative weights $\langle w_1^t, w_2^t, \ldots, w_N^t \rangle$, where $\sum_{i=1}^{N} w_i^t = 1$ and $w_i^t = \frac{n_i^t \cdot p_i^t}{R^t}$ represents the fraction of the portfolio's value corresponding to asset $i$. For each period $t$, the main task in PO is to specify the portfolio weight vector $\langle w_1^t, w_2^t, \ldots, w_N^t \rangle$. The portfolio return corresponding to period $t$ is defined as $r^t = \frac{R^t - R^{t-1}}{R^{t-1}}$. The main objective of PO is to maximise the portfolio returns and simultaneously minimise the associated risk. Risk is typically measured in terms of the standard deviation of portfolio returns and drawdowns. Consider a non-increasing sequence of portfolio values $\langle R^{t_1}, R^{t_1+1}, R^{t_1+2}, \ldots, R^{t_2} \rangle$ for $0 \leq t_1 < t_2 \leq T_H$. A drawdown measures the biggest fall in portfolio value and is defined as:

$$Drawdown = \frac{R^{t_1} - R^{t_2}}{R^{t_1}}. \tag{9}$$

A wide variety of metrics have been proposed for measuring risk-adjusted return. Let $\bar{r}$ denote the average return of the portfolio and $r_f$ denote the risk-free rate. The risk-free rate indicates the return rate on risk-free assets like government bonds. Next, we give a brief description of some popular risk-adjusted return metrics.

**Sharpe Ratio:** It is one of the most popular measures of risk-adjusted return. It indicates the excess return achieved by the portfolio, in comparison to a risk-free asset, per unit risk. Sharpe ratio [160] uses the standard deviation of returns as a measure of risk and is defined as:

$$Sharpe\ ratio = \frac{\bar{r} - r_f}{\sigma}, \tag{10}$$

where $\sigma$ denotes the standard deviation of portfolio returns.

**Sortino Ratio:** It is defined similarly to the Sharpe ratio except that it uses downside deviation as the risk measure instead of standard deviation. Downside deviation is defined as the standard deviation of portfolio returns that are

negative. Sortino ratio is defined as:

$$Sortino\ ratio = \frac{\bar{r} - r_f}{\sigma^-}, \tag{11}$$

where $\sigma^-$ denotes the downside deviation of portfolio returns.

**Calmar Ratio:** It uses maximum drawdown as the risk measure instead of deviation of returns. Maximum drawdown refers to the largest drawdown encountered during the course of investment. Calmar ratio is defined as:

$$Calmar\ ratio = \frac{\bar{r} - r_f}{D_{max}}, \tag{12}$$

where $D_{max}$ denotes the maximum drawdown.

**Sterling Ratio:** It is similar to the Calmar ratio except that it uses the average drawdown as a measure of risk instead of the maximum drawdown. Average drawdown refers to the average of all drawdowns encountered during the course of investment. Sterling ratio is defined as:

$$Sterling\ ratio = \frac{\bar{r} - r_f}{D_{avg}}, \tag{13}$$

where $D_{avg}$ denotes the average drawdown.

Even after diversification, determining optimal asset proportions in a portfolio is challenging due to the dynamic nature of financial markets, resulting in uncertainty around future asset prices. Negatively correlated assets might become positively correlated later due to mergers and acquisitions leading to an extra element of uncertainty in PO. Furthermore, the possible choices for constructing a portfolio are humongous: stocks, bonds, commodities, currencies, derivatives, etc.

Despite the aforementioned advantages of diversification, many practitioners participate in trading on a single financial asset. Trading can be viewed as a special case of PO with the portfolio containing just a single financial asset alongside a cash reserve. PO is the general case of trading in which we trade on more than one asset. For each period $1 \leq t \leq T_H$, the decision-making task in trading is determining $n^t$, the number of units of the asset to own for the period $t$. Trading is modelled as a continuous control task in terms of number of units of asset to trade, $\Delta_{n^t} = n^t - n^{t-1}$, for each period $1 \leq t \leq T_H$, to maximise risk-adjusted return. Some works consider $\Delta_{n^t}$ to be a constant quantity and formulate trading as a discrete control problem with the action set $\{buy, sell, hold\}$. Traders enter a long position when they make a purchase, a short position when they make a sale and a neutral position when they are holding zero units of a financial asset.

Trading and PO are control problems where the task is determining optimal actions at each time step to maximise long-term gains. Trading and PO are challenging tasks due to the uncertainty surrounding future asset prices. The prices of assets are not only influenced by demand and supply but also by a wide variety of external factors like government policies, geo-political situations, crude oil, currency exchange rates, pandemics, wars, etc. Quite often, a practitioner buys an asset and its price decreases and vice versa. Financial markets are dynamic and contain a significant amount of noisy data, adding to the complexity of trading and PO. Additionally, there are practical constraints like transaction costs, capital gain taxes, slippage in prices due to order placement delays, etc. which eat up a chunk of profits, making it hard to generate profits consistently.

PO has been modelled using a wide variety of approaches, with many originating from the finance domain itself. The fixed allocation ratio strategy focuses on maintaining predetermined allocation ratios, based on risk preferences, by rebalancing the portfolio once or twice a year. The maximum diversification strategy maximises the diversity of a portfolio by selecting minimally correlated assets. Stochastic portfolio theory focuses on constructing portfolios that have the capability of outperforming benchmark market indices like the Standard and Poor's 500 (S&P 500). The

mean-variance (MV) portfolio optimisation [122], introduced by Markowitz in 1952, is a classical 2-step approach for PO. Firstly, it estimates the expected returns and obtains a covariance matrix of the prices of financial assets. Then, it either maximises the returns for a given risk level or minimises the risk for a specified portfolio return to obtain profitable portfolio allocations. However, it makes a lot of unrealistic assumptions like zero transaction costs, normal distribution of returns, etc. Also, estimating expected returns is a challenging task.

Many trading strategies have been proposed by practitioners [40], predominantly based on technical analysis. Some prominent ones include trend-following strategies, mean-reverting strategies, etc. Researchers have attempted to model trading using different approaches. Many traditional approaches integrate domain knowledge and formulate financial problems in a stochastic control framework. These problems are solved analytically by making a lot of simplistic assumptions. Forecasting approaches have been used extensively for predicting the prices of financial assets over fixed horizons. Forecasting approaches can provide predictive signals but they do not model trading positions directly. Mapping of predicted prices to actual trading positions is an overhead. RL can model trading positions as actions and output them directly, bypassing the forecasting step.

### 2.4 Reinforcement learning in finance

Researchers have used RL to model various financial applications [59, 141, 180]. Moody et al. [129–132] introduce recurrent reinforcement learning (RRL) for trading and PO, representing policy using a recurrent neural network (RNN). They use asset prices as input features and show that policy-based approaches perform better than value-based approaches like Q-learning. Their work has inspired numerous follow-up investigations [2, 6, 120]. Recently, DRL approaches have been used extensively to tackle trading and PO. Zhang et al. [202] use deep Q-network (DQN), PG and A2C for trading. Liu et al. [108] propose multiple PG-based DRL agents like A2C, PPO, DDPG, SAC, etc. for trading and PO. Yang et al. [195] use an ensemble of A2C, PPO and DDPG agents for stock trading. Sood et al. [168] present a comparative study between DRL approaches and the MV model for PO. We refer the reader to some recent surveys [57, 71, 125, 133, 174] for further details on applications of RL in finance.

Gradient-based RL approaches for trading and PO need the objective functions to be differentiable, which becomes challenging to obtain when incorporating practical considerations like risk preferences, transaction costs, etc. Depending on their risk appetite, some users impose thresholds on the maximum weight of an asset in the portfolio. Incorporating transaction costs also introduces non-differentiability in the objective function due to the usage of the absolute value function. Additionally, some risk-adjusted return metrics are not always differentiable and their approximations have to be used. EC can help RL in this aspect due to its ability to deal with non-differentiable objective functions. RL algorithms produce a single solution while EC algorithms can produce a population of solutions, which gives users the flexibility to choose between multiple solutions. Additionally, EC can also help RL in narrowing down feature sets, evolving trading agents, searching hyperparameters, etc.

### 2.5 Evolutionary computation in finance

Similar to RL, researchers have used EC for various financial applications [86]. Trading has been tackled using EC to discover trading rules. Neely et al. [135] use GP to develop trading rules based on technical indicators. The trading rules demonstrate good performance on different currency datasets. Dempster et al. [49] use GA to discover trading rules at regular intervals for better adaptation to dynamic financial markets. Hu et al. [85] provide a detailed survey on the use of EC for discovering trading rules. Yelleti et al. [183] use multi-objective EC algorithms to develop trading strategies using technical indicators. PO has been modelled using single-objective [23, 184] as well as multi-objective EC algorithms

[145, 164], that focus on conflicting objectives like maximising portfolio return and minimising the associated risk. Many works formulate PO as a single-period investment problem, neglecting the timely rebalancing component. RL can assist EC in this aspect due to its ability to handle sequential decision-making through its agent-environment framework. Additionally, RL can help EC in adapting hyperparameters, balancing the exploration-exploitation trade-off, etc.

### 2.6 Hybridisation of RL and EC

The performance of EC algorithms depends on their hyperparameters, which are often problem-sensitive, resulting in poor generalisation. Conversely, RL offers good generalisation but its performance deteriorates in the absence of domain knowledge [166]. RL is online and can handle changes in the environment. On the other hand, EC is offline by default and the performance degrades when dealing with dynamic environments. However, the biggest advantage of EC is its simplicity and inherent parallelisation. Salimans et al. [155] show that ES can be used as a scalable alternative to RL. Due to their complementary nature, there is an incentive to hybridise RL and EC to get the best of both worlds. Numerous studies demonstrate that EC can improve the performance of RL and vice versa. EC has been shown to improve RL in aspects related to function approximation [190], directed exploration [41], generating interpretable policies [74], improving sample efficiency [95], enhancing stability [137], vanishing gradients [147], improving scalability through parallelisation [155], etc. On the other hand, RL has been demonstrated to assist EC in hyperparameter adaptation [178], balancing exploration and exploitation [190], genetic locus selection [106], noisy optimisation [150], etc. EC+RL hybrids have been successfully used in many applications [19, 104, 148, 166, 189]. We refer the reader to some recent surveys [11, 53, 167, 191, 205] for further details on EC+RL hybrids.

## 3 EC IMPROVING RL IN FINANCE - TIGHTLY COUPLED APPROACHES

Many practical objective functions in finance are non-differentiable; making it infeasible to use gradient-based RL approaches like PG and its variants. Consequently, researchers have used EC for gradient-free policy search (Section 3.1). The finance domain comprises various features like prices, trading volumes, technical indicators, fundamental features, macroeconomic variables, etc. The selection of state features heavily influences the profitability of RL-based agents for trading and PO. In Section 3.2, we discuss various works that use EC to select an appropriate set of state features for RL. Financial environments are non-stationary and hence, strategies that worked in the past might cease to be profitable. Also, a group of strategies tends to provide more robust judgement compared to a single strategy. Accordingly, researchers have used EC to evolve a population of trading agents (Section 3.3) and rules for asset selection in PO (Section 3.4). Lastly, EC has also been used to control overfitting in RL-based trading systems (Section 3.5). In all these hybridisation approaches, EC is an integral component and RL cannot function in disjunction with EC. Hence, this section surveys approaches where EC improves RL in a tightly coupled manner for financial applications like trading and PO (Fig. 3). We present a summary of such techniques in Table 1.

### 3.1 Gradient-free policy search

Value-based RL approaches face issues related to convergence, the curse of dimensionality, etc., especially in continuous action spaces. RRL [130] is a policy-based approach that manages to avoid all these issues related to value-based approaches. However, it has issues of its own like vanishing gradients, exploding gradients, etc., arising due to the use of backpropagation through time (BPTT) [188] in recurrent neural network (RNN) structure [51]. Many practical objective functions in finance are non-differentiable. RNNs also face difficulties in modelling long sequences [15]. Consequently,

Table 1. Summary of approaches where EC improves RL in a tightly coupled manner for financial problems

| EC Component | RL Component | Improved Aspect |
|---|---|---|
| Evolino | Recurrent RL | Gradient-free policy search [111] |
| Genetic Algorithm | MADDPG | Gradient-free policy search [119] |
| Evolutionary Strategies | Policy Search | Gradient-free policy search [5, 69] |
| Genetic Algorithm | Q-learning | State feature selection [9, 13, 50] |
| Genetic Algorithm | Deep Q-learning | Evolving population of trading agents [77, 78] |
| Genetic Algorithm | RL | Evolving hierarchy of agents [159] |
| Genetic Programming | RL | Evolving rules for asset selection [22] |
| Genetic Algorithm | Recurrent RL | Controlling overfitting [199–201] |

gradient-free RL [147] approaches have gained significant attention. Lu et al. [111] propose the use of gradient-free approaches like ES [17], Nelder-Mead method [136] in place of BPTT. The authors use Evolino [156], an EC approach that computes optimal mappings between the hidden-layer nodes and the output while evolving the weights connected to the hidden-layer nodes. The authors use dropout [170] to control overfitting and long short-term memory (LSTM) [80] networks to overcome the limitations of RNNs. The proposed trading system is more profitable using the Sortino ratio as the objective function, instead of the Sharpe ratio, when tested on 30-minute interval data from the USD/GBP exchange rate. Although the authors incorporate transaction costs, they consider fixed position size ($\Delta_{n^t}$ is constant), which is quite simplistic.

PG methods like multi-agent deep deterministic policy gradient (MADDPG) [110] face the issue of flat gradients, resulting in convergence-related issues. Maree et al. [119] use GA instead of gradient-based optimisation to fix the issue of flat gradients in MADDPG. The proposed system employs four different agents for PO: three for modelling profit, risk, and sustainability while the last for managing the other three agents. The three agents produce portfolio allocation $\langle w_1^t, w_2^t, \ldots, w_N^t \rangle$ as actions while the final action is determined by the manager agent based on their weighted average. All three agents use different reward functions. The agent modelling profit uses the step-wise change in portfolio value, $R^t - R^{t-1}$, as the reward while the agent modelling risk utilises the Sharpe ratio as the reward. For the agent modelling sustainability, a novel reward is introduced based on the environmental, social, and governance (ESG) score [37] of the portfolio assets. ESG score models sustainability and is used as a long-term risk indicator. The reward of the manager agent is formulated as a weighted combination of the rewards of the other three agents. The GA-based approach significantly outperforms vanilla MADDPG on a portfolio of three Dow Jones Industrial Average (DJIA) stocks in terms of profitability, risk, and sustainability. Some limitations of this work include ignoring transaction costs ($c = 0$) and the small size of the portfolio ($N = 3$).

