








Constrained Portfolio Management Using Action Space Decomposition for Reinforcement Learning

David Winkel^{1,2}(✉) , Niklas Strauß^{1,2} , Matthias Schubert^{1,2} ,
Yunpu Ma^{1,2} , and Thomas Seidl^{1,2} 

¹ LMU Munich, Munich, Germany

{winkel,strauss,schubert,ma,seidl}@dbs.ifi.lmu.de

² Munich Center for Machine Learning (MCML), Munich, Germany

Abstract. Financial portfolio managers typically face multi-period optimization tasks such as short-selling or investing at least a particular portion of the portfolio in a specific industry sector. A common approach to tackle these problems is to use constrained Markov decision process (CMDP) methods, which may suffer from sample inefficiency, hyperparameter tuning, and lack of guarantees for constraint violations. In this paper, we propose Action Space Decomposition Based Optimization (ADBO) for optimizing a more straightforward surrogate task that allows actions to be mapped back to the original task. We examine our method on two real-world data portfolio construction tasks. The results show that our new approach consistently outperforms state-of-the-art benchmark approaches for general CMDPs.

Keywords: Reinforcement Learning · Constrained Action Space · Decomposition · CMDP · Portfolio Optimization

1 Introduction

Constrained portfolio optimization is an important problem in finance. A typical example is a portfolio that must have at least 40% of the total portfolio value invested in environmentally friendly companies at each time step of the investment horizon or a portfolio that is not permitted to invest more than 20% in a particular industry sector. Another example of an action constraint task is a 130-30 strategy, in which the portfolio manager bets on group A of (potentially) overperforming stocks against group B of (potentially) underperforming stocks. This strategy is carried out by short-selling stocks worth 30% of the investment budget from Group B and leveraging the investment into stocks worth 130% of the investment budget from Group A.

The action space for these tasks can be considered as a continuous distribution of weights for a given set of assets. Therefore, reinforcement learning (RL) with policy gradient [16] is well-suited for this task. Because the invested capital

totals 100%, the codomain of the policy function is typically assumed to be a standard simplex. Existing solutions model the policy using a softmax output layer [1] or based on the Dirichlet distribution [20]. However, the constraints mentioned above cause a change in the shape of the policy’s codomain, making the standard solutions no longer directly applicable.

A way to optimize policies for tasks with constrained action spaces is by using approaches for CMDPs with constraints on the action spaces. However, state-of-the-art general approaches for CMDPs often have drawbacks such as expensive training loops, sample inefficiency, or only guarantees for asymptotical constraint compliance [2, 4, 10, 19, 21].

In this paper, we propose ADBO, a dedicated approach for dealing with the two important types of investment tasks mentioned previously: (a) investment tasks that invest *at least* or *at most* a certain percentage of a portfolio in a specific group of assets, and (b) short-selling tasks. ADBO can overcome the aforementioned shortcomings of general policy optimization methods for CMDPs. This is achieved by decomposing the non-standard-simplex action space into a surrogate action space. Solutions found in the surrogate action space can then be mapped back into the original constrained action space. In contrast to the non-standard-simplex action space, the surrogate action space is designed to be easily represented in the policy function approximator, allowing us to model the problem as a standard Markov decision process (MDP). Due to the lack of penalties and reward shaping, finding an optimal policy for an MDP is less complex than finding an optimal policy for a CMDP with constrained actions. Furthermore, ADBO ensures that the actions adhere to the constraints both during and after training.

In the experimental section, we demonstrate that the ADBO approach can handle two types of investment tasks using real-world financial data. The first task focuses on investing each time step at least a certain percentage of the portfolio in companies considered to be environmentally sustainable. The second task allows the agent to short-sell selected stocks, i.e., allowing for negative portfolio weights. Our proposed approach outperforms the state-of-the-art benchmark approaches for handling CMDPs on various criteria in both tasks.

