SAFE EXPLORATION IN LINEAR EQUALITY CONSTRAINT

Anonymous authors
Paper under double-blind review

ABSTRACT

With the extensive research and application, some shortcomings of reinforcement learning methods are gradually revealed. One of the considerable problems is that it is difficult for reinforcement learning methods to strictly satisfy the constraints. In this paper, a Singular Value Decomposition-based non-training method called 'Action Decomposition Regular' is proposed to achieve safe exploration. By adopting linear dynamics model, our method decomposes the action space into a constraint dimension and a free dimension for separate control, making policy strictly satisfy the linear equality constraint without limiting the exploration region. In addition, we show how our method should be used when the action space is limited and convex, which makes the method more suitable for real-world scenarios. Finally, we show the effectiveness of our method in a physically-based environment and prevail where reward shaping fails.

1 INTRODUCTION

In the past ten years, reinforcement learning(RL)(Sutton & Barto, 2018) has made significant breakthroughs in many fields, such as games(Mnih et al., 2013; Schaul et al., 2015; Mnih et al., 2015; Hasselt et al., 2015; Wang et al., 2016), robotics(Gu et al., 2017), autonomous vehicles(Sallab et al., 2017), healthcare(Yu et al., 2019). In the reinforcement learning task, the agent can obtain the policy of making the action that maximizes the long-term return. Although it can improve one's own policy through trial and error learning under the interaction with the environment, it is difficult to strictly ensure the safety of the actions output by its policy(García et al., 2015). Therefore, the constraint problem has become one of the active research contents in reinforcement learning recently.

In the application, making such actions that violate constraints will bring serious consequences in some fields. Therefore never violating these constraints is a strict necessity in many scenarios, such as the stability of robots and avoidance of pedestrians or obstacles appearing in front of the vehicle during autonomous driving(Levinson et al., 2011; Amodei et al., 2016). In the real world, the linear equality constraints are relatively common, for example, we want the robot to achieve a certainly required configuration on a certain trajectory, where the constraint may appear at different instants in any dimension; or the robot center of mass is restricted at the beginning of the movement(Laine & Tomlin, 2019). And all these complex constraints typically take the form of linear equality constraints. Therefore, it is necessary to have a method that can ensure these constraints to be strictly satisfied in the real world.

Researchers have carried out much meaningful research on how to better satisfy the constraint. Dalal et al. (2018) achieve good results in satisfying hard constraints, but it relies heavily on the security layer of data training and cannot cross domains. Tessler et al. (2019) can solve the mean value constraints or discounted sum constraints, but there is no guarantee that the constraints can be met during the training process. More importantly, the existing learning-based methods can hardly satisfy the constraints. In fact, the constraint guarantee for the agent's behavioral decision-making benefits from knowledge about the causal mechanism that controls it, such as the dynamic model(Fisac et al., 2019). Fortunately, the designer of an agent always knows or approximately knows its dynamics(Fisac et al., 2019). For example, Lutter et al. (2020) adopt the linear dynamic model of the robot and finally, make the optimal strategy policy their action limit. This inspires people to find a balance between data-driven and model-based technology.

Among the existing model-based methods, the idea of using the linear dynamic model is common(Aswani et al., 2013; 2012). Although most robots have nonlinear dynamic models, there are already many methods based on the linearization of the model. For example, sequential quadratic programming requires the continuous local approximation of the problem and then transforms it into the constrained linear quadratic regulator problem(Giftthaler et al., 2018). And iLQR(Levine & Koltun, 2013) is a method with linearizing a nonlinear model, which often appears as baselines in experiments about model-based reinforcement learning. And there are many theories about the stability of linearized systems(Spong, 1995; Russ, 2021). For convenience, this paper only discusses the case of the linear dynamic model.

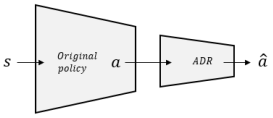


Figure 1: ADR is used to modify the action output by policy in the process of training and application. s denotes state, a denotes action. And $\hat{a}$ denotes modified action by ADR.