Many existing approaches for PO assume stationarity although financial markets are non-stationary. Ha et al. [69] propose an evolutionary meta-RL approach for PO to deal with the non-stationarity of financial markets. Their approach slices the long-term trading activity into multiple short-term activities to improve the adaptation to the dynamic financial markets. In addition, it has been shown that EC approaches can compete with conventional gradient-based approaches to train deep neural networks for RL [173]. Ha et al. [69] use ES instead of conventional policy-based RL methods. Their approach produces better results than multiple baselines on cryptocurrency data. Ala et al. [5] use ES to avoid gradient-related issues in policy-based approaches for PO.

The use of EC algorithms for training policy networks not only helps in addressing gradient-related issues of PG methods but also improves robustness to policy initialisation, noise, sparse rewards, long horizons and action frequency

[73, 147, 155]. The choice of objective functions is no longer restricted to the class of differentiable functions. Also, the inherent parallelisation of EC algorithms enables speed-up in training time and enhances scalability. Additionally, gradient-free policy search using EC algorithms opens up the possibility of utilising well-established neuroevolution approaches that can improve the performance even further [173]. However, care should be taken as some EC algorithms are prone to entrapment in local optima.

### 3.2 Selecting state features for RL using EC

Selecting suitable state features for RL is challenging due to the abundance of features in finance: prices, volumes, technical indicators, fundamental features, etc. Financial features are typically unable to generate profits when used in isolation [49]. Hence, a combination of financial features is used to generate trading signals. Many works simply use a combination of popular technical indicators alongside prices as state features without giving a reason behind their choice [185, 193, 202]. EC has been shown to assist RL in selecting state features for trading. Dempster et al. [50] have made one of the initial attempts to use EC+RL hybrids for trading. In their previous work [48], the authors use Q-learning [186] and GP [140] for foreign exchange (FX) trading. However, the Q-learning approach exhibits overfitting leading to poor generalisation. To address this issue, the authors propose evolutionary reinforcement learning (ERL) for constraining the inputs to the RL module [50]. GA [81] is used to determine the best possible subset of technical indicators to be used as state features in the RL module. Eight technical indicators with parameters recommended by Achelis et al. [3] are considered for selection. The genes of chromosomes (refer Section 2.2) are represented as binary variables indicating the inclusion or exclusion of corresponding technical indicators. GA is used to search the space of 8-dimensional binary vectors. The fitness function of GA is chosen as the returns generated by the RL module in the evaluation period using the indicators with gene value 1 as state features. Here, EC and RL are tightly coupled as the GA module invokes the RL evaluator for fitness calculation (Fig. 7). This has also been done in a loosely coupled manner as discussed in Section 4.1. The indicators having gene value 1 in the best solution vector found by GA are included as state features for RL during testing. The authors consider two action sets for RL: $\{buy, sell\}$ and $\{buy, sell, neutral\}$. The difference in portfolio value between consecutive time steps, $R^t - R^{t-1}$, is chosen as the reward. ERL approach outperforms vanilla RL for both action sets on 1-minute interval data of three FX rates: GBP/USD, USD/CHF, and USD/JPY. The proposed system generates profits at transaction costs corresponding to $c \leq 4$ bp and the returns are mixed at $c = 10$ bp. The importance of neutral action is also demonstrated in the presence of transaction costs. Some results are shown to be statistically significant using a non-parametric binomial test. However, the authors do not incorporate risk in the objective function.

Bates et al. [13] incorporate indicators derived from the order book and order flow data alongside technical indicators. Twelve order book-based indicators are derived from net open orders of different types: take-profit orders, stop-loss orders, new orders, all orders, etc. These help to model demand and supply pressure in the market. Nine order flow-based indicators are derived from daily order volume generated by different types of customers: retail, institutional, speculative, non-speculative, etc. These help in modelling customer behaviour. The proposed system is tested on 3 FX rates: GBP/USD, USD/EUR and USD/JPY. Superior results are obtained even at high transaction costs corresponding to $c = 10$ bp, implying that there is some merit in considering indicators derived from order book and order flow data alongside technical indicators. Although the authors incorporate transaction costs, they consider the simple case of fixed position size ($\Delta_{n^t}$ is constant).

Austin et al. [9] include price series data alongside technical indicators and generate positive returns on 1-minute interval FX rates. However, the ERL system is profitable only up to transaction costs corresponding to $c = 2$ bp. Before

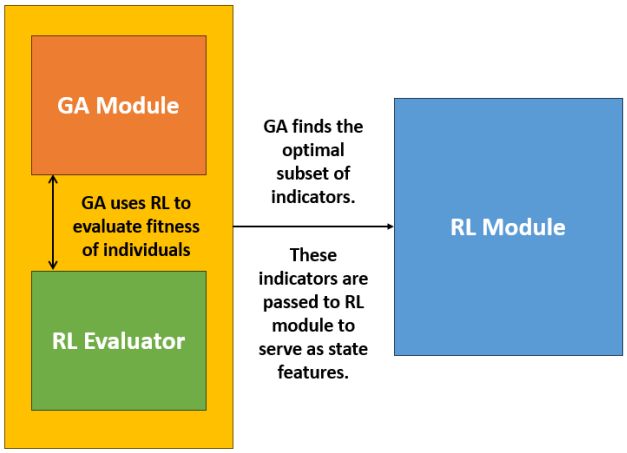



Fig. 7. Tightly coupled approach for state feature selection proposed by Dempster et al. [50].

integrating order book and order flow data, a statistical analysis is performed to demonstrate that FX rates are indeed influenced by order book and order flow data. Consequently, gross flow and net flow indicators are derived for four types of clients: institutional, corporate, speculative, and others. Twelve indicators are derived from order book data similar to Bates et al. [13]. Tests are conducted on daily data of three FX rates using order flow indicators, order book indicators, and technical indicators in isolation as well as in combination. When considered in isolation, order book indicators outperform order flow indicators which in turn outperform technical indicators. The combination of technical indicators with either order book indicators or order flow indicators produces profits even at high transaction costs corresponding to $c = 10$ bp. However, the combination of order book and order flow indicators is relatively less profitable demonstrating the importance of technical indicators. One limitation of this work is that order flow and order book data are not publicly available. The authors obtain this proprietary data through HSBC, an important market maker in FX markets.

Some advantages of using EC algorithms for feature subset selection include their simplicity and minimal requirements on the optimisation front. Also, EC offers flexibility to use any user-defined fitness function for feature subset selection; in contrast with approaches like principal component analysis (PCA) which work with their well-defined objective function. Additionally, the inherent parallelisation of EC algorithms offers increased scalability. However, care should be taken while choosing the feature set for subset selection using GA as adding an extra feature doubles the search space. Also, invoking the RL module for fitness calculation is computationally expensive.

### 3.3 Evolving population of trading agents

Developing RL-based autonomous agents for financial applications is challenging due to issues like reward sparsity, imperfect information, etc. Also, the actions of agents can impact the environment, especially in small markets. Hirchoua et al. [78] propose the evolution of a population of deep Q-learning (DQL) agents for trading that learn during their lifetime and transfer their knowledge to the next generation. This helps in handling situations that the agent has not encountered before. Additionally, each agent learns from a different dataset to maintain diversity and deal with various levels of risk and complexity [77]. State space in trading is considered to be partially observable and hence partially

observable Markov decision process (POMDP) [91] are used instead of usual MDP. Similar to GA [81], this approach starts with a population of randomly initialised agents. During their lifetime, the agents execute trades and based on the accumulated reward, a set of best agents are selected for the mutation phase. Elitism is incorporated which retains the best agent directly in the next generation. Over the generations, evolution enforces the rejection of risky tactics culminating in stable trading strategies. Eventually, the winner of the final generation is tested against baselines on eight stocks. The proposed system successfully navigates through six different economic crises as it refines its strategies by transferring knowledge instead of bothering about seasonal events [55]. Since this work uses a value-based RL approach, it faces issues related to convergence, the curse of dimensionality, etc., especially in continuous action spaces.

The human brain comprises dense local clusters of neurons capable of different functions [12]. These clusters specialise in learning a task by adaptation [169]. Serrano et al. [159] propose a hierarchical system for PO that mimics the human brain by combining different techniques. RL is used for making quick local decisions, neural networks serve as memory, cluster managers make global decisions and EC transfers learned information to future generations. The hierarchy of agents comprises asset bankers, market bankers, and a CEO banker. CEO banker is at the top of the hierarchy with various market bankers under it. Market bankers specialise in a particular type of market, like equities or bonds, and have a cluster of asset bankers under them. Market bankers can also specialise in the same type of market but at different risk levels. Asset bankers use RL to learn trading on different assets within the market. The market banker selects the best asset banker under him and eventually based on the selection of different market bankers, the CEO banker makes the final investment decisions at the portfolio level. Additionally, the entire hierarchy of agents keeps evolving and transferring their learning to future generations. The authors use a genetic learning approach where information is transmitted through network weights rather than the nodes themselves. It has been found that random genetic changes tend to be more successful than systematic changes as they improve generalisation [142]. Hence, random neural networks [60] are used which imitate the working of biological neurons more accurately than conventional neural networks. In these networks, signals are transmitted in a spiking manner as impulses rather than in an analog manner. The proposed system is evaluated on eight assets from bond and derivative markets and promising results are obtained. Although more robust judgement is obtained using a hierarchy of agents, heavy computation is required to evolve a population of agents.

### 3.4 Evolving rules for asset selection

While tackling PO, a real-world portfolio manager often goes through a three-step process. Firstly, the markets are decided from which assets are targeted for selection in the portfolio. Then, on a timely basis, a set of $N$ potentially profitable assets are selected in the portfolio. Lastly, the portfolio is rebalanced at some frequency to generate portfolio allocation $\langle w_1^t, w_2^t, \ldots, w_N^t \rangle$. Casanova [22] proposes the use of RL in conjunction with GP [98] to select a set of $N$ potentially profitable assets daily. A population of rules for discovering potentially profitable assets is initialised with each rule having a corresponding portfolio. This population of rules is evolved using portfolio return as the fitness function. The proposed system is tested on IBEX35 stocks and outperforms some of the best Spanish investment funds. This approach of pre-screening helps in avoiding bad financial assets becoming a part of the portfolio. Although the authors consider transaction costs corresponding to $c = 20$ bp in their work, their objective function does not incorporate risk. Another limitation of this approach is the heavy computation required to evolve a population of rules for asset selection. However, the inherent parallelisation ability of EC can help in reducing computation time. Additionally, the universe of financial assets is huge.

### 3.5 Controlling overfitting through elitism

RRL [132] has been used to develop many financial trading systems [2, 6, 120]. Researchers have found that RRL finds autocorrelations in price changes of high-frequency financial data and hence generates high profits. Most of the RRL-based research has been conducted for high-frequency trading on stocks and currencies [47, 64]. Very little research has been done on using RRL for low-frequency trading where the weak autocorrelation might reduce profits [66]. Also, most of the studies utilise technical analysis while ignoring fundamental analysis and, econometric analysis which hold high explanatory power, especially in low-frequency trading [100]. Zhang et al. [199] use fundamental and econometric indicators alongside technical indicators. The best subset of indicators is selected using GA and fed to the RRL module as state features. The search space of GA comprises a mixture of eight fundamental, technical, and econometric indicators. This approach of screening indicators using GA is similar to that of Dempster et al. [50] which uses chromosomes with binary genes to depict the inclusion or exclusion of corresponding indicators. The major contribution by Zhang et al. [199] is incorporating the concept of elitism by modifying RRL's parameter update scheme. In EC, elitism refers to preserving the best candidate solution from the previous generation into the next generation. Maringer et al. [120] advocate the use of a set of trading agents instead of just one trading agent to explore the search space more effectively. The trading agent with the best Sharpe ratio in the evaluation phase is termed an elitist trader. The elitist trader is outperformed by other trading agents in the testing phase indicating signs of overfitting. Average elitism refers to the use of a set of trading agents instead of just one trading agent and then using their average contributions for parameter updates. The GA-RRL trading system consists of 100 trading agents from which an elitist set of best-performing trading agents is selected based on Sharpe ratios obtained during the evaluation phase. All the trading agents in the elitist set contribute equally to the parameter updates in RRL. The proposed system significantly outperforms the buy-and-hold and random trading strategy on daily data of 238 S&P 500 stocks. Moreover, the average elitist scheme outperforms the single elitist trader. The results demonstrate that incorporating average elitism helps in controlling overfitting in RRL-based trading systems.

Zhang et al. [200] present another RRL parameter update scheme called multiple-elitist to deal with stocks with high correlation. In this scheme, the parameter updates for a particular stock involve the contribution from other correlated stocks besides its own contribution. The returns obtained by trading on S&P stocks are profitable and stable. Zhang et al. [201] extend their GA-RRL approach by incorporating volatility indicators and volume indicators [25] alongside technical indicators, fundamental indicators, and price series in the state of the RRL module. Average elitism is used to encounter overfitting. The trading system is evaluated on the daily data of 180 S&P stocks. Out of 100 simulation traders, five are selected into the elite set based on Sharpe ratios obtained during the evaluation phase. A variant of the Sharpe ratio is used as the fitness function for GA. The best chromosomes produced by the GA module are obtained for all 180 stocks. Testing is done in four scenarios based on how the indicators from the best chromosomes discovered by GA are fed to the RRL module. In the tailored scenario, the indicators from an asset's best chromosome are fed to RRL for that asset's testing. In the general scenario, the indicators that appeared in the best chromosomes of more than 90 stocks are fed to RRL for all the stocks. In the all-in scenario, all 10 indicators are fed to RRL while in the all-zero scenario, only the prices are fed to RRL without any indicator. The all-zero scenario is equivalent to the vanilla RRL approach [130]. Since the performance of RRL trading systems might depend on initial parameters, 100 different runs are performed for each stock. Transaction costs correspond to $c = 3$ bp. The GA-RRL trading system obtains superior performance compared to the random trading strategy. While the GA-RRL approach is unable to obtain a higher Sharpe ratio than the buy-and-hold strategy, it produces more stable results. Apart from the all-zero scenario, returns are

positive and stable in the other three scenarios establishing the merit in feeding indicators. The best performance of GA-RRL is obtained in the tailored and general scenario. Also, the standard deviations are higher in the all-in scenario compared to the tailored and general scenario implying that feeding all indicators leads to increased noise at the RRL terminal. Thus, a suitable combination of indicators should be used.

It is worth noting that fundamental and econometric indicators are relatively harder to obtain than technical indicators. Also, since the fundamental and econometric indicators are not updated very frequently, interpolation has to be applied. One big advantage of incorporating average elitism in EC is that it controls overfitting without compromising on the diversity of candidate solutions. Multiple elitism is useful in scenarios when the financial assets are highly correlated.

## 4 EC IMPROVING RL IN FINANCE - LOOSELY COUPLED APPROACHES

This section surveys approaches where EC improves the performance of RL in a loosely coupled manner for financial applications (Fig. 3). Recall that selecting appropriate state features for RL in financial applications is challenging due to the vast number of available features encompassing prices, volumes, technical indicators, fundamental features and macroeconomic variables. In Section 3.2, we surveyed approaches where EC is used to select a suitable set of state features for RL in a tightly coupled manner. Researchers have also used EC to select state features for RL in a loosely coupled manner (Section 4.1). The performance of RL is impacted by the choice of hyperparameters. Due to the dynamic nature of financial environments, the RL hyperparameters have been searched using EC approaches on a timely basis (Section 4.2). EC has also been used to tune technical indicators for RL-based trading agents (Section 4.3) and label generation (Section 4.4). A summary of such techniques is presented in Table 2.