2 Related Work

CMDPs were introduced by [3] to model constrained sequential decision tasks. constrained Reinforcement Learning (CRL) approaches for finding optimal policies for CMDPs have a wide range of applications, including finance [7, 20], autonomous electric vehicle routing [14], network traffic [9], and robotics [2, 8]. A **Trust Region**-based approach was introduced by [2] to find optimal policies for CMDPs that may still exhibit constraint violation due to approximation errors. Another approach proposed by [6] is based on **prior knowledge** and involves a one-time pretraining to predict simple one-step dynamics of the environment. **Lagrangian-based** approaches are another option for dealing with CMDPs. These approaches convert the original constraint optimization problem

into an unconstrained optimization problem by applying a Lagrangian relaxation to the constraints. Lagrangian-based approaches can be classified into two types: The first type is **Primal-Dual algorithms**, in which the Lagrange multipliers for a saddle point problem are chosen *dynamically* [5, 19]. The second type of Lagrangian-based approach employs **manually selected Lagrange multipliers**, which remain *static*, as shown in [13, 17]. Instead of a *saddle point problem*, as in the first type, using a static Lagrange multiplier transforms the problem into a maximization problem, which is more stable and computationally less expensive to solve. Some approaches carefully select Lagrange multipliers to model preferences in a trade-off problem rather than as a means to enforce constraints in an optimization problem. This is commonly seen in risk-return trade-off settings, such as in [7, 17, 20].

The **factorization of high-dimensional action spaces** in RL, i.e., splitting action spaces into smaller sub-action spaces as a Cartesian product, is an active area of research that has resulted in improved scalability and training performance. In their work, [11] introduce the Sequential DQN approach, which trains the agent for a sequence of n 1-dimensional actions rather than training the agent for n -dimensional actions of the original action space, effectively factorizing the original action space. The approach by [18] introduces an action branching architecture, which models the policies for the sub-action spaces in parallel. Our approach, like theirs, uses a Cartesian product of sub-action spaces. However, the sub-action spaces in our new approach ADBO are the outcome of a decomposition based on the Minkowski sum, resulting in a surrogate action space rather than a factorization of the original action space.

3 Problem Setting

We consider an agent that needs to allocate wealth across N different assets over T time steps. The allowed actions of the agent are defined by the investor’s investment task and are contained in the *constrained action space* \mathcal{A} . The investment task type $T1$ requires the investor to invest at least c_{T1} of the portfolio into assets from group V_{T1} . In practice, these group definitions are often linked to individual risk profiles, industry sectors, or features such as being an environmentally friendly investment. The action space for investment task type $T1$ is then defined as

$$\mathcal{A}_{T1} = \left\{ a \in \mathbb{R}^N : \sum_{i=0}^{N-1} a_i = 1, \sum_{i \in V_{T1}} a_i \geq c_{T1}, a_i \geq 0, c_{T1} > 0 \right\}$$

and represents an N -dimensional convex polytope. Task type $T1$ also includes cases that require investing *at most* c_{T1} into assets in V_{T1} because this case is equivalent to investing *at least* $(1 - c_{T1})$ into the remaining assets a_i for $i \in I \setminus V_{T1}$.

The investment task type $T2$ represents investors who believe that a group of assets V_{T2} will underperform relatively compared to the rest of the investment universe I . The investor pays a borrowing fee to short-sell assets in group

V_{T_2} worth $|c_{T_2}|$ of his total portfolio value and then uses the freed-up cash to invest $1 + |c_{T_2}|$ into assets of the other investment universe. The action space for investment task type T_2 is defined as

$$\mathcal{A}_{T_2} = \left\{ a \in \mathbb{R}^N : \sum_{i=0}^{N-1} a_i = 1, \sum_{j \in V_{T_2}} a_j = c_{T_2}, a_j \leq 0, a_k \geq 0 \forall k \in I \setminus V_{T_2}, c_{T_2} < 0 \right\}$$

and represents an N -dimensional convex polytope as well.

The **observation space** is defined as $\mathcal{O} = \mathcal{W} \times \mathcal{V} \times \mathcal{U}$ where $\mathcal{W} \subseteq \mathbb{R}^+$ is the current absolute wealth level, $\mathcal{V} \subseteq \mathbb{R}^N$ is the current relative portfolio weight of each of the N assets, and $\mathcal{U} \subseteq \mathbb{R}^N$ represents all the observed single asset returns from the previous time step.

The economic return of each asset is individually modeled for each time step by the random vector $\Theta = [\Theta_0, \dots, \Theta_{N-1}] \in \mathcal{U}$. The portfolio return is then a random variable with an expected value denoted as $\mathbb{E}[\Theta_{PF}] = a^\top \mathbb{E}[\Theta]$ with the portfolio weights $a \in \mathcal{A}$. There are two potential sources of cost to consider for the agent: First, the transaction costs caused by changes in the portfolio weights a_t in time step t by the agent defined as $tc_t = (|a_t - v_t|)^\top c$, where $v_t \in \mathcal{V}$ and vector $c = [c_0, \dots, c_{N-1}]$ represents the asset-specific transaction costs caused by trading a specific asset. Second, borrowing fees in case the agent is allowed to short-sell assets. These costs occur every period as long as assets are short-sold, i.e., assigned to a negative portfolio weight. The borrowing fees are defined as $bf_t = (\mathbb{1}_{a_i < 0} \circ a_t)^\top b$ where $\mathbb{1}_{a_i < 0}$ is an indicator vector signaling for each individual asset a_i if the current portfolio weight is negative, \circ is an operator for element-wise vector multiplication, and the vector $b = [b_0, \dots, b_{N-1}]$ represents asset-specific borrowing fees per time step.

The **reward** for the agent is a combination of transaction costs tc , borrowing fees bf , and a realization ϑ_{PF} of the random variable of the portfolio's economic return Θ_{PF} , i.e., $r = \vartheta_{PF} - tc - bf$. The agent's goal is to maximize the expected cumulative reward, which we will refer to as *total economic payoff*.

4 Solution as CMDP

A CMDP is an extension of an MDP and is defined as a tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma, \mathcal{C})$ where \mathcal{S} is the set of states, \mathcal{A} is the set of actions, R is the immediate reward function, which maps transition tuples to their respective expected reward, i.e., $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. P denotes the transition probability function, whereas $P(s_{t+1}|s_t, a_t)$ gives the probability of transitioning to state $s_{t+1} \in \mathcal{S}$ given state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$. The parameter $\gamma \in [0, 1)$ represents a discount factor. $\mathcal{C} = \{C_0, \dots, C_m\}$ is a set of immediate constraint functions $C_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ for $i \in \{0, \dots, m\}$ that map transition tuples to the respective cost. We let $r_{t+1} := R(s_t, a_t, s_{t+1})$ and define the return for a trajectory τ as the observed discounted cumulative rewards. The objective function J is then defined as the expected return for a given policy π , i.e., $J(\pi) := \mathbb{E}_{\tau \sim P(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_{t+1} \right]$.

The expected cumulative discounted immediate cost for constraint i under policy π is defined as $J_{C_i}(\pi) := \mathbb{E}_{\tau \sim P(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t C_i(s_t, a_t, s_{t+1}) \right]$. We also define constant trajectory constraint limits d_0, \dots, d_m . The optimization problem for the CMDP is then defined as:

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad J(\pi) &= \mathbb{E}_{\tau \sim P(\tau|\pi)}(G) = \mathbb{E}_{\tau \sim P(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_{t+1} \right] \\ \text{s.t.} \quad J_{C_i}(\pi) &\leq d_i \quad \forall i \end{aligned}$$

In the following, we will show how to formulate the tasks defined in Sect. 3 as a CMDP. In Sect. 3, we defined the observation space \mathcal{O} , the constrained action space \mathcal{A}_i , and the reward R . The transition function P and the state space \mathcal{S} are unknown. However, we assume that we can sample transitions from an environment. Therefore, we can employ reinforcement learning based on a learned state representation function to learn effective policies. To address the action constraints of tasks $T1$ and $T2$, we define the following cost function for each respective task $i \in \{1, 2\}$: $C_{T_i}(s_t, a_t, s_{t+1}) = \mathbb{1}_{a_t \notin \mathcal{A}_{T_i}} \cdot \zeta$ where constant $\zeta > 0$ indicates the non-zero cost of a constraint violation. The respective constraint for each task is then defined as $J_{C_{T_i}}(\pi) \leq 0$.

5 Action Space Decomposition Based Optimization

We define a surrogate MDP $(\mathcal{S}, \tilde{\mathcal{A}}, R, P, \gamma)$ and ensure that there exists a surjective function $f: \tilde{\mathcal{A}} \rightarrow \mathcal{A}$ that allows reaching any $a \in \mathcal{A}$ from at least one $\tilde{a} \in \tilde{\mathcal{A}}$. For a formal description of our method, we first introduce the Minkowski sum:

Definition 1. *Given two sets A and B of vectors in n -dimensional Euclidean space, the **Minkowski sum** of A and B is generated by adding each vector in A to each vector in B , i.e., the set $A + B = \{a + b | a \in A, b \in B\}$ in which we refer to A and B as Minkowski summands.*

In our setting, the Minkowski sum describes how multiple decomposed action sets can be combined to reconstruct the original constraint action set. The masked scaled standard simplex (MSSS) describes a part of the original constrained action which can be described as a simplex:

Definition 2. *Let mask $M \subseteq \{0, \dots, N - 1\}$. MSSS is defined as:*

$$\text{MSSS}_{M,c} = \left\{ y \in \mathbb{R}^N : \sum_{j \in M} y_j = c, y_i = 0 \forall i \in I \setminus M \right\}$$

with either $(c \geq 0 \wedge y_i \geq 0 \forall i \in M)$ or $(c < 0 \wedge y_i \leq 0 \forall i \in M)$.