### 4.1 Selecting state features for RL using EC

Human traders use a set of rules derived from technical indicators to formulate their trading strategies. This activity is fairly individualised as traders use distinct sets of indicators and interpret the signals differently. Since financial markets are non-stationary, trading rules exhibit varying effectiveness at different times. Also, different indicators are useful in different financial markets. Hence, optimal trading rules vary according to the situation and a rule is not guaranteed to be profitable in all scenarios. Usually, a set of trading rules is used to determine the final decision. A trading strategy comprises entry and exit rules. Entry rules dictate the conditions of entering a trade (taking a long or short position) while the exit rules indicate when to return to the neutral position. Hryshko et al. [82] propose a hybrid trading system that uses GA to search the space of trading strategies formed using subsets of 10 popular technical indicators. The indicators present in the best strategy discovered by GA serve as state features for the RL component based on Q-learning. The intuition behind this choice is that if the trading strategy based on a certain set of indicators is found to be the best by the GA then those indicators are capable of predicting prices. The action set considered is $\{buy, hold, sell\}$ while the difference in portfolio value between consecutive time steps, $R^t - R^{t-1}$, serves as the reward. The trading system is tested in FX markets on 5-minute interval data of the EUR/USD exchange rate and generates moderate profits at low transaction costs ($c = 2$ to 4 bp). However, later the system generates very meagre profits implying that old technical indicators have become obsolete due to changed market conditions and the GA module needs to be run again. The loosely coupled hybrid trading system proposed by Hryshko et al. [82] differs in two ways from the tightly coupled approach of Dempster et al. [50]. Firstly, the chromosomes used by GA of Hryshko et al. denote trading rules based on conditions dictated by indicators and are of variable length while the chromosomes used by Dempster et al. depict the inclusion or exclusion of indicators and are encoded as binary vectors of fixed length.

Table 2. Summary of approaches where EC improves RL in a loosely coupled manner for financial problems

| EC Component | RL Component | Improved Aspect |
|---|---|---|
| Genetic Algorithm | Q-learning | State feature selection [82–84] |
| Simulated Annealing | RL | State feature selection [88] |
| Differential Evolution | Recurrent RL | Hyperparameter search [121] |
| Genetic Algorithm | Recurrent RL | Hyperparameter adaptation [165] |
| Particle Swarm Optimisation | RL | Tuning technical indicators [21] |
| Differential Evolution | Deep Q-learning | Label generation [44] |

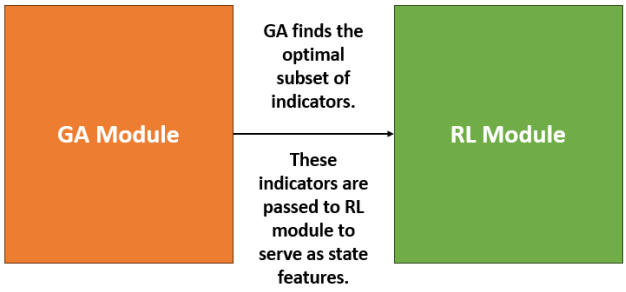



Fig. 8. Loosely coupled approach for state feature selection proposed by Hryshko et al. [82].

Secondly, the fitness function of Dempster et al. (Fig. 7) uses the returns generated by the RL module in the evaluation phase while Hryshko et al. do not invoke the RL module for fitness calculation (Fig. 8). Since the trading rules are capable of executing trades themselves, the Sterling ratio is used as the fitness function. This makes the loosely coupled approach of Hryshko et al. computationally lightweight. This is particularly helpful in finance as the non-stationary nature of environments requires the GA module to be invoked on a timely basis. Some limitations of this work include the simplistic assumption of fixed position size ($\Delta_{n^t}$ is constant) and ignoring risk in the objective function of RL.

Hryshko et al. [83, 84] extend their hybrid trading system by introducing online and offline modes. In online mode, the trading system executes trades without human intervention while in offline mode, the trading system provides analysis to the human trader who makes final trading decisions. Forecasts can also be used as state features in RL. Jamali et al. [88] propose a trading system that combines various techniques like RL, SA [96], multiple regression, and technical analysis for FX trading. Multiple regression is used to forecast the FX rates for the next 15 days. SA is used to refine the forecasts as long forecasts are often less accurate. RL module uses the forecasts as state features and generates trading actions. The final decision is taken based on the outputs of the RL module and technical analysis performed using the relative strength index (RSI). If both RL and RSI indicate buy or sell, then only assets are bought or sold. Otherwise, if RL and RSI disagree, the action taken is hold. The trading system is tested on daily data from the EUR/USD exchange rate and generates small profits. Some limitations of this work include ignoring risk and transaction costs ($c = 0$).

### 4.2 Searching optimal hyperparameters

Maringer et al. [120] combine RRL with regime switching (RS), a statistical modelling technique, to deal with situations where different regimes characterise financial data. The RS-RRL system comprises a transition variable that is used to

make decisions about switching between regimes. Maringer et al. [121] experiment with 4 different choices of transition variable and the results show that using volume variable produces a higher Sharpe ratio compared to conditional volatility, implied volatility, and RSI indicator. DE [171] is used to search the optimal hyperparameters for the RS-RRL system. RRL's performance is influenced by its hyperparameter values. Song et al. [165] use GA for adapting RRL's hyperparameters. GA not only supplies good initial hyperparameters but also evolves them over time to adapt to the dynamic financial markets. One limitation of this approach is the heavy computation involved in adapting RRL hyperparameters using EC on a timely basis. However, computation time can be reduced using parallelisation, an inherent capability of EC algorithms.

### 4.3 Tuning technical indicators

Although financial environments are non-stationary, it has been observed that certain patterns are repetitive [109]. Thus, trading systems should not only adjust to market changes but also be able to recognise previously successful patterns. Technical indicators are functions of prices of financial assets or other market data like trading volumes, open interest, etc. Technical indicators help to discover price trends, reversal of trends, and various price fluctuations. Most technical indicators have parameters that are chosen based on financial wisdom. One popular example is the number of periods for which the moving average is calculated. These parameters can also be tuned using EC to choose suitable values. Bollinger Band [20] is a technical indicator that utilises the moving average and standard deviation of prices to generate trading signals. Butler et al. [21] use a modification of learning classifier systems (LCS) [179] to adapt a population of Bollinger Bands, instead of classifiers as in traditional LCS. These adaptive Bollinger Bands (ABBs) correspond to time periods of varying lengths. The authors propose a dynamic multi-objective variant of heterogeneous particle swarm optimisation (HPSO) [56] to tune the parameters of ABBs. The objectives are to maximise both the profits and trading activity. Similar to LCS, a match set of ABBs is selected based on trading performance and this match set is used to deduce the suitable action. Based on the action taken, a reward is obtained from the trading environment in terms of return percentage. The proposed system is tested on simulated as well as real data and is shown to outperform baselines like the buy-and-hold and canonical Bollinger Bands. This approach of tuning parameters of technical indicators using EC reduces the reliance on conventionally accepted values. Although the authors consider transaction costs corresponding to $c = 25$ bp, they do not incorporate risk in their work. Also, the objective of maximising trading activity results in higher transaction costs in aggregate. The advantage of this approach is that the proposed setup is general and can be extended to other indicators apart from Bollinger Bands. However, frequent tuning can make computation heavy.

### 4.4 Label generation in transfer learning based approach

Trading is a challenging application for RL due to convergence-related issues. To overcome these issues, Costa et al. [44] transform time series into images as this approach has previously shown potential in classification-based trading [39]. The previous 64 values of price, volume, RSI, and moving average convergence divergence (MACD) are transformed into 64×64 images and eventually concatenated into 4×64×64. DE [171] is used to optimise a parametric rule-based strategy. This strategy labels the images with buy, hold, or sell to maximise gaps between subsequent buys and sells. RL agents might take many episodes to converge so transfer learning is incorporated. Transfer learning [177, 187] is a learning paradigm in which knowledge acquired through one task is transferred to another. Transfer learning approaches take a model trained on one task and use it as a starting point to train a new model for another task. This is particularly helpful in improving convergence as training is done from a better starting point. A convolutional neural

network (CNN) based classifier is trained using the labelled images and then transfer learning is used from this trained supervised model to improve the convergence of the deep Q-learning model. A negative reward is introduced for hold action as the RL agent took hold action most of the time to minimise loss. The trading system is tested on hourly data of various stocks from the New York Stock Exchange (NYSE). However, the results are not very good as the trading system incurred losses on many stocks. Some limitations of this work include fixed position size ($\Delta_{n^t}$ is constant) and ignoring transaction costs ($c = 0$).

## 5 RL IMPROVING EC IN FINANCE - TIGHTLY COUPLED APPROACHES

This section surveys approaches where RL improves the performance of EC in a tightly coupled manner for financial applications (Fig. 3). Recall from Section 4.1 that different technical indicators are effective in different situations [82]. Technical indicators are often used in combination rather than in isolation [49]. Hence, trading rules that were successful in the past might become obsolete in future. There is a need for trading systems that can dynamically select appropriate technical indicators at different times. In Section 5.1, we survey approaches where RL is used to assist EC in selecting indicators dynamically for trading and PO. RL has also been used to generate trading signals for EC algorithms (Section 5.2). A summary of such techniques is presented in Table 3.

### 5.1 Selecting technical indicators dynamically

GNP [115] possesses the architectural capability to handle dynamic problems but it is offline by default. Hence, it has been combined with different RL techniques to integrate the online aspect. In these GNP-RL hybrids, GNP evolves directed graph structures while RL refines the graph structures produced by GNP. Evolution focuses on the diversification of candidate solutions while learning focuses on intensification to obtain high-quality solutions from promising regions of the search space. Mabu et al. [112, 113] propose GNP-AC for stock trading by combining GNP and actor-critic (AC) method. The importance index (IMX) is also introduced for technical indicators. IMX is a function that maps technical indicators to trading signals with 1 representing a strong buying signal and -1 representing a strong selling signal. AC is used to learn the optimal value of a threshold utilised to predict whether stock prices will go up or down based on IMX values. AC is used because it can handle continuous actions. The best solution obtained in the validation period is used for testing on 20 stocks from the Tokyo stock exchange and it demonstrates superior performance than the buy-and-hold baseline on 14 stocks. Chen et al. [29] combine GNP with SARSA for stock trading. Candlestick charts [139] are incorporated alongside technical indicators. Sub-nodes are introduced within GNP nodes and each of them is mapped to a technical indicator or candlestick pattern. Each sub-node comprises a next node, a corresponding delay, and a threshold for buying or selling. GNP-SARSA [27] uses time delays for determining the suitable number of technical indicators and candlestick patterns to be selected for making decisions of buying or selling in the processing nodes. The role of SARSA is to select the optimal sub-node in an $\epsilon$-greedy manner. Crossover and mutation (defined in Section 2.2) are used to evolve GNP graph structures. The hybrid technique [28] performs better than the buy-and-hold strategy on 14 out of 16 Tokyo exchange stocks. Chen et al. [32] also incorporate a method for adapting IMX functions through evolution and extend their work to make it real-time by using a sliding window approach [33].

GNP has also been used to tackle PO. In traditional GNP, there is a possibility of some nodes never getting used as it is not necessary to transition back to the start node. Chen et al. [31, 34, 35] introduce genetic network programming with control nodes (GNP-CN) by incorporating a set of control nodes for each asset in the portfolio. Node transitions start from control nodes and when a certain number of processing nodes have been executed, the transition happens to another control node. GNP control nodes represent the breadth of the search space while the depth of the search space

Table 3. Summary of approaches where RL improves EC in a tightly coupled manner for financial problems

| RL Component | EC Component | Improved Aspect |
|---|---|---|
| Actor-critic | GNP | Dynamic indicator selection [112, 113] |
| SARSA | GNP | Dynamic indicator selection [27–29, 32] |
| SARSA | GNP-RA | Dynamic indicator selection [114, 116, 117] |
| SARSA | GNP | Dynamic indicator selection in real-time [33] |
| SARSA | GNP-CN | Dynamic indicator selection for PO [31, 34, 35] |
| SARSA | TA-GNP | Dynamic indicator selection for PO in real-time [26, 30] |
| RL | GA | Generating signals for portfolio assets [24] |
| Recurrent RL | PSO | Handling constraints in portfolio optimisation [7] |
| A3C | PSO | Multi-objective portfolio optimisation [138] |

is represented by the processing nodes activated by each control node. The proposed method is tested on 10 stocks from the Tokyo stock exchange and the portfolio approach generates more profits than the combined profits generated by applying traditional GNP-RL to the 10 stocks individually. This is because the capital gets redistributed in such a way that more capital is assigned to stocks generating more profits. Chen et al. further extend their work on PO by proposing time adapting genetic network programming (TA-GNP) [26, 30]. TA-GNP focuses on adapting the trading system to the changing market through a sliding window approach.

Mabu et al. [114, 116, 117] propose genetic network programming with rule accumulation (GNP-RA) by incorporating the concept of rule accumulation in GNP-RL trading systems. In GNP-RA, trading rules generated during evolution are stored and buying or selling decisions are made by taking contributions from rules produced across all generations instead of just the rules present in the last generation. This helps in controlling overfitting. Li et al. [68, 101, 196, 198] introduce the concept of subroutines where each subroutine behaves as a miniature version of GNP-RL. These subroutines can be executed parallelly during the evolution of the main GNP. This helps in improving the efficiency of GNP-RL trading systems. Xu et al. [194] incorporate pruning of redundant nodes in GNP-RL systems. Yang et al. [197] evolve a group of cooperating trading agents to improve the robustness of GNP-RL systems. Li et al. [102] utilise an estimation of distribution algorithm (EDA) to improve the scalability of stock trading systems. Chen et al. [36] integrate a statistical model with GNP-RL for stock trading. Bahar et al. [10] integrate fuzzy concepts in GNP-RL systems to generate trading signals. One major limitation of most of these works is ignoring transaction costs ($c = 0$). Also, since technical indicators are selected frequently, heavy computation is involved. However, effective parallelisation can be done to reduce training time.