The surrogate action space is modeled as the Cartesian product of independent sub-action spaces $\tilde{\mathcal{A}} = \tilde{\mathcal{A}}_1 \times \tilde{\mathcal{A}}_2$. The sub-action spaces $\tilde{\mathcal{A}}_i$ with $i \in \{1, 2\}$ are required to have the two properties: (a) being a decomposition of \mathcal{A} in such a way

that the Minkowski sum (see Definition 1) of all the sub-action spaces $\tilde{\mathcal{A}}_i$ is \mathcal{A} , i.e., $\mathcal{A} = \tilde{\mathcal{A}}_1 + \tilde{\mathcal{A}}_2$, and (b) being an MSSS as defined in Definition 2. Property (a) guarantees the existence of function f that can be defined as $f(\tilde{a}) = \tilde{a}_1 + \tilde{a}_2 = a$ with $\tilde{a} = [\tilde{a}_1, \tilde{a}_2] \in \tilde{\mathcal{A}} \subset \mathbb{R}^{2N}$ and $\tilde{a}_i \in MSSS_i \subset \mathbb{R}^N$ for $i \in \{1, 2\}$, i.e., a summation of vectors in a subspace of \mathbb{R}^N . Property (b) allows utilizing well-established RL methods for handling standard simplex action spaces with only minor modifications by adding a scaling and masking logic in order to model single MSSS action spaces.

The following two theorems show that constrained action spaces as defined in Sect. 3 can be decomposed into two MSSS that satisfy both of the requirements mentioned above. Theorem 1 describes the decomposition for task T1:

Theorem 1. *Any convex polytope $P \neq \emptyset$ defined as*

$$\sum_{i \in I} x_i = 1, x_i \geq 0 \quad \forall i \in I, \sum_{i \in V_1} x_i \geq c_1$$

with $c_1 > 0$, $I = \{0, \dots, N-1\}$ and $V_1 \subseteq I$ can be decomposed into two MSSSs:

$$MSSS_{S_1, z_1}, \text{ i.e. } \forall y_1 \in MSSS_{S_1, z_1} : \sum_{S_1} y_{i,1} = z_1 \quad \text{with } S_1 = V_1 \text{ and } z_1 = c_1$$

$$MSSS_{S_2, z_2}, \text{ i.e. } \forall y_2 \in MSSS_{S_2, z_2} : \sum_{S_2} y_{i,2} = z_2 \quad \text{with } S_2 = I \text{ and } z_2 = 1 - c_1$$

so that the Minkowski sum of the MSSSs equals the original polytope P .

Correspondingly, the following theorem formulates the decomposition of the action space in task T2:

Theorem 2. *Any convex polytope $P \neq \emptyset$ defined as*

$$\sum_{i \in I} x_i = 1, \sum_{i \in V_1} x_i = c_1, x_i \geq c_1 \quad \forall i \in V_1, x_i \leq 0 \quad \forall i \in V_1, x_i \geq 0 \quad \forall i \in I \setminus V_1$$

with $c_1 < 0$, $I = \{0, \dots, N-1\}$ and $V_1 \subseteq I$ can be decomposed into two MSSSs:

$$MSSS_{S_1, z_1}, \text{ i.e. } \forall y_1 \in MSSS_{S_1, z_1} : \sum_{S_1} y_{i,1} = z_1 \quad \text{with } S_1 = V_1 \text{ and } z_1 = c_1$$

$$MSSS_{S_2, z_2}, \text{ i.e. } \forall y_2 \in MSSS_{S_2, z_2} : \sum_{S_2} y_{i,2} = z_2 \quad \text{with } S_2 = I \setminus V_1 \text{ and } z_2 = 1 - c_1$$

so that the Minkowski sum of the MSSSs equals the original polytope P .

Theorem 1 and 2 can be proven by showing that the two sets of closed half-spaces, one describing the polytope P and the other describing the Minkowski sum of the two MSSSs, are equal resulting in the equality of the two polytopes.

ADBO is based on the PPO algorithm [15]. The agent's policy network is designed in such a way that the action representation is distributed across multiple *independent* segments, i.e., one head for each MSSS. A *shared* state encoder,

on the other hand, provides a learned state representation to both heads. For the state encoder we use a neural network of four fully connected layers of size 1024, 512, 256, and 64 with ReLU activation functions followed by GTrXL element allowing to handle tasks requiring memory. The GTrXL element is based on [12]. The element is composed of a single *transformer unit* with a single encoder layer as well as a single decoder layer with four attention heads and an embedding size of 64. While the sub-actions in ADBO are stochastically independent, the parameters of the two distributions from which the sub-actions are drawn are partially coordinated, i.e., parts of the actions rely on the same shared latent state representation. To further ensure coordination between the sub-actions, the different sub-actions are all evaluated using a joint reward. This means that if a *joint action* performs poorly, all independent segments receive a poor reward signal, regardless of individual sub-action performance.

We use a Dirichlet distribution to model each MSSS in the architecture of the policy function approximator. The expected value of a random vector $X = [X_0, \dots, X_{N-1}]$ following a Dirichlet distribution with a parameter vector of $\alpha = [\alpha_0, \dots, \alpha_{N-1}]$ is defined as $\mathbb{E}[X_i] = \alpha_i \cdot \left(\sum_{n=0}^{N-1} \alpha_n \right)^{-1}$ with $\alpha_i > 0$ for $i \in \{0, \dots, N-1\}$. By adjusting the parameter vector of a Dirichlet distribution and applying a linear scaling transformation, we can create a random variable with the set of all possible realizations equaling $MSSS_{M,c}$. The set M contains index values which we map to an N -dimensional indicator vector $\mathbb{1}_M$, with the vector's elements set to one if their respective index occurs in M and zero otherwise. The parameter vector passed to the Dirichlet distribution is calculated as $\alpha_{\mathbb{1}_M} = \max(\alpha \circ \mathbb{1}_M, \epsilon)$, where α is the initial parameter vector before applying the masking and $\epsilon > 0$ is an arbitrary small number. The operator \circ represents element-wise multiplication for vectors. In the final step, a linear scaling transformation is applied, i.e., $Y = c \cdot X$ with $X \sim Dir(\alpha_{\mathbb{1}_M})$.

ADBO requires the uses of two MSSSs, i.e., $MSSS_{M_1,c_1}$ and $MSSS_{M_2,c_2}$. It should be noted that the gradient of the policy during training is based on a policy interacting with the surrogate action space $\tilde{\pi}(\cdot|s)$ rather than a policy interacting with the original constrained action space $\pi(\cdot|s)$. We only use f to convert \tilde{a} into a representation a that can interact with the environment. Various inputs \tilde{a} for f may sum to the same output value a , resulting in f being a many-to-one function. For some $\tilde{a} \in \tilde{\mathcal{A}}$, this results in $\mathbb{P}(\tilde{a}|s) \neq \mathbb{P}(a|s)$ with $a = f(\tilde{a})$. However, we argue that finding *one* possible representation for an action a belonging to an optimal policy for the original problem is sufficient from an optimization standpoint.

6 Experiments

The environment is based on [20], and uses the same real-world financial data from the Nasdaq-100 index that was fetched and processed using the qlib package.¹ The investment universe of the environment consists of 13 assets, one of

¹ <https://github.com/microsoft/qlib/tree/main>.

which is cash. The remaining 12 assets are chosen at random from a list of 35 stocks that remain after filtering the Nasdaq-100 data set for companies that have been part of the index since January 1, 2010 and have no missing data. The original Nasdaq-100 data set is supplemented with the Environmental Score Metric (ESM) assigned by financial data provider LSEG.² The score rates a company’s environmental sustainability based on various evaluation categories, such as carbon emissions, willingness to innovate in this field, and transparency in reporting relevant information. The score ranges from 0 to 100, representing the percentiles of a ranking system.