### 5.2 Generating signals for EC algorithms

Chang et al. [24] use RL to generate trading signals for different stocks. These trading signals are used by GA to determine appropriate portfolio allocation $\langle w_1^t, w_2^t, \ldots, w_N^t \rangle$. In the real world, PO involves many constraints. A portfolio is often restricted to having a maximum of $N$ assets. This is known as the cardinality constraint and is imposed as it becomes difficult to manage a portfolio with a large number of assets. Quantity constraint imposes limits on the possible asset proportions: for each period $1 \leq t \leq T$, $\max_{i=1}^{N} w_i^t \leq w_{max}$, where $w_{max} \in (0, 1]$ denotes the maximum proportion of capital assigned to any asset in the portfolio. These limits help in maintaining diversity within the portfolio. Round-lot constraint restricts the number of shares of an asset to be an integer. This is necessary because there is no provision for buying a fractional quantity of an asset. Pre-assignment constraint means that the investor puts an asset into the

portfolio for the entire investment horizon. Quite often, there are assets whose value is expected to rise for a long time and investors want such assets in their portfolio. Class constraint places restrictions on each asset class to promote diversity. Solving constrained PO problems is quite challenging. Some researchers have used existing algorithms like GA, PSO, etc. while others have proposed novel EC algorithms specifically for constrained PO. Empirical studies claim that PSO works better for constrained PO [43]. However, PSO has been used to solve constrained PO for long-only portfolios that do not involve short selling by allowing negative weights. In short selling [153], when the price of a financial asset is high, its shares are borrowed and sold. Later, when the price of the asset comes down, the shares are purchased and returned to the lender. This entire process results in significant profits. It is worth noting that short selling is not allowed for fund managers in various countries.

Almahdi et al. [6] extend the traditional version of RRL for PO. However, their approach is meant for unconstrained PO. In their subsequent work, Almahdi et al. [7] use RRL in conjunction with 4 variants of PSO to tackle constrained PO: standard PSO, improved particle swarm optimisation (IPSO), drift particle swarm optimisation (DPSO), many optimisation liaisons particle swarm optimisation (MOLPSO). Different variants of PSO are used as each of them provides a different way of finding solutions. In all four PSO algorithms, the Calmar ratio is used as the fitness function because it is empirically found to be performing better. Calmar ratio is also used as the objective function for RRL instead of the usual Sharpe ratio. RRL is used to assist PSO in handling negative weights by generating short or long signals. The proposed constrained PO system achieves better performance than multiple benchmark constrained PO systems on S&P 100 stocks, especially when the transaction costs are high. EC algorithms are black-box optimisation approaches that ignore the structure of the problem while searching for solutions. Solving constrained PO is a challenging task and EC algorithms can face convergence-related issues. Nigatu [138] uses A3C in conjunction with PSO to deal with multi-objective PO. PSO is used to fine-tune the portfolio weights produced by A3C according to investors' needs. The authors consider the objectives of maximising returns and minimising risk. This is a departure from the majority of proposed works that use single-objective formulation to optimise risk-adjusted return metrics like the Sharpe ratio.

## 6 RL IMPROVING EC IN FINANCE - LOOSELY COUPLED APPROACHES

This section surveys approaches where RL improves the performance of EC in a loosely coupled manner for financial applications (Fig. 3). RL has been used to adapt hyperparameters of EC algorithms. Unlike the previous 3 categories, research conducted in this category is scarce. Possible reasons for limited research are the abundance of handcrafted hyperparameter adaptation strategies and the heavy computational demand imposed by RL algorithms for supplying hyperparameter values at each time step. The performance of EC algorithms depends on the choice of their hyperparameters. Coming up with the optimal hyperparameter values of EC algorithms is challenging. These hyperparameters are often problem-sensitive resulting in poor generalisation capability of EC algorithms. On the other hand, RL offers good generalisation but its performance deteriorates in the absence of domain knowledge [166]. There is an incentive in combining EC and RL to get the best of both worlds. Jia et al. [89] use PG to adapt the hyperparameters of PSO and the resulting PG-PSO hybrid is used for PO. In the case of the original PSO algorithm, hyperparameters are fixed initially. However, this often leads to premature convergence and the problem of getting stuck in local optima. To balance the exploration-exploitation better, many variants of PSO have been introduced which adjust the hyperparameters of PSO based on some predetermined criteria. PG enables PSO to modify its hyperparameters automatically without fixing the adjustment criteria in advance. Jia et al. [89] utilise a policy network to interact with a PSO-based environment. The policy network input comprises a 2-dimensional state while the action space consists of six actions based on addition or subtraction of 0.05 from each of the 3 PSO hyperparameters. For every generation of PSO, the 2-dimensional

state consists of the sum of the differences of candidate solutions from their personal best solution and the global best solution respectively. A reward of +1 is given if the fitness improves in the next generation while a reward of -1 is provided if the fitness deteriorates. The fitness function for PSO is formulated as the Sharpe ratio of the entire portfolio. PG-PSO hybrid is shown to be superior to three PSO-based algorithms on six benchmarks. Also, the PG-PSO hybrid produces a better Sharpe ratio on a portfolio with $N = 20$ assets. Although the authors incorporate risk in their work, they ignore transaction costs ($c = 0$). Adapting hyperparameters of EC algorithms improves performance in dynamic environments but computation becomes expensive.

## 7 RESEARCH GAPS AND OPEN PROBLEMS

Researchers have used various EC+RL hybrids to model financial applications like trading and PO. However, there are some limitations to these approaches too. This section discusses research gaps in existing works and mentions some open problems identified through this survey.

### 7.1 Unrealistic assumptions

Many existing works make unrealistic assumptions that do not hold in the real world [152]. Some prominent ones are mentioned below:

- Transaction costs have often been ignored ($c = 0$) in existing works [44, 88, 89, 119]. This is a simplistic assumption as the absence of transaction costs encourages frequent trades.
- Some existing works [21, 22, 50, 88] focus on maximising returns without incorporating risk. In reality, most practitioners are risk-averse and try to maximise risk-adjusted returns [128].
- Many works [13, 44, 82, 111] assume that traders take long, neutral or short positions of fixed size ($\Delta_{n^t}$ is constant). However, this does not hold in the real world as quite often traders buy varying amounts of assets depending on their current wealth.
- Most trading and PO systems proposed in the literature use continuous time formulations. However, trades are placed in specified windows of around 6-7 hours on working days. The continuous time formulations ignore slippage in prices as the next day's opening price need not be the same as the current day's closing price.
- Most works assume liquidity of assets at all times. This is quite unrealistic as assets are not always available for placing trades [8].
- Another common assumption is that the trading actions do not affect prices. However, the prices are impacted by trading decisions, especially in small markets [93].
- Existing works often ignore the delays in order placements which are not very uncommon in financial markets. This also contributes to slippage in prices. Quite often, the prices at which the algorithm performs the calculations are different from the prices realised in real-world financial markets due to order placement delays.
- Most works concerning PO assume the problem to be unconstrained by ignoring many real-world constraints (discussed in detail in Section 5.2).
- Normal distributions are assumed for returns in many works that use analytical approaches. However, it has been shown that returns exhibit fat tails as big losses are quite likely [152].

Due to all these assumptions, most existing trading and PO systems do not port well to the real world. There is a need to develop more realistic trading and PO systems that do not make unrealistic assumptions for simplicity.

### 7.2 Aspects related to modelling of RL and EC components

In Appendix E, we outline the different ways existing works have modelled the RL and EC components in the solutions proposed for financial problems like trading and PO. However, there is still a large scope for enhancement as discussed below:

**State features.** Existing works using RL for financial applications have focused on close prices of assets as state features. However, the use of open, high, and low prices of financial assets is mostly unexplored. Another possibility is to use a combination of all 4 components of open-high-low-close (OHLC) data like arithmetic mean. Trading volumes and ticks data can also be incorporated as part of the state. Jamali et al. [88] incorporate forecasts obtained through multiple regression in the state representation. A significant amount of development has been seen in time series forecasting techniques [123, 144, 151]. Forecasts obtained from state-of-the-art techniques are yet to be tried out as state features for RL algorithms. Practitioners identify trading opportunities using econometric, mathematical, and statistical models for price movement. Incorporation of the output of such models as state features could result in better performance. The use of candlestick patterns is quite common in the analysis done by practitioners. However, the exploration of candlestick patterns in RL-based solutions is very limited. The usage of other sophisticated indicators as state features is also worth studying. Existing works [82] have focused on using boolean indicators. Efforts can be made to utilise more fine-grained indicators which contain relatively more information.

**Choice of actions.** Many trading systems proposed in literature [13, 44, 82, 111] focus on generating discrete trading signals {buy, hold, sell} at different time steps. Trading quantity is either considered to be of fixed size ($\Delta_{n^t}$ is constant) or left at the discretion of human traders using the system. Very few works model trading quantities as actions [201]. However, using trading quantities as actions increases computational complexity. There is a need to develop trading systems that handle this trade-off effectively. Most works dealing with PO model actions as a vector of portfolio allocation $\langle w_1^t, w_2^t, \ldots, w_N^t \rangle$ [203] instead of actual trading quantities of portfolio assets [195]. The focus is mainly on the portfolio rebalancing component and determining actual trades, placed after each rebalancing, is an overhead [22]. PO systems should also incorporate mechanisms to monitor portfolio assets by replacing poorly performing assets with potentially profitable assets. Lastly, negative weights can be allowed to model short selling (refer Section 5.2).

**Reward formulation.** The finance domain comprises a wide variety of utility functions [61] like constant absolute risk aversion (CARA), constant relative risk aversion (CRRA), etc. for modelling risk-aversion. The study of these utility functions as rewards is a potential research direction. Most risk-adjusted return metrics like the Sharpe ratio are defined as ratios. Penalty-based risk-adjusted return metrics [59] can also be tried out as reward functions. Recently, risk has been modelled in terms of the probability of achieving preset goals by investors [45]. Similar ideas can be incorporated to formulate novel reward functions.

**Fitness functions.** Extension ideas similar to reward functions can be applied to fitness functions as well. Utility functions like CARA, CRRA, etc. can be tried out alongside penalty-based risk-adjusted return metrics.

### 7.3 Aspects related to enhancement of solution approaches

This section discusses some ideas that can be used to enhance the solution approaches for financial problems like trading and PO.

**Adversarial concepts.** Quite often, adversarial traders generate false triggers in financial markets leading to suboptimal actions by victim traders [146]. Modelling of adversarial attacks and defences on intelligent trading and PO systems is another potential area of research that can greatly improve robustness in real-world financial markets.

**Multi-objective reinforcement learning (MORL).** Financial problems involve multiple objectives like maximising returns, minimising risk, minimising transaction costs, etc. Some of these objectives conflict with each other. Currently, the trade-off is captured by using metrics like the Sharpe ratio, Calmar ratio, etc. Some approaches use a weighted approach to reconcile multiple objectives into one single objective [143]. However, this requires extensive hyperparameter tuning to get the optimal weights for a given problem. MORL [72, 105] has the capability of dealing with multiple objectives. The use of MORL techniques for financial applications is another potential avenue for research.

**Planning approaches.** Most works surveyed in this paper have focused on using model-free RL approaches like Q-learning, SARSA, etc. Monte Carlo tree search (MCTS) and its variants have achieved significant success in playing complex games like chess and Go [161–163]. However, the use of such approaches in financial problems is less explored [143, 182]. The application of planning approaches to financial problems could prove to be another interesting area of research. However, planning approaches require a good enough model of the environment, which is difficult to obtain due to the non-stationary nature of financial markets.

**Modelling using POMDP.** Partially observable MDPs are used when the agent cannot observe the underlying state completely. This is often the case in the finance world as a common trader or portfolio manager might not be aware of all the underlying factors governing the evolution of prices of financial assets. Although it makes perfect sense to use POMDPs for modelling financial problems, their usage has been explored very little [77].

**Novel solution approaches.** RL is evolving rapidly and many novel algorithms have been proposed recently. Also, there's a plethora of EC algorithms available in the literature. Attempts can be made to develop novel EC+RL hybrids and analyse their performance for financial applications. There are various ways of hybridising RL and EC for financial applications such that one improves the performance of the other as discussed in Sections 3-6. EC algorithms can also be tried out in the framework of RL where the improvement in fitness function serves as the reward. In such a scenario, RL acts as an evolutionary search paradigm. Recently, researchers have shown interest in solving hard combinatorial optimisation problems using RL [124]. Similar ideas could be used to address the PO problem. Another possible research direction could be learning the choice of algorithms (in an algorithm portfolio scenario [65]), rather than their hyperparameters.

### 7.4 Aspects related to intricacies of finance domain

Apart from the unrealistic assumptions discussed in Section 7.1, various issues have not been addressed properly in the existing works. Some prominent ones are discussed below:

**Turbulent scenarios.** Unexpected scenarios like pandemics, wars, etc. lead to turbulence in financial markets resulting in huge losses for traders and PO managers [52, 87]. There is a requirement for approaches that avoid or limit losses in such scenarios.

**Short selling.** Most existing works have not incorporated the concept of short selling (refer Section 5.2) while designing trading and PO systems. Incorporating short selling adds to the realism of intelligent trading and PO systems.

**Integrating fundamental analysis for low-frequency trading.** Technical indicators have been extensively used in existing trading and PO systems but the usage of fundamental indicators is explored very little. Technical indicators are suited for high-frequency trading while fundamental indicators are used for long-term investment decisions. Zhang et al. [201] incorporate fundamental indicators as part of their state representation. Fundamental analysis through the use of natural language processing (NLP) on news articles and blogs to detect the public perception of financial assets is another potential research area.

**Tuning technical indicators.** Many existing works simply use technical indicators with default parameters or as suggested by financial experts [50]. Butler et al. [21] study ways of tuning Bollinger Bands. With the advances in computing capacity, it is possible to tune the parameters of indicators rather than relying on conventionally accepted values.

**Explainability.** One of the biggest reasons behind the non-acceptance of intelligent decision-making systems in finance is the lack of explainability. The development of more explainable and interpretable approaches that justify financial intuition can help in addressing this issue.

**Transaction costs.** The presence of transaction costs in financial markets discourages frequent trades and portfolio rebalancing. This increases the importance of hold/neutral action in RL-based trading systems. Many works [44, 88, 89, 119] have simply ignored the transaction costs for simplicity. Thresholds can also be introduced to reduce spurious movements from the neutral position arising due to insignificant fluctuations in financial markets.

**Extension to other related problems.** Most studies have focused mainly on stocks and currencies (Fig. 6). Efforts can be made to extend the proposed approaches to highly volatile assets like crypto-currencies [42], derivatives (futures and options) [79], etc. Although the focus of this study has been on trading and PO, there are other sequential decision-making problems in finance like market-making [18], hedging [107], pricing [99], etc. which roughly follow the same core structure. A study of the performance of EC+RL hybrids can be done on such problems. Additionally, sequential decision-making problems from the banking and insurance domains can be studied.

### 7.5 Other open issues

This section discusses some other important open issues:

**Lack of realistic financial simulators.** One of the biggest reasons for the recent success of RL [127, 161, 162] has been the availability of good simulators that model the dynamics of games, physics, etc. There is a lack of realistic simulators of financial environments. Most existing works use historical data for training and testing the proposed methods. Although such methods are fine for academic research and development, they usually do not port well to the real world due to the ever-changing nature of financial markets. Good financial simulators allow RL-based agents to learn by interaction instead of relying on historical data.

**Lack of established benchmarks and common statistical tests.** There is a lack of common and established benchmarks for comparing different solution approaches. Researchers have tested their proposed approaches on datasets that differ in terms of assets, time period, frequency, etc. Also, there is no common consensus on baselines used for comparison. Developing established benchmarks can help significantly in rigorously evaluating the proposed approaches. Another issue that needs to be addressed is the development of commonly accepted statistical significance tests with a special emphasis on financial intricacies.

**Scalability.** In financial environments, the state spaces get quite large due to the availability of a large number of technical indicators and their enormous number of variants. Existing PO systems work well when operating on small datasets like Dow Jones, NIFTY 50, etc. However, when it comes to large datasets like the S&P 500, the performance deteriorates. There is a need to come up with techniques that scale well to larger datasets. Also, the incorporation of an extra indicator essentially doubles the search space. So, exploring more state features is a computationally heavy job. However, linear speed-up is still possible by parallelisation which is a speciality of EC algorithms.

## 8 CONCLUSION

Although a lot of automation has been accomplished in finance, crucial decision-making is still done by humans. Humans suffer from various behavioural biases like overreaction [46], overconfidence [63], regret [14], loss aversion [92], etc. which affect their decisions. There is a need for intelligent systems that not only overcome behavioural biases but also save time by reducing the workload of practitioners. Trading and portfolio optimisation are two notable sequential decision-making tasks in the finance world. RL is the most natural approach for such tasks but it has its limitations. Researchers have shown that EC can be used to improve the performance of RL and vice versa. The hybrids of EC and RL have been successfully used for solving problems from various domains. In this article, we survey the different ways of hybridising EC and RL for trading and portfolio optimisation. We propose a novel taxonomy of such hybrid techniques that classifies the EC+RL hybrid approaches in finance into four major categories. The classification is done based on whether EC improves RL or RL improves EC and further if the hybrids are tightly coupled or loosely coupled. We discuss all four categories in great detail. Despite the significant advancements in the development of intelligent systems for trading and portfolio optimisation, there is still potential for further enhancement. We discover various research gaps in the existing literature and identify some open problems for future works. These include prevalent unrealistic assumptions, improvement in solution approaches and incorporating practical considerations from the finance domain. Addressing these research gaps and open problems would result in better portability to real-world financial markets and widespread acceptance of intelligent systems in the finance community.

## REFERENCES


[1] Siti Nazifah Zainol Abidin and Maheran Mohd Jaffar. 2012. A review on Geometric Brownian Motion in forecasting the share prices in Bursa Malaysia. *World Applied Sciences Journal* 17, 1 (2012), 82–93.

[2] Amine Mohamed Aboussalah and Chi-Guhn Lee. 2020. Continuous control with stacked deep dynamic recurrent reinforcement learning for portfolio optimization. *Expert Systems with Applications* 140 (2020), 112891.

[3] Steven B Achelis. 2001. *Technical Analysis from A to Z.* McGraw Hill Education.

[4] David Ackley and Michael Littman. 1991. Interactions between learning and evolution. *Artificial life II* 10 (1991), 487–509.

[5] Wilhelm Ala-Krekola et al. 2021. Financial portfolio management with evolution strategies-based reinforcement learning. (2021).

[6] Saud Almahdi and Steve Y Yang. 2017. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications* 87 (2017), 267–279.

[7] Saud Almahdi and Steve Y Yang. 2019. A constrained portfolio trading system using particle swarm algorithm and recurrent reinforcement learning. *Expert Systems with Applications* 130 (2019), 145–156.

[8] Yakov Amihud, Haim Mendelson, Lasse Heje Pedersen, et al. 2006. Liquidity and asset prices. *Foundations and Trends® in Finance* 1, 4 (2006), 269–364.

[9] Mark P Austin, Graham Bates, Michael AH Dempster 3, Vasco Leemans, and Stacy N Williams. 2004. Adaptive systems for foreign exchange trading. *Quantitative Finance* 4, 4 (2004), 37–45.

[10] Hosein Hamisheh Bahar, Mohammad H Fazel Zarandi, and Akbar Esfahanipour. 2016. Generating ternary stock trading signals using fuzzy genetic network programming. In *2016 Annual Conference of the North American Fuzzy Information Processing Society (NAFIPS)*. IEEE, 1–6.

[11] Hui Bai, Ran Cheng, and Yaochu Jin. 2023. Evolutionary Reinforcement Learning: A Survey. *Intelligent Computing* 2 (2023), 0025.

[12] Danielle Smith Bassett and ED Bullmore. 2006. Small-world brain networks. *The neuroscientist* 12, 6 (2006), 512–523.

[13] R Graham Bates, Michael AH Dempster, and Yazann S Romahi. 2003. Evolutionary reinforcement learning in FX order book and order flow analysis. In *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings.* IEEE, 355–362.

[14] David E Bell. 1982. Regret in decision making under uncertainty. *Operations research* 30, 5 (1982), 961–981.

[15] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5, 2 (1994), 157–166.

[16] Hans-Georg Beyer. 2001. *The theory of evolution strategies.* Springer Science & Business Media.

[17] Hans-Georg Beyer and Hans-Paul Schwefel. 2002. Evolution strategies–a comprehensive introduction. *Natural computing* 1 (2002), 3–52.

[18] Taweh Beysolow II and Taweh Beysolow II. 2019. Market making via reinforcement learning. *Applied Reinforcement Learning with Python: With OpenAI Gym, Tensorflow, and Keras* (2019), 77–94.

[19] Vandana Bharti, Bhaskar Biswas, and Kaushal Kumar Shukla. 2022. QL-SSA: An Adaptive Q-Learning based Squirrel Search Algorithm for Feature Selection. In *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–7.

[20] John Bollinger. 1992. Using bollinger bands. *Stocks & Commodities* 10, 2 (1992), 47–51.

[21] Matthew Butler and Dimitar Kazakov. 2012. A learning adaptive Bollinger band system. In *2012 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*. IEEE, 1–8.

[22] Isidoro J Casanova. 2010. Tradinnova-LCS: Dynamic stock portfolio decision-making assistance model with genetic based machine learning. In *IEEE Congress on Evolutionary Computation*. IEEE, 1–8.

[23] Tun-Jen Chang, Sang-Chin Yang, and Kuang-Jung Chang. 2009. Portfolio optimization problems in different risk measures using genetic algorithm. *Expert Systems with applications* 36, 7 (2009), 10529–10537.

[24] Ying-Hua Chang and Ming-Sheng Lee. 2017. Incorporating Markov decision process on genetic algorithms to formulate trading strategies for stock markets. *Applied Soft Computing* 52 (2017), 1143–1153.

[25] Gong-meng Chen, Michael Firth, and Oliver M Rui. 2001. The dynamic relation between stock returns, trading volume, and volatility. *Financial Review* 36, 3 (2001), 153–174.

[26] Yan Chen, Shingo Mabu, and Kotaro Hirasawa. 2010. A model of portfolio optimization using time adapting genetic network programming. *Computers & operations research* 37, 10 (2010), 1697–1707.

[27] Yan Chen, Shingo Mabu, Kotaro Hirasawa, and Jinglu Hu. 2007. Enhancement of trading rules on stock markets using genetic network programming with Sarsa learning. In *SICE Annual Conference 2007*. IEEE, 2700–2707.

[28] Yan Chen, Shingo Mabu, Kotaro Hirasawa, and Jinglu Hu. 2007. Genetic network programming with sarsa learning and its application to creating stock trading rules. In *2007 IEEE Congress on Evolutionary Computation*. IEEE, 220–227.

[29] Yan Chen, Shingo Mabu, Kotaro Hirasawa, and Jinglu Hu. 2007. Trading rules on stock markets using genetic network programming with sarsa learning. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. 1503–1503.

[30] Yan Chen, Shingo Mabu, Etsushi Ohkawa, and Kotaro Hirasawa. 2009. Constructing portfolio investment strategy based on time adapting genetic network programming. In *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2379–2386.

[31] Yan Chen, Shingo Mabu, Kaoru Shimada, and Kotaro Hirasawa. 2008. Construction of portfolio optimization system using genetic network programming with control nodes. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. 1693–1694.

[32] Yan Chen, Shingo Mabu, Kaoru Shimada, and Kotaro Hirasawa. 2009. A genetic network programming with learning approach for enhanced stock trading model. *Expert Systems with Applications* 36, 10 (2009), 12537–12546.

[33] Yan Chen, Shingo Mabu, Kaoru Shimada, and Kotaro Hirasawa. 2009. Real time updating genetic network programming for adapting to the change of stock prices. *IEEJ Transactions on Electronics, Information and Systems* 129, 2 (2009), 344–354.

[34] Yan Chen, Etsushi Ohkawa, Shingo Mabu, Kaoru Shimada, and Kotaro Hirasawa. 2008. Stock trading model for multi-brands optimization based on Genetic Network Programming with control nodes. In *2008 SICE Annual Conference*. IEEE, 664–669.

[35] Yan Chen, Etsushi Ohkawa, Shingo Mabu, Kaoru Shimada, and Kotaro Hirasawa. 2009. A portfolio optimization model using Genetic Network Programming with control nodes. *Expert Systems with Applications* 36, 7 (2009), 10735–10745.

[36] Yan Chen and Xuancheng Wang. 2015. A hybrid stock trading system using genetic network programming and mean conditional value-at-risk. *European Journal of Operational Research* 240, 3 (2015), 861–871.

[37] Alexandre Clément, Élisabeth Robinot, and Léo Trespeuch. 2023. The use of ESG scores in academic literature: A systematic literature review. *Journal of Enterprising Communities: People and Places in the Global Economy* (2023).

[38] Alfredo V Clemente, Humberto N Castejón, and Arjun Chandra. 2017. Efficient parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1705.04862* (2017).

[39] Naftali Cohen, Tucker Balch, and Manuela Veloso. 2020. Trading via image classification. In *Proceedings of the First ACM International Conference on AI in Finance*. 1–6.

[40] Jennifer Conrad and Gautam Kaul. 1998. An anatomy of trading strategies. *The Review of Financial Studies* 11, 3 (1998), 489–519.

[41] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. 2018. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *Advances in neural information processing systems* 31 (2018).

[42] Shaen Corbet, Brian Lucey, Andrew Urquhart, and Larisa Yarovaya. 2019. Cryptocurrencies as a financial asset: A systematic analysis. *International Review of Financial Analysis* 62 (2019), 182–199.

[43] Tunchan Cura. 2009. Particle swarm optimization approach to portfolio optimization. *Nonlinear analysis: Real world applications* 10, 4 (2009), 2396–2406.

[44] Guinther K da Costa, Leandro dos S Coelho, and Roberto Z Freire. 2020. Image Representation of Time Series for Reinforcement Learning Trading Agent. In *Congresso Brasileiro de Automática-CBA*, Vol. 2.

[45] Sanjiv Ranjan Das, Daniel N Ostrov, Anand Radhakrishnan, and Deep Srivastav. 2018. A new approach to goals-based wealth management. *Available at SSRN 3117765* (2018).

[46] Werner FM De Bondt and Richard Thaler. 1985. Does the stock market overreact? *The Journal of finance* 40, 3 (1985), 793–805.

[47] Michael AH Dempster and Vasco Leemans. 2006. An automated FX trading system using adaptive reinforcement learning. *Expert systems with applications* 30, 3 (2006), 543–552.

[48] Michael AH Dempster, Tom W Payne, Yazann Romahi, and Giles WP Thompson. 2001. Computational learning techniques for intraday FX trading using popular technical indicators. *IEEE Transactions on neural networks* 12, 4 (2001), 744–754.

[49] Michael Alan Howarth Dempster and Chris M Jones. 2001. A real-time adaptive trading system using genetic programming. *Quantitative Finance* 1, 4 (2001), 397.

[50] Michael Alan Howarth Dempster and Yazann S Romahi. 2002. Intraday FX trading: An evolutionary reinforcement learning approach. In *Intelligent Data Engineering and Automated Learning—IDEAL 2002: Third International Conference Manchester, UK, August 12–14, 2002 Proceedings 3*. Springer, 347–358.

[51] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. 2016. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems* 28, 3 (2016), 653–664.

[52] Sinem Derindere Köseoğlu, Burcu Adıgüzel Mercangöz, Khalid Khan, and Suleman Sarwar. 2023. The impact of the Russian-Ukraine war on the stock market: a causal analysis. *Applied Economics* (2023), 1–11.

[53] Madalina M Drugan. 2019. Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. *Swarm and evolutionary computation* 44 (2019), 228–246.

[54] Agoston E Eiben and James E Smith. 2015. *Introduction to evolutionary computing*. Springer.

[55] Dennis Eilers, Christian L Dunis, Hans-Jörg von Mettenheim, and Michael H Breitner. 2014. Intelligent trading of seasonal effects: A decision support algorithm based on reinforcement learning. *Decision support systems* 64 (2014), 100–108.

[56] Andries P Engelbrecht. 2010. Heterogeneous particle swarm optimization. In *Swarm Intelligence: 7th International Conference, ANTS 2010, Brussels, Belgium, September 8-10, 2010. Proceedings 7*. Springer, 191–202.

[57] Thomas G Fischer. 2018. *Reinforcement learning in financial markets-a survey*. Technical Report. FAU Discussion Papers in Economics.

[58] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. 2018. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning* 11, 3-4 (2018), 219–354.

[59] Sumitra Ganesh, Nelson Vadori, Mengda Xu, Hua Zheng, Prashant Reddy, and Manuela Veloso. 2019. Reinforcement learning for market making in a multi-agent dealer market. *arXiv preprint arXiv:1911.05892* (2019).

[60] Erol Gelenbe. 1989. Random neural networks with negative and positive signals and product form solution. *Neural computation* 1, 4 (1989), 502–510.

[61] Hans U Gerber and Gérard Pafum. 1998. Utility functions: from risk theory to finance. *North American Actuarial Journal* 2, 3 (1998), 74–91.

[62] Eduardo A Gerlein, Martin McGinnity, Ammar Belatreche, and Sonya Coleman. 2016. Evaluating machine learning classification for financial trading: An empirical approach. *Expert Systems with Applications* 54 (2016), 193–207.

[63] Simon Gervais and Terrance Odean. 2001. Learning to be overconfident. *The review of financial studies* 14, 1 (2001), 1–27.

[64] Carl Gold. 2003. FX trading via recurrent reinforcement learning. In *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings*. IEEE, 363–370.

[65] Carla P Gomes and Bart Selman. 2001. Algorithm portfolios. *Artificial Intelligence* 126, 1-2 (2001), 43–62.

[66] Denise Gorse. 2011. Application of stochastic recurrent reinforcement learning to index trading. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.

[67] Benjamin Graham. 2017. *The intelligent investor*. HarperCollins Publishers Inc.

[68] Yunqing Gu, Shingo Mabu, Yang Yang, Jianhua Li, and Kotaro Hirasawa. 2011. Trading rules on stock markets using Genetic Network Programming-Sarsa learning with plural subroutines. In *SICE Annual Conference 2011*. IEEE, 143–148.