We compare **ADBO** to three other state-of-the-art approaches for optimizing policies in CMDPs. **RCPO** is proposed by [19] and belongs to the class of Lagrangian-based approaches. The interior-point policy optimization approach **IPO** is introduced by [10]. **P3O** is proposed by [21] and uses a first-order optimization over an unconstrained objective with a penalty term equal to the original constraint objective. All benchmark approaches are implemented in the RLLib framework³ based on their papers and publically available.⁴

Two experimental settings are examined: the **SUSTA setting** is based on task type *T1*. The investor must invest at least 40% of his capital in the top 20% of environmentally sustainable companies, i.e., companies with an ESM score of 80 or higher. A score of 80 or higher “indicates excellent relative [...] performance and a high degree of transparency in reporting material” by a company.⁵ The **SHORT setting** is based on task type *T2*. It employs a **130-30 strategy**, a popular long/short equity strategy among investors to invest 130% of the available capital in stocks they believe will outperform and short-sell stocks worth 30% of the available capital they believe will underperform. In the experiments, we choose the companies Automatic Data Processing Inc., Paccar Inc., and Amgen Inc. to be sold short based on being the worst performers in 2020, the final year before the start of the backtesting period.

\mathcal{A}_{short} is not a subset of the standard simplex because negative weights are permitted. As a result, the RCPO, IPO, and P3O approaches must be modified to be applicable to SHORT setting. The agent performed very poorly in initial tests using \mathbb{R}^N as a base action space and applying constraints accordingly and was unable to learn meaningful policies. Instead, using a standard simplex as the base action space and applying action scaling produced better results. For *action scaling*, the agent uses the output of a Dirichlet distribution as an encoded action $\tilde{a} = [\tilde{a}_0, \dots, \tilde{a}_{N-1}]$ that is then transformed, i.e., scaled into the final action $a = [a_0, \dots, a_{N-1}]$: the cumulative weights of the stocks sold *short* and the cumulative weights of the stocks bought *long* are added up in their absolute values, resulting in a scaling factor $\alpha_{total} = |\alpha_{long}| + |\alpha_{short}|$. Then, for all elements i of the encoded action $a_i = \tilde{a}_i \cdot \alpha_{total}$ that are bought, a positive scaling factor is applied, and for all elements j of the encoded action $a_j = \tilde{a}_j \cdot (-\alpha_{total})$

² <https://www.lseg.com/>.

³ <https://docs.ray.io/en/master/rllib/index.html>.

⁴ https://github.com/DavWinkel/RL_ADBO.

⁵ <https://www.refinitiv.com/en/sustainable-finance/esg-scores>.

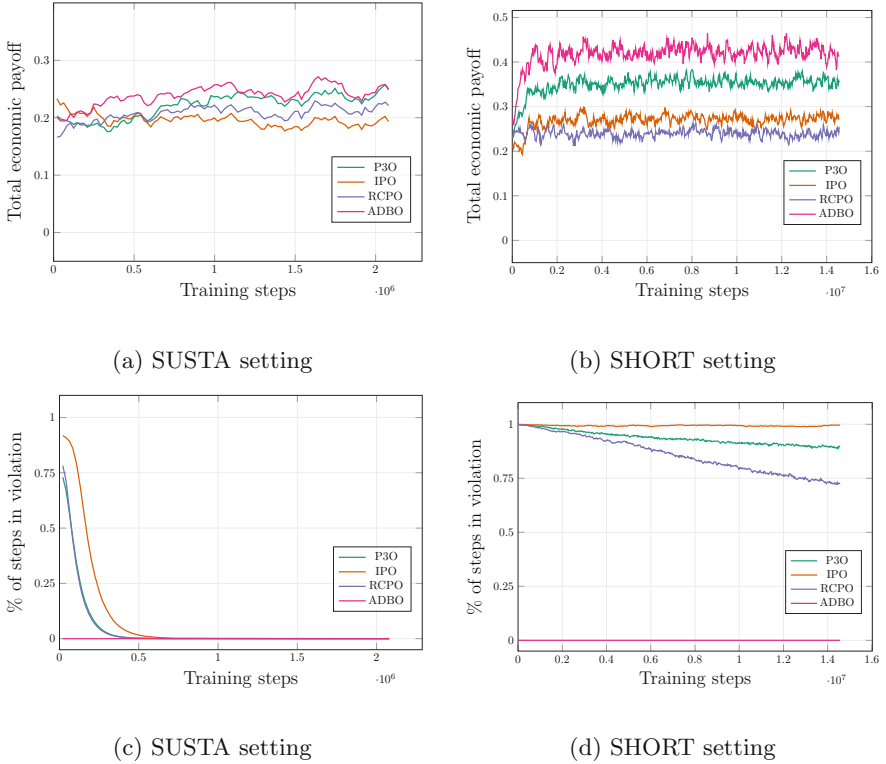


Fig. 1. Performance during training for all four approaches in the SUSTA setting and the SHORT setting regarding Total economic payoff and % of steps in violation.

that are sold short, a negative scaling factor is applied. It should be noted that actions generated as described above are no longer guaranteed to sum up to 1.0. Because IPO is a logarithmic barrier function-based approach that does not apply to equality constraints, we must additionally soften equality constraints of the form $x = c$ to inequality constraints that allow values in a α -neighborhood of x , i.e., $x \leq c + \alpha$ and $x \geq c - \alpha$.