[69] Myoung Hoon Ha, Seung-geun Chi, Sangyeop Lee, Yujin Cha, and Moon Byung-Ro. 2021. Evolutionary meta reinforcement learning for portfolio optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 964–972.

[70] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[71] Ben Hambly, Renyuan Xu, and Huining Yang. 2023. Recent advances in reinforcement learning in finance. *Mathematical Finance* 33, 3 (2023), 437–503.

[72] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. 2022. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems* 36, 1 (2022), 26.

[73] Verena Heidrich-Meisner and Christian Igel. 2008. Evolution strategies for direct policy search. In *International Conference on Parallel Problem Solving from Nature*. Springer, 428–437.

[74] Daniel Hein, Steffen Udluft, and Thomas A Runkler. 2019. Generating interpretable reinforcement learning policies using genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 23–24.

[75] Geoffrey E Hinton, Steven J Nowlan, et al. 1987. How learning can guide evolution. *Complex systems* 1, 3 (1987), 495–502.

[76] Kotaro Hirasawa, Masafumi Okubo, Hiropon Katagiri, J Hu, and Junichi Murata. 2001. Comparison between genetic network programming (GNP) and genetic programming (GP). In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, Vol. 2. IEEE, 1276–1282.

[77] Badr Hirchoua, Imadeddine Mountasser, Brahim Ouhbi, and Bouchra Frikh. 2021. Evolutionary Deep Reinforcement Learning Environment: Transfer Learning-Based Genetic Algorithm. In *The 23rd International Conference on Information Integration and Web Intelligence*. 242–249.

[78] Badr Hirchoua, Imadeddine Mountasser, Brahim Ouhbi, and Bouchra Frikh. 2022. Continuous Evolutionary Deep Reinforcement Learning Environment: Augmented Transfer Learning-Based Genetic Algorithm. *Journal of Data Intelligence* 3, 3 (2022), 333–349.

[79] Ali Hirsa and Salih N Neftci. 2013. *An introduction to the mathematics of financial derivatives*. Academic press.

[80] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[81] John H Holland. 1992. Genetic algorithms. *Scientific american* 267, 1 (1992), 66–73.

[82] Andrei Hryshko and Tom Downs. 2004. System for foreign exchange trading using genetic algorithms and reinforcement learning. *International journal of systems science* 35, 13-14 (2004), 763–774.

[83] Andrei Hryshko and Tom Downs. 2005. A machine learning approach to intraday trading on foreign exchange markets. In *Intelligent Data Engineering and Automated Learning-IDEAL 2005: 6th International Conference, Brisbane, Australia, July 6-8, 2005. Proceedings 6*. Springer, 588–595.

[84] Andrei Hryshko and Tom Downs. 2006. Development of machine learning software for high frequency trading in financial markets. In *Business Applications and Computational Intelligence*. IGI Global, 406–430.

[85] Yong Hu, Kang Liu, Xiangzhou Zhang, Lijun Su, EWT Ngai, and Mei Liu. 2015. Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing* 36 (2015), 534–551.

[86] Hitoshi Iba and Claus C Aranha. 2012. *Practical applications of evolutionary computation to financial engineering*. Springer.

[87] Shaista Jabeen, Muhammad Farhan, Muhammad Ahmad Zaka, Muhammad Fiaz, and Mobina Farasat. 2022. COVID and world stock markets: A comprehensive discussion. *Frontiers in Psychology* 12 (2022), 763346.

[88] Hana Jamali, Younes Chihab, Iván García-Magariño, and Omar Bencharef. 2023. Hybrid Forex prediction model using multiple regression, simulated annealing, reinforcement learning and technical analysis. *Int J Artif Intell ISSN* 2252, 8938 (2023), 8938.

[89] Xuanyu Jia and Xin Cai. 2022. A Policy Gradient Based Particle Swarm Optimizer for Portfolio Optimization Problem. In *2022 41st Chinese Control Conference (CCC)*. IEEE, 1991–1996.

[90] Voratas Kachitvichyanukul. 2012. Comparison of three evolutionary algorithms: GA, PSO, and DE. *Industrial Engineering and Management Systems* 11, 3 (2012), 215–223.

[91] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence* 101, 1-2 (1998), 99–134.

[92] Daniel Kahneman and Amos Tversky. 2013. Prospect theory: An analysis of decision under risk. In *Handbook of the fundamentals of financial decision making: Part I*. World Scientific, 99–127.

[93] Jonathan M Karpoff. 1987. The relation between price changes and trading volume: A survey. *Journal of Financial and quantitative Analysis* 22, 1 (1987), 109–126.

[94] James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, Vol. 4. IEEE, 1942–1948.

[95] Harshad Khadilkar. 2023. Supplementing Gradient-Based Reinforcement Learning with Simple Evolutionary Ideas. *arXiv preprint arXiv:2305.07571* (2023).

[96] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *Science* 220, 4598 (1983), 671–680.

[97] Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems* 12 (1999).

[98] John R Koza. 1994. Genetic programming as a means for programming computers by natural selection. *Statistics and computing* 4 (1994), 87–112.

[99] Erich Kutschinski, Thomas Uthmann, and Daniel Polani. 2003. Learning competitive pricing strategies by multi-agent reinforcement learning. *Journal of Economic Dynamics and Control* 27, 11-12 (2003), 2207–2218.

[100] Baruch Lev and S Ramu Thiagarajan. 1993. Fundamental information analysis. *Journal of Accounting research* 31, 2 (1993), 190–215.

[101] JianHua Li, QinBiao Meng, Yang Yang, Shingo Mabu, Yifei Wang, and Kotaro Hirasawa. 2010. Trading rules on stock markets using genetic network programming with subroutines. In *Proceedings of SICE Annual Conference 2010*. IEEE, 3084–3088.

[102] Xianneng Li, Wen He, and Kotaro Hirasawa. 2014. Creating stock trading rules using graph-based estimation of distribution algorithm. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 731–738.

[103] Konstantinos Liagkouras and K Metaxiotis. 2018. A new efficiently encoded multiobjective algorithm for the solution of the cardinality constrained portfolio optimization problem. *Annals of Operations Research* 267 (2018), 281–319.

[104] Greeshma Lingam, Rashmi Ranjan Rout, Durvasula VLN Somayajulu, and Soumya K Ghosh. 2020. Particle swarm optimization on deep reinforcement learning for detecting social spam bots and spam-influential users in twitter network. *IEEE Systems Journal* 15, 2 (2020), 2281–2292.

[105] Chunming Liu, Xin Xu, and Dewen Hu. 2014. Multiobjective reinforcement learning: A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 3 (2014), 385–398.

[106] Haoqiang Liu, Zefang Zong, Yong Li, and Depeng Jin. 2023. NeuroCrossover: An intelligent genetic locus selection scheme for genetic algorithm using reinforcement learning. *Applied Soft Computing* 146 (2023), 110680.

[107] Peng Liu. 2023. A review on derivative hedging using reinforcement learning. *Journal of Financial Data Science* (2023), 1.

[108] Xiao-Yang Liu, Hongyang Yang, Jiechao Gao, and Christina Dan Wang. 2021. FinRL: Deep reinforcement learning framework to automate trading in quantitative finance. In *Proceedings of the second ACM international conference on AI in finance*. 1–9.

[109] Andrew W Lo. 2004. The adaptive markets hypothesis: Market efficiency from an evolutionary perspective. *Journal of Portfolio Management, Forthcoming* (2004).

[110] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* 30 (2017).

[111] David W Lu. 2017. Agent inspired trading using recurrent reinforcement learning and lstm neural networks. *arXiv preprint arXiv:1707.07338* (2017).

[112] Shingo Mabu, Yan Chen, Kotaro Hirasawa, and Jinglu Hu. 2007. Genetic network programming with actor-critic and its application to stock trading model. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. 2263–2263.

[113] Shingo Mabu, Yan Chen, Kotaro Hirasawa, and Jinglu Hu. 2007. Stock trading rules using genetic network programming with actor-critic. In *2007 IEEE Congress on Evolutionary Computation*. IEEE, 508–515.

[114] Shingo Mabu and Kotaro Hirasawa. 2011. Enhanced rule extraction and classification mechanism of genetic network programming for stock trading signal generation. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 1659–1666.

[115] Shingo Mabu, Kotaro Hirasawa, and Jinglu Hu. 2007. A graph-based evolutionary algorithm: Genetic network programming (GNP) and its extension using reinforcement learning. *Evolutionary computation* 15, 3 (2007), 369–398.

[116] Shingo Mabu, Kotaro Hirasawa, Masanao Obayashi, and Takashi Kuremoto. 2013. Enhanced decision making mechanism of rule-based genetic network programming for creating stock trading signals. *Expert Systems with Applications* 40, 16 (2013), 6311–6320.

[117] Shingo Mabu, Yuzhu Lian, Yan Chen, and Kotaro Hirasawa. 2010. Generating stock trading signals based on matching degree with extracted rules by genetic network programming. In *Proceedings of SICE Annual Conference 2010*. IEEE, 1164–1169.

[118] Burton G Malkiel. 1989. Efficient market hypothesis. In *Finance*. Springer, 127–134.

[119] Chari Maree and Christian W Omlin. 2022. Balancing Profit, Risk, and Sustainability for Portfolio Management. In *2022 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr)*. IEEE, 1–8.

[120] Dietmar Maringer and Tikesh Ramtohul. 2012. Regime-switching recurrent reinforcement learning for investment decision making. *Computational Management Science* 9 (2012), 89–107.

[121] Dietmar Maringer and Jin Zhang. 2014. Transition variable selection for regime switching recurrent reinforcement learning. In *2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*. IEEE, 407–413.

[122] Harry Markowitz. 1952. Portfolio Selection. *The Journal of Finance* 7, 1 (1952), 77–91.

[123] Ricardo P Masini, Marcelo C Medeiros, and Eduardo F Mendes. 2023. Machine learning advances for time series forecasting. *Journal of economic surveys* 37, 1 (2023), 76–111.

[124] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. 2021. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research* 134 (2021), 105400.

[125] Terry Lingze Meng and Matloob Khushi. 2019. Reinforcement learning in financial markets. *Data* 4, 3 (2019), 110.

[126] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.

[127] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[128] Franco Modigliani and Leah Modigliani. 1997. Risk-adjusted performance. *Journal of portfolio management* 23, 2 (1997), 45–54.

[129] John Moody and Matthew Saffell. 1998. Reinforcement learning for trading. *Advances in Neural Information Processing Systems* 11 (1998).

[130] John Moody and Matthew Saffell. 2001. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks* 12, 4 (2001), 875–889.

[131] John Moody, Matthew Saffell, Yuansong Liao, and Lizhong Wu. 1998. Reinforcement learning for trading systems and portfolios: Immediate vs future rewards. In *Decision Technologies for Computational Finance: Proceedings of the fifth International Conference Computational Finance*. Springer, 129–140.

[132] John Moody, Lizhong Wu, Yuansong Liao, and Matthew Saffell. 1998. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting* 17, 5-6 (1998), 441–470.

[133] Amirhosein Mosavi, Yaser Faghan, Pedram Ghamisi, Puhong Duan, Sina Faizollahzadeh Ardabili, Ely Salwana, and Shahab S Band. 2020. Comprehensive review of deep reinforcement learning methods and applications in economics. *Mathematics* 8, 10 (2020), 1640.

[134] John J Murphy. 1999. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin.

[135] Christopher Neely, Paul Weller, and Rob Dittmar. 1997. Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *Journal of financial and Quantitative Analysis* 32, 4 (1997), 405–426.

[136] John A Nelder and Roger Mead. 1965. A simplex method for function minimization. *The computer journal* 7, 4 (1965), 308–313.

[137] Hieu Trung Nguyen, Khang Tran, and Ngoc Hoang Luong. 2022. Combining Soft-Actor Critic with Cross-Entropy Method for Policy Search in Continuous Control. In *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[138] Bizuayehu Nigatu. 2020. *A Multi-Objective, Multiperiod Portfolio Optimization Using Particle Swarm Optimization and Asynchronous Advantage Actor-Critic: Investors' Behavioral Characterization*. Ph. D. Dissertation. Colorado Technical University.

[139] Steve Nison. 2001. *Japanese candlestick charting techniques: a contemporary guide to the ancient investment techniques of the Far East*. Penguin.

[140] Michael O'Neill. 2009. Riccardo Poli, William B. Langdon, Nicholas F. McPhee: A Field Guide to Genetic Programming: Lulu. com, 2008, 250 pp, ISBN 978-1-4092-0073-4.

[141] Hyungjun Park, Min Kyu Sim, and Dong Gu Choi. 2020. An intelligent financial portfolio trading strategy using deep Q-learning. *Expert Systems with Applications* 158 (2020), 113573.

[142] Merav Parter, Nadav Kashtan, and Uri Alon. 2008. Facilitated variation: how evolution learns from past environments to generalize to new environments. *PLoS computational biology* 4, 11 (2008), e1000206.

[143] Sunandita Patra, Mahmoud Mahfouz, Sriram Gopalakrishnan, Daniele Magazzeni, and Manuela Veloso. 2023. FinRDDL: Can AI Planning be used for Quantitative Finance Problems? *FinPlan 2023* (2023), 36.

[144] Fotios Petropoulos, Daniele Apiletti, Vassilios Assimakopoulos, Mohamed Zied Babai, Devon K Barrow, Souhaib Ben Taieb, Christoph Bergmeir, Ricardo J Bessa, Jakub Bijak, John E Boylan, et al. 2022. Forecasting: theory and practice. *International Journal of Forecasting* 38, 3 (2022), 705–871.

[145] Antonin Ponsich, Antonio Lopez Jaimes, and Carlos A Coello Coello. 2012. A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications. *IEEE Transactions on evolutionary computation* 17, 3 (2012), 321–344.

[146] Tālis J Putniņš. 2012. Market manipulation: A survey. *Journal of economic surveys* 26, 5 (2012), 952–967.

[147] Hong Qian and Yang Yu. 2021. Derivative-free reinforcement learning: a review. *Frontiers of Computer Science* 15, 6 (2021), 156336.

[148] Majdi I Radaideh and Koroush Shirvan. 2021. Rule-based reinforcement learning methodology to inform evolutionary algorithms for constrained optimization of engineering applications. *Knowledge-Based Systems* 217 (2021), 106836.

[149] Kanchan Rajwar, Kusum Deep, and Swagatam Das. 2023. An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges. *Artificial Intelligence Review* 56, 11 (2023), 13187–13257.

[150] Pratyusha Rakshit, Amit Konar, and Atulya K Nagar. 2020. Q-learning induced artificial bee colony for noisy optimization. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[151] Aravilli Atchuta Ram, Sandarbh Yadav, Yelleti Vivek, and Vadlamani Ravi. 2024. Deep Reinforcement Learning for Financial Forecasting in Static and Streaming Cases. *Journal of Information & Knowledge Management* (2024), 2450080.

[152] Ashwin Rao and Tikhon Jelvis. 2022. *Foundations of Reinforcement Learning with Applications in Finance*. CRC Press.

[153] Adam V Reed. 2013. Short selling. *Annu. Rev. Financ. Econ.* 5, 1 (2013), 245–258.

[154] Gavin A Rummery and Mahesan Niranjan. 1994. *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK.

[155] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864* (2017).

[156] Jürgen Schmidhuber, Daan Wierstra, Matteo Gagliolo, and Faustino Gomez. 2007. Training recurrent networks by evolino. *Neural computation* 19, 3 (2007), 757–779.

[157] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.

[158] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[159] Will Serrano. 2018. The random neural network with a genetic algorithm and deep learning clusters in fintech: smart investment. In *Artificial Intelligence Applications and Innovations: 14th IFIP WG 12.5 International Conference, AIAI 2018, Rhodes, Greece, May 25–27, 2018, Proceedings 14*. Springer, 297–310.

[160] William F Sharpe. 1966. Mutual fund performance. *The Journal of business* 39, 1 (1966), 119–138.

[161] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.

[162] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815* (2017).

[163] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144.

[164] Prisadarng Skolpadungket, Keshav Dahal, and Napat Harnpornchai. 2007. Portfolio optimization using multi-obj ective genetic algorithms. In *2007 IEEE Congress on Evolutionary Computation*. IEEE, 516–523.

[165] Yupu Song. 2017. *A Forex Trading System Using Evolutionary Reinforcement Learning*. Ph. D. Dissertation. Worcester Polytechnic Institute.

[166] Yanjie Song, Luona Wei, Qing Yang, Jian Wu, Lining Xing, and Yingwu Chen. 2023. RL-GA: A reinforcement learning-based genetic algorithm for electromagnetic detection satellite scheduling problem. *Swarm and Evolutionary Computation* (2023), 101236.

[167] Yanjie Song, Yutong Wu, Yangyang Guo, Ran Yan, Ponnuthurai Nagaratnam Suganthan, Yue Zhang, Witold Pedrycz, Yingwu Chen, Swagatam Das, Rammohan Mallipeddi, et al. 2023. Reinforcement Learning-assisted Evolutionary Algorithm: A Survey and Research Opportunities. *arXiv preprint arXiv:2308.13420* (2023).

[168] Srijan Sood, Kassiani Papasotiriou, Marius Vaiciulis, and Tucker Balch. 2023. Deep Reinforcement Learning for Optimal Portfolio Allocation: A Comparative Study with Mean-Variance Optimization. *FinPlan* 2023, 2023 (2023), 21.

[169] Olaf Sporns, Dante R Chialvo, Marcus Kaiser, and Claus C Hilgetag. 2004. Organization, development and function of complex brain networks. *Trends in cognitive sciences* 8, 9 (2004), 418–425.

[170] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[171] Rainer Storn and Kenneth Price. 1997. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341.

[172] Ivana Strumberger, Eva Tuba, Nebojsa Bacanin, Marko Beko, and Milan Tuba. 2018. Hybridized artificial bee colony algorithm for constrained portfolio optimization problem. In *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[173] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. 2017. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567* (2017).

[174] Shuo Sun, Rundong Wang, and Bo An. 2023. Reinforcement learning for quantitative trading. *ACM Transactions on Intelligent Systems and Technology* 14, 3 (2023), 1–29.

[175] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[176] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).

[177] Matthew E Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10, 7 (2009).

[178] Michele Tessari and Giovanni Iacca. 2022. Reinforcement learning based adaptive metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1854–1861.

[179] Ryan J Urbanowicz and Jason H Moore. 2009. Learning classifier systems: a complete introduction, review, and roadmap. *Journal of Artificial Evolution and Applications* 2009 (2009).

[180] Mendhikar Vishal, Vadlamani Ravi, and Ramanuj Lal. 2024. Ensemble Deep Reinforcement Learning for Financial Trading. In *Machine Learning Approaches in Financial Analytics*. Springer, 191–207.

[181] Edoardo Vittori. 2022. *Augmenting traders with learning machines*. Ph. D. Dissertation. Politecnico di Milano.

[182] Edoardo Vittori, Amarildo Likmeta, and Marcello Restelli. 2021. Monte carlo tree search for trading and hedging. In *Proceedings of the Second ACM International Conference on AI in Finance*. 1–9.

[183] Yelleti Vivek, P Shanmukh Kali Prasad, Vadlamani Madhav, Ramanuj Lal, and Vadlamani Ravi. 2024. Optimal Technical Indicator Based Trading Strategies Using Evolutionary Multi Objective Optimization Algorithms. *Computational Economics* (2024), 1–51.

[184] Liad Wagman. 2003. Stock portfolio evaluation: An application of genetic-programming-based technical analysis. *Genetic Algorithms and Genetic Programming at Stanford* 2003 (2003), 213–220.

[185] Shuyang Wang and Diego Klabjan. 2023. An Ensemble Method of Deep Reinforcement Learning for Automated Cryptocurrency Trading. (2023).

[186] Christopher John Cornish Hellaby Watkins. 1989. *Learning from delayed rewards*. Ph. D. Dissertation. King's College, Cambridge United Kingdom.

[187] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big data* 3 (2016), 1–40.

[188] Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560.

[189] Spencer K White, Tony Martinez, and George Rudolph. 2010. Generating a novel sort algorithm using Reinforcement Programming. In *IEEE Congress on Evolutionary Computation*. IEEE, 1–8.

[190] Shimon Whiteson. 2006. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research* 7 (2006).

[191] Shimon Whiteson. 2012. Evolutionary computation for reinforcement learning. *Reinforcement Learning: State-of-the-art* (2012), 325–355.

[192] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8 (1992), 229–256.

[193] Zeyu Xia, Mingde Shi, and Changle Lin. 2023. Stock Trading Strategy Developing Based on Reinforcement Learning. In *Proceedings of the 2nd International Academic Conference on Blockchain, Information Technology and Smart Finance (ICBIS 2023)*. Atlantis Press, 156–164.

[194] Wei Xu, Lutao Wang, Shingo Mabu, and Kotaro Hirasawa. 2012. Genetic network programming with credit. In *2012 Proceedings of SICE Annual Conference (SICE)*. IEEE, 1769–1777.

[195] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. 2020. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the first ACM international conference on AI in finance*. 1–8.

[196] Yang Yang, Yunqing Gu, Shingo Mabu, and Kotaro Hirasawa. 2012. Genetic Network Programming-Sarsa with Multi-Subroutines for Trading Rules on Stock Markets. *IEEJ Transactions on Electronics, Information and Systems* 132, 3 (2012), 439–447.

[197] Yang Yang, Zhaoping He, Shingo Mabu, and Kotaro Hirasawa. 2012. A cooperative coevolutionary stock trading model using genetic network programming-sarsa. *Journal of Advanced Computational Intelligence and Intelligent Informatics* 16, 5 (2012), 581–590.

[198] Yang Yang, JianHua Li, Shingo Mabu, and Kotaro Hirasawa. 2010. GNP-Sarsa with subroutines for trading rules on stock markets. In *2010 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 1161–1165.

[199] Jin Zhang and Dietmar Maringer. 2013. Indicator selection for daily equity trading with recurrent reinforcement learning. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*. 1757–1758.

[200] Jin Zhang and Dietmar Maringer. 2014. Two parameter update schemes for recurrent reinforcement learning. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1449–1453.

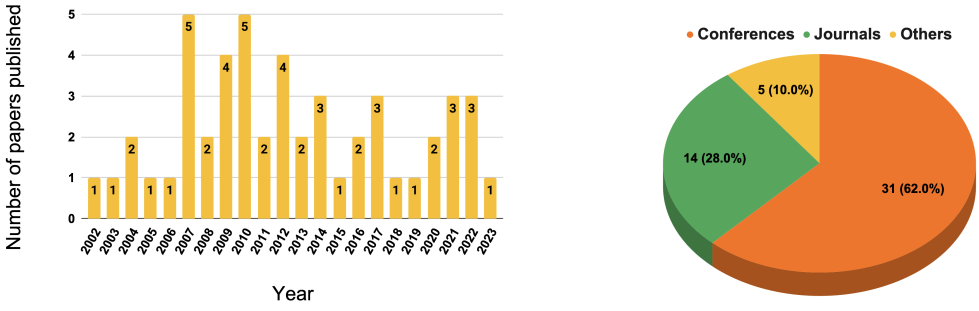
[201] Jin Zhang and Dietmar Maringer. 2016. Using a genetic algorithm to improve recurrent reinforcement learning for equity trading. *Computational Economics* 47 (2016), 551–567.

[202] Zihao Zhang, Stefan Zohren, and Stephen Roberts. 2019. Deep reinforcement learning for trading. *arXiv preprint arXiv:1911.10107* (2019).
[203] Zihao Zhang, Stefan Zohren, and Stephen Roberts. 2020. Deep learning for portfolio optimization. *arXiv preprint arXiv:2005.13665* (2020).
[204] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and evolutionary computation* 1, 1 (2011), 32–49.
[205] Qingling Zhu, Xiaoqiang Wu, Qiuzhen Lin, Lijia Ma, Jianqiang Li, Zhong Ming, and Jianyong Chen. 2023. A survey on evolutionary reinforcement learning algorithms. *Neurocomputing* 556 (2023), 126628.
[206] Eric Zivot. 2017. *Introduction to computational finance and financial econometrics.* Chapman & Hall Crc.

# APPENDICES

## A PAPER COLLECTION METHODOLOGY

For this work, more than 50 papers published between 2000 and 2023 have been surveyed (Fig. 9a). These papers have been collected through keyword searches on web search engines like Google Scholar and important databases like ACM Digital Library, IEEE Xplore, Science Direct, Springer Link, etc. Additionally, forward citations of important papers in the research area have also been used for paper collection. Papers published in both conferences and journals have been used for the survey. Some relevant articles from other sources like arXiv, ProQuest, etc. have also been considered (Fig. 9b).




(a) (b)

Fig. 9. Distribution of surveyed papers (a) based on year of publication (b) based on their sources.

## B TABLE OF ABBREVIATIONS

Table 4 presents the meanings of abbreviations used in the paper, sorted by alphabetical order.

Table 4. Meanings of abbreviations used in the paper

| Abbreviation | Meaning |
|---|---|
| ABB | Adaptive Bollinger Bands |
| AC | Actor-Critic |
| A2C | Advantage Actor-Critic |
| A3C | Asynchronous Advantage Actor-Critic |
| bp | Basis points (1 bp = 0.01%) |
| BPTT | Back-Propagation Through Time |
| CARA | Constant Absolute Risk Aversion |
| CNN | Convolutional Neural Network |
| CRRA | Constant Relative Risk Aversion |
| DE | Differential Evolution |
| DJIA | Dow Jones Industrial Average |
| DPSO | Drift Particle Swarm Optimisation |

| Abbreviation | Meaning |
|---|---|
| DQL | Deep Q-Learning |
| DQN | Deep Q-Network |
| DRL | Deep Reinforcement Learning |
| EC | Evolutionary Computation |
| EDA | Estimation of Distribution Algorithm |
| ERL | Evolutionary Reinforcement Learning |
| ES | Evolutionary Strategies |
| ESG | Environmental, Social, and Governance |
| ETF | Exchange Traded Fund |
| FX | Foreign Exchange |
| GA | Genetic Algorithm |
| GP | Genetic Programming |
| GNP | Genetic Network Programming |
| GNP-CN | Genetic Network Programming with Control Nodes |
| GNP-RA | Genetic Network Programming with Rule Accumulation |
| HPSO | Heterogeneous Particle Swarm Optimisation |
| IMX | Importance Index |
| IPSO | Improved Particle Swarm Optimisation |
| LCS | Learning Classifier System |
| LSTM | Long Short-Term Memory |
| MACD | Moving Average Convergence Divergence |
| MADDPG | Multi-Agent Deep Deterministic Policy Gradient |
| MCTS | Monte Carlo Tree Search |
| MDP | Markov Decision Process |
| MOLPSO | Many Optimisation Liaisons Particle Swarm Optimisation |
| MORL | Multi-Objective Reinforcement Learning |
| MV | Mean-Variance |
| NLP | Natural Language Processing |
| NYSE | New York Stock Exchange |
| OHLC | Open-High-Low-Close |
| PG | Policy Gradient |
| PO | Portfolio Optimisation |
| POMDP | Partially Observable Markov Decision Process |
| PPO | Proximal Policy Optimisation |
| PSO | Particle Swarm Optimisation |
| QL | Q-Learning |
| RL | Reinforcement Learning |
| RRL | Recurrent Reinforcement Learning |
| RNN | Recurrent Neural Network |

| Abbreviation | Meaning |
|---|---|
| RS | Regime Switching |
| RSI | Relative Strength Index |
| SA | Simulated Annealing |
| SAC | Soft Actor-Critic |
| S&P 500 | Standard and Poor's 500 |
| SARSA | State-Action-Reward-State-Action |
| TA-GNP | Time Adapting Genetic Network Programming |
| TRPO | Trust Region Policy Optimisation |
| WRDS | Wharton Research Data Services |

## C TABLE OF NOTATIONS

In Table 5, we describe various notations used in the paper, ordered by appearance in the paper.

Table 5. Description of notations used in the paper

| Notation | Description |
|---|---|
| $S$ | Set of states in RL |
| $A$ | Set of actions in RL |
| $T$ | Transition function in RL |
| $R$ | Reward function in RL |
| $\gamma$ | Discount factor in RL |
| $\pi$ | Policy followed by RL agent |
| $V^{\pi}$ | Value function in RL |
| $Q^{\pi}$ | Action-value function in RL |
| $\pi_{\boldsymbol{\theta}}$ | Policy $\pi$ parameterised by $\boldsymbol{\theta}$ |
| $A^{\pi}$ | Advantage function in RL |
| $\alpha$ | Learning rate |
| $\mu^{\pi_\theta}$ | On-policy distribution under policy $\pi_\theta$ |
| $P$ | Population size in EC |
| $D$ | Dimensionality of search space in EC |
| $\mathbf{x^i}$ | $i^{th}$ candidate solution in EC |
| $x^i_j$ | $j^{th}$ dimension of $i^{th}$ candidate solution in EC |
| $f$ | Fitness function in EC |
| $N$ | Number of financial assets in the portfolio |
| $T_H$ | Horizon of trading or PO in terms of number of time-periods |
| $p^t_i$ | Price of asset $i$ at time $t$ |
| $n^t_i$ | Number of units of asset $i$ in the portfolio for the period $t$ |
| $c^t_i$ | Transaction costs incurred on asset $i$ for the period $t$ |

| Notation | Description |
|---|---|
| $c$ | Transaction cost constant (expressed in basis points) |
| $R^t$ | Value of portfolio after period $t$ |
| $w_i^t$ | Weight of asset $i$ in the portfolio for the period $t$ |
| $\bar{r}$ | Average return of the portfolio |
| $r_f$ | Risk-free return rate |
| $D_{max}$ | Maximum drawdown |
| $D_{avg}$ | Average drawdown |
| $\Delta_{n^t}$ | Number of units of asset traded in period $t$ |

## D TAXONOMIES

A taxonomy of RL is depicted in Fig. 10. RL algorithms can be broadly classified into two major categories: model-free RL and model-based RL. Model-free RL does not require a model of the environment while in model-based RL, a good enough model of the environment is utilised. However, a good enough model of the environment is not always available, especially in non-stationary environments like financial markets. Hence, model-free RL has been extensively used for financial applications. Model-free RL is further classified into 3 categories: value-based, policy-based, and actor-critic methods. A taxonomy of nature-inspired metaheuristics is depicted in Fig. 11 which can be classified into two categories based on their source of inspiration: evolutionary algorithms and swarm intelligence algorithms. Evolutionary algorithms are inspired by biological evolution while swarm-intelligence algorithms are inspired by the group behaviour of living organisms. It is worth noting that many researchers do not consider a clear distinction between these two categories due to their similar ways of operation. Quite often, algorithms like particle swarm optimisation are considered under the umbrella of evolutionary algorithms [54, 90, 167, 204, 205].