To evaluate the four approaches, we will report performance during and after training for both the SUSTA setting and the SHORT setting. The total economic payoff defined in Sect. 3 is used to measure economic performance. The results of the SUSTA setting will be discussed first. The training in the SUSTA setting lasts 500 iterations and consists of approximately 2.1 million training steps. Figure 1a shows that ADBO and P3O perform best during training by steadily improving their total economic payoff. RCPO also shows improvements, although at a much slower rate. Table 1 shows the evaluation of economic performance following training completion in two setups: in the **(A) environment setup** 1000 trajectories are sampled from the same environment used for the training. ADBO generates the highest total economic payoff in the SUSTA setting, followed by

P3O, RCPO, and IPO. In the **(B) backtesting setup** a single trajectory, namely the real-world Nasdaq-100 trajectory in 2021, is used for evaluation. In the backtesting year 2021, the overall yearly performance of the Nasdaq-100 index was above average, returning 27.5%, indicating that the individual stocks that comprise the index were also performing well. As a result, the four approaches generated high returns in the (B) backtesting setup, with ADBO performing best, followed by P3O. In the SHORT setting, the training time had to be increased significantly. This increase was required because IPO, RCPO, and P3O failed to generate constraint-compliant actions satisfactorily. However, due to insufficient training progress, which will be discussed in detail later in this section, the training was eventually stopped after 3500 iterations, consisting of approximately 14.7 million training steps. Figure 1b depicts the evolution of the total economic payoff during training. After roughly 1 million training steps, the performance of ADBO converges to a level that it then maintains for the remainder of the training. P3O improves its performance over 3 million training steps until it reaches a stable level. IPO improves its performance during the first million training steps and then stabilizes, whereas RCPO does not show significant improvements in total economic payoff during training. Table 1 shows the performance evaluation in the SHORT setting after the training is completed. In the (A) environment setup, ADBO performs best, with an average total economic payoff of 42.72%, followed by P3O with 35.12%. ADBO outperforms its benchmark approaches by a wide margin in the (B) backtesting setup, achieving a total economic payoff of 102.05%.

The experiments show that violations of the action constraints occurred during the training of IPO, RCPO, and P3O in the SUSTA setting. Figure 1c shows that this is especially true at the beginning of the training phase, while the number of time steps with actions in violation decreases almost to zero later on. After completion of the training in the (A) environment setup, RCPO is the only approach generating actions in violations, as shown in Table 1. However, violations occur only on a small number of time steps, i.e., nine out of 12'000 time steps. All approaches are free of constraint violations in the (B) backtesting setup. For the SHORT setting, the majority of actions generated by the approaches IPO, RCPO, and P3O violated the constraints during training. However, as training time progresses, the number of actions in constraint violation decreases for RCPO and P3O. As a result, the training time was increased sevenfold when compared to SUSTA setting. Nevertheless, the training was eventually halted due to insufficient speed in reducing constraint violations. Figure 1d shows the best-performing variants of the agents after extensive tuning of their hyperparameters. Table 1 displays the evaluation results after the training in the SHORT setting was completed. In the SHORT setting, IPO, RCPO, and P3O fail to generate results free of constraint violations for both the (A) environment and (B) backtesting setups. ADBO, on the other hand, guarantees by design actions free of violations during and after training.

Table 1. Evaluation after training is completed. (A) environment setup has a total of 12'000 time steps (1000 trajectories), (B) backtesting setup has a single trajectory with 12 time steps.