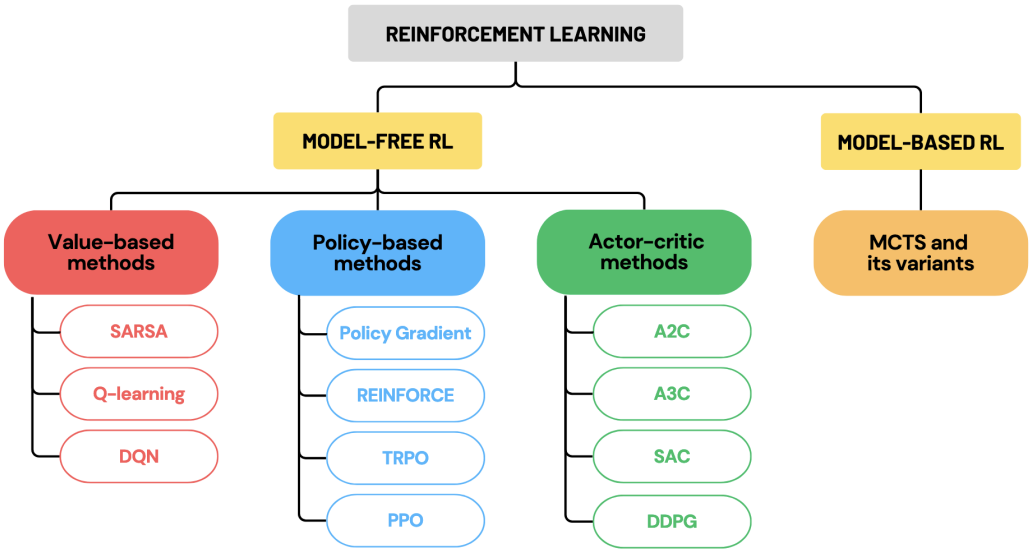



Fig. 10. Taxonomy of reinforcement learning

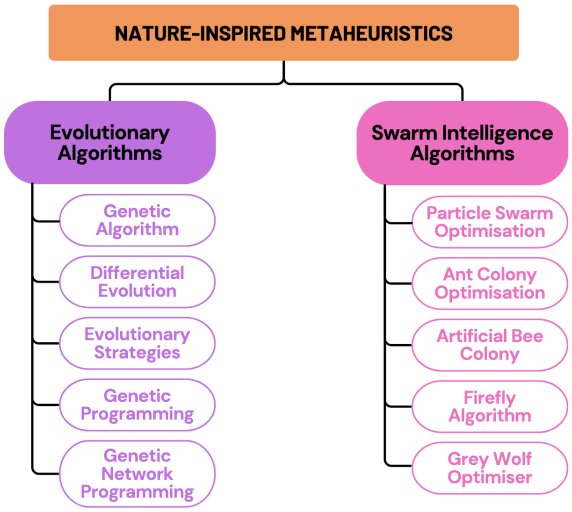



Fig. 11. Taxonomy of nature-inspired metaheuristics

## E DESIGN CHOICES

We present a compact reference to the design choices made in the literature surveyed in this paper. The choice of state features, actions and reward functions is crucial for RL. We discuss the various choices made by researchers for state features (Appendix E.1), actions (Appendix E.2) and reward functions (Appendix E.3). The fitness function is one of the most important components of EC algorithms. Appendix E.4 discusses the various choices of fitness functions made in the literature for trading and PO. In Appendix E.5, we mention the various RL and EC algorithms that have been hybridised for financial applications. We also provide a brief discussion on popular datasets (Appendix E.6), evaluation metrics (Appendix E.7), baselines and benchmarks (Appendix E.8).

### E.1 State features

In approaches where RL is the driver algorithm, price series of financial assets are often used as state features [130]. Most works focus on closing prices. According to efficient market hypothesis [118], only a few previous observations contain valuable information. However, the number of valuable past observations is a hyperparameter. Zhang et al. [202] use the past 60 observations of normalised close price series as state features. They also incorporate returns generated in the previous month, 2 months, 3 months, and a year as features in the state description. Jamali et al. [88] include the average prices of the previous week and month in their state description. Some works normalise the price series data before feeding it as state features. Zhang et al. [201] use the mean and standard deviation of returns from the training phase for performing standard normalisation. Maree et al. [119] use the wavelet transform on the prices. Indicators assist traders in detecting trends, reversal of trends, and other swings. Quite often, technical indicators like moving averages, relative strength index (RSI), etc. are also used as the state features [50]. These technical indicators are either obtained from financial data sources or calculated manually from price series and other data. The trading system proposed by Hryshko et al. [82] uses 10 commonly used indicators. Zhang et al. [201] incorporate fundamental and volatility indicators alongside technical indicators in their trading system. They use ten indicators: RSI 3, RSI

9, RSI 14, RSI 30, price to earnings ratio, price to cash flow ratio, debt to equity market ratio, positive volume index, negative volume index, and conditional volatility. Indicators derived from order flow and order book data have also been incorporated into the state representation as demonstrated by Austin et al. [9]. However, such data is not available in the public domain. In PO, current holdings in portfolio assets and balance in hand are also used as state features [195]. GNP-based systems [29] have also incorporated candlestick patterns alongside technical indicators.

### E.2 Actions

Most works addressing trading [13, 44, 82, 111] consider fixed position sizes ($\Delta_{n^t}$ is constant) and model actions in terms of discrete trading positions: {buy, sell, hold}. Maree et al. [119] use discretised actions to incorporate the quantity of purchase or sale of assets. Zhang et al. [202] study continuous action spaces where each action can take a value in the range [-1, 1], with -1 representing the maximally short position and 1 corresponding to the maximally long position. For PO, most works model actions as a vector of capital allocation ratios among portfolio assets. The allocation vector $\langle w_1^t, w_2^t, \ldots, w_N^t \rangle$ comprises values in the interval $[0, 1]$ summing to 1. Some works model actions as a vector with each element corresponding to the suitable action for the respective financial asset. Yang et al. [195] use a discretised action space where each portfolio asset is assigned an integer value $k \in \{-K, -(K-1), \ldots, -1, 0, 1, \ldots, K-1, K\}$, for $K > 0$. A positive value of $k$ indicates the purchase of $k$ units of financial assets while a negative value of $k$ indicates the sale of $k$ units. This approach results in an action space of size $(2K+1)^N$, with $N$ being the number of assets in the portfolio. Also, integrity checks need to be done in this approach to ensure that the agent does not buy more than its maximum capacity.

### E.3 Reward functions

Trading and PO systems are usually optimised by maximising utility functions of profit, wealth, etc. As the majority of traders and portfolio managers are risk-averse, risk-adjusted returns have also been used. Moody et al. [130] propose a differential version of the Sharpe ratio obtained using exponential moving average and standard deviation of returns. The differential Sharpe ratio facilitates recursive updating by avoiding the recomputation of mean and standard deviation of returns for the whole trading period. Also, it assigns more weightage to recent returns compared to older returns. It is very efficient for trading and PO systems that rely on gradient-based optimisation. The Sterling ratio is used in scenarios where there is a low tolerance for large losses [82]. Almahdi et al. [6] advocate the use of the Calmar ratio as the reward for PO. Zhang et al. [202] incorporate volatility scaling in their reward function: position size is increased when the market is less volatile and reduced when the market is more volatile. Apart from delayed rewards in terms of risk-adjusted returns, step rewards have also been used. The difference in the portfolio value at consecutive time steps, $R^t - R^{t-1}$, is used as a reinforcement signal by Dempster et al. [48]. Sometimes, there are convergence issues when using raw returns as the reward. This happens because the agent tries to minimise loss by maintaining the hold position. Costa et al. [44] introduce a negative reward for the hold action to deal with this issue. Apart from risk, sustainability is another factor that has been incorporated into the reward function. Maree et al. [119] use ESG score [37] to model sustainability.

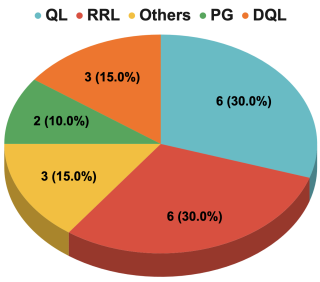
### E.4 Fitness functions

The choice of fitness functions in EC is similar to that of the reward functions in RL. Hryshko et al. [82] use the Sterling ratio as the fitness function while Almahdi et al. [7] advocate the use of Calmar ratio as the fitness function. Zhang et al. [201] use a variant of the Sharpe ratio as the fitness function. Dempster et al. [50] use returns generated by the

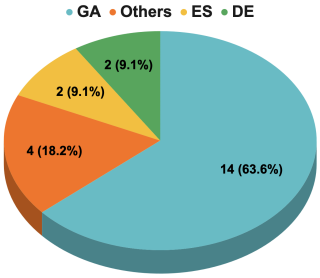
RL evaluator as fitness values. The combined profit generated by the portfolio assets has also been used as the fitness function [22]. Jia et al. [89] formulate the fitness function as the Sharpe ratio of the entire portfolio.

### E.5 Choice of algorithms

Fig. 12 depicts the different choices of EC and RL algorithms in approaches where EC helps RL (in both tightly and loosely coupled manner) for financial applications like trading and PO. RL is the final decision-making component here and EC helps in improving RL. QL and RRL are the most common RL approaches in this regime while some works are based on PG and other RL techniques. GA is the most commonly used EC algorithm while some works use DE, ES and other EC algorithms. Fig. 13 depicts the different choices of RL and EC algorithms in approaches where RL helps EC (in both tightly and loosely coupled manner) to tackle trading and PO. EC is the main decision-making component here and RL helps in improving EC. In this regime, GNP and its variants are the most commonly used EC algorithms while some works use PSO and other EC algorithms. SARSA is the most heavily used RL algorithm while some researchers use AC and other RL algorithms.

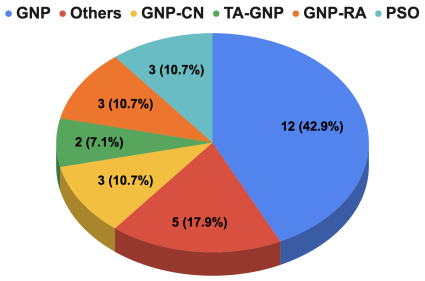



(a) Choice of RL algorithms

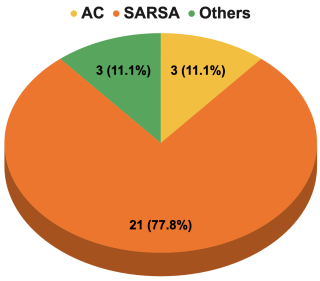



(b) Choice of EC algorithms

Fig. 12. Choice of algorithms in approaches where RL is improved by EC in finance.




(a) Choice of EC algorithms




(b) Choice of RL algorithms

Fig. 13. Choice of algorithms in approaches where EC is improved by RL in finance.

### E.6 Popular datasets

Although there is a wide variety of datasets available in the finance domain, we mention some prominent ones. Researchers have used data corresponding to various classes of financial assets like stocks, currencies, commodities, etc.

Yahoo Finance is one of the best public sources of freely available financial data. It also provides a Python library that can be used to extract data with much ease. Data of the S&P 500 companies is available at the Bloomberg Terminal. Although Bloomberg offers paid service, it also provides data about indicators. Dow Jones' stock data can be obtained from Wharton Research Data Services (WRDS). Historical data for foreign exchange markets can be obtained from CQG Data Factory. It also provides tick data which covers individual trades. Quandl is another source of financial datasets. Table 6 presents some popular data sources. The majority of the existing works have used real-world financial data (Fig. 6). Nonetheless, some works have used simulated financial time series data [182]. Butler et al. [21] simulate data as a random walk with auto-regressive trend processes [132]. Geometric Brownian motion is often used for simulating the prices of financial assets [1].

Table 6. Popular sources of financial data

| Data Source | Website |
|---|---|
| Yahoo Finance | https://finance.yahoo.com |
| Bloomberg Terminal | https://www.bloomberg.com/professional/solution/bloomberg-terminal |
| WRDS | https://wrds-www.wharton.upenn.edu |
| CQG Data Factory | https://www.cqg.com/ |
| Quandl | https://demo.quandl.com/ |
| Binance API | https://www.binance.com/en/binance-api |
| Trading View | https://www.tradingview.com |

### E.7 Evaluation metrics

Cumulative returns and annualised returns are two popular evaluation metrics in financial applications like trading and PO. Annual volatility in terms of the standard deviation of returns over a year is often used as a metric of risk. Maximum drawdown and average drawdown are some other metrics that have been used to capture risk. Since the majority of traders and PO managers are risk averse, risk-adjusted return metrics like the Sharpe ratio, Sterling ratio, Calmar ratio, etc. have been used for comparison in many works. One noteworthy point is that due to the involvement of stochasticity in most of the proposed approaches, the evaluation metrics should be computed as an average over multiple runs.

### E.8 Baselines and benchmarks

Buy-and-hold is a hard-to-beat baseline in markets with strong upward trends. Classical time series momentum strategies have also been used as baselines for comparing the performance of proposed approaches [202]. Some works have also used a zero intelligence random trading strategy as a baseline for comparison [201]. Trend-following and mean-reverting strategies used by practitioners can also be used as baselines for trading. Popular market indices like DJIA, S&P 500, etc. are often used as benchmarks for comparing the performance of portfolios. Some baselines used by researchers for PO include fixed allocation ratio, maximum diversification, uniform buy-and-hold, etc. The fixed allocation ratio strategy focuses on maintaining preset allocation ratios among assets by rebalancing the portfolio from time to time. The maximum diversification strategy focuses on incorporating minimally correlated assets in the portfolio to minimise the risk of losses through diversification. Uniform buy-and-hold allocates the funds uniformly to portfolio assets and holds the bought assets until the investment horizon. Although a wide variety of baselines have been used by researchers for trading and PO, there is a lack of commonly established benchmarks for comparison.