	SUSTA setting		SHORT setting	
	Total econ. payoff (12 months)	Total violations	Total econ. payoff (12 months)	Total violations
(A) environment				
RCPO	0.2238	0	0.2418	8656
IPO	0.2013	0	0.2721	11943
P3O	0.2561	9	0.3512	10865
ADBO (Ours)	0.2603	0	0.4272	0
(B) backtesting				
RCPO	0.4640	0	0.5285	9
IPO	0.3499	0	0.6262	12
P3O	0.5475	0	0.7654	11
ADBO (Ours)	0.5758	0	1.0205	0

7 Conclusion

In this paper, we train agents to manage investment portfolios over multiple periods, given two types of tasks that are commonly encountered in practice. Task type $T1$ constrains the allocation of a particular group of assets, e.g., assets belonging to a specific industry sector. Task type $T2$ requires the investor to short-sell one group of assets while increasing the investment in another. We propose ADBO, which finds a performant policy for a surrogate MDP rather than for the more complex CMDP. The surrogate MDP is based on an action space decomposition of the original action space. We show that ADBO outperforms general CMDP approaches for both task types in experimental settings. For future work, we will examine extensions of action space decomposition based on the Minkowski sums to a broader group of convex polytopes.

References

1. Abrate, C., et al.: Continuous-action reinforcement learning for portfolio allocation of a life insurance company. In: Dong, Y., Kourtellis, N., Hammer, B., Lozano, J.A. (eds.) ECML PKDD 2021. LNCS (LNAI), vol. 12978, pp. 237–252. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86514-6_15
2. Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained policy optimization. In: International Conference on Machine Learning, pp. 22–31. PMLR (2017)
3. Altman, E.: Constrained Markov decision processes: stochastic modeling. Routledge (1999)
4. Ammar, H.B., Tutunov, R., Eaton, E.: Safe policy search for lifelong reinforcement learning with sublinear regret. In: International Conference on Machine Learning, pp. 2361–2369. PMLR (2015)

5. Bhatnagar, S., Lakshmanan, K.: An online actor-critic algorithm with function approximation for constrained Markov decision processes. *J. Optim. Theory Appl.* **153**(3), 688–708 (2012)
6. Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., Tassa, Y.: Safe exploration in continuous action spaces. arXiv preprint [arXiv:1801.08757](https://arxiv.org/abs/1801.08757) (2018)
7. Di Castro, D., Tamar, A., Mannor, S.: Policy gradients with variance related risk criteria. arXiv preprint [arXiv:1206.6404](https://arxiv.org/abs/1206.6404) (2012)
8. Gu, S., Holly, E., Lillicrap, T., Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 3389–3396. IEEE (2017)
9. Hou, C., Zhao, Q.: Optimization of web service-based control system for balance between network traffic and delay. *IEEE Trans. Autom. Sci. Eng.* **15**(3), 1152–1162 (2017)
10. Liu, Y., Ding, J., Liu, X.: Ipo: Interior-point policy optimization under constraints. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 4940–4947 (2020)
11. Metz, L., Ibarz, J., Jaitly, N., Davidson, J.: Discrete sequential prediction of continuous actions for deep RL. arXiv preprint [arXiv:1705.05035](https://arxiv.org/abs/1705.05035) (2017)
12. Parisotto, E., et al.: Stabilizing transformers for reinforcement learning. In: International Conference on Machine Learning, pp. 7487–7498. PMLR (2020)
13. Peng, X.B., Abbeel, P., Levine, S., Van de Panne, M.: DeepMimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph. (TOG)* **37**(4), 1–14 (2018)
14. Qin, Z., Chen, Y., Fan, C.: Density constrained reinforcement learning. In: International Conference on Machine Learning, pp. 8682–8692. PMLR (2021)
15. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint [arXiv:1707.06347](https://arxiv.org/abs/1707.06347) (2017)
16. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems 12 (1999)
17. Tamar, A., Mannor, S.: Variance adjusted actor critic algorithms. arXiv preprint [arXiv:1310.3697](https://arxiv.org/abs/1310.3697) (2013)
18. Tavakoli, A., Pardo, F., Kormushev, P.: Action branching architectures for deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)
19. Tessler, C., Mankowitz, D.J., Mannor, S.: Reward constrained policy optimization. In: International Conference on Learning Representations (2018)
20. Winkel, D., Strauß, N., Schubert, M., Seidl, T.: Risk-aware reinforcement learning for multi-period portfolio selection. In: Amini, M.R., Canu, S., Fischer, A., Guns, T., Kralj Novak, P., Tsoumakas, G. (eds.) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2022. LNCS, vol. 13718. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-26422-1_12
21. Zhang, L., et al.: Penalized proximal policy optimization for safe reinforcement learning. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, pp. 3744–3750 (2022)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