In this paper, we propose the 'Action Decomposition Regular'(ADR) as shown in Fig 1. Using Singular Value Decomposition(SVD) approach, ADR decomposes the action space into a constraint dimension containing all constraint information and the remaining free dimension. The goal is to achieve better policy exploration without violating linear equality constraints at all. Under the above idea, we find a balance between the model-based technology's control of constraints and the data-driven policy learning method. It is worth mentioning that our method is non-training and can conjunct any efficient continuous-control RL method. The main contributions of this paper are as follows:

1. We propose a non-training method called ADR that can make the reinforcement learning strictly satisfy the constraints without restricting the system's ability to explore. And the method does not need to make assumptions about the dimensions of the constraints.
2. We give an action correction scheme with the property of Pareto optimal solution(Van Moffaert & Nowé, 2014) in convex action space and give the proof.
3. The effectiveness of the method is verified in a simulation environment with physical properties. The simulation shows good results where reward shaping fails.

2 Related Work

Implementing policy security through constrained reinforcement learning is an active research content(Amodei et al., 2016).

The algorithm based on Constrained Markov Decision Processes (CMDP)(Kallenberg, 1983; Ross, 1985; Ross & Varadarajan, 1989; Altman, 1999; Le et al., 2019) is a common method. CPO(Achiam et al., 2017) is an algorithm based on CMDP, mainly inspired by TRPO(Schulman et al., 2015), to find a surrogate function that is the lower bound of the original objective function and the upper bound of the original constraint. RCPO(Tessler et al., 2019) uses the idea of PPO(Schulman et al., 2017; Heess et al., 2017), introduces the lagrange method, and solves the problem based on the adaptively updated lagrange multiplier. And a RCPO-based method uses PID to control the lagrange multiplier(Stooke et al., 2020). Recently Zhang et al. (2020) propose FOCOPS, which first finds the optimal update policy by solving a constrained optimization problem in the non-parameterized policy space, then projects the updated policy back into the parametric policy space. However, these methods require a long training process. They are shown to solve the mean value constraints or discounted sum constraints. As such, it is difficult to ensure that the constraints are met as much as possible during the training process, even for any simple constraints.

Modifying the exploration process is another way to solve the constraint problem. In Dalal et al. (2018), their method requires first using data to train a security layer to modify actions according to certain criteria. Although they have achieved excellent results in their experiments, the problem is that security is very dependent on the security layer, and the linear relationship of the predicted cost may not be established. The solution of Amos & Kolter (2017) relies on a complete Quadratic Programming solver, but their solution is too expensive to calculate.

In addition, there are many methods that agree to be model-based. One possible approach is to try to perform imitation learning on the trajectory obtained by the model-based optimal control policy, i.e., DAgger(Ross et al., 2011). But as stated by Bellegarda & Byl (2020), when facing with areas of

Table 1: Comparison between various method

	Satisfies constraints during training	**Combines with any learning-based continuous-control algorithm**	**Never limit the exploration region**	**No limitation in dimension**
ADR(this paper)	✓	✓	✓	✓
Achiam et al. (2017)	×	×	×	×
Dalal et al. (2018)	✓	✓	✓	×
Reward shaping	×	✓	✓	✓

state space that the expert trajectory has not visited before, policy learned only from expert data may perform poorly in these areas. And Fisac et al. (2019) propose a general safety framework based on Hamilton–Jacobi reachability methods. This safety framework also can work in conjunction with any efficient learning algorithm. But this method is computationally intensive and limited in dimension. Aswani et al. (2013) use the method about the robust model-predictive control approach and achieve good results in some problems such as quadrotor flight. But it limits the exploration ability of the system. And Berkenkamp et al. (2016; 2017) both limit the exploration region of the method. The method in Sadraddini & Belta (2016) is conservative since it does not update the model.

Reward shaping is a natural alternative to constraints, influencing the agent by artificially shaping negative rewards in the state space(Dalal et al., 2018; Ng et al., 1999). But it often needs to design a modified reward function through expert knowledge(Randløv & Alstrøm, 1998) or neural network methods(Burda et al., 2018) in advance. In other words, it needs to know the occurrence of constraints in advance, but many urgent constraints are sudden.

Our method overcomes the shortcomings mentioned above. A comparison with the different approaches is provided in Table 1.

3 PRELIMINARIES

3.1 MARKOV DECISION PROCESS (MDP)

A Markov Decision Process (MDP) (Sutton & Barto, 2018) is defined by 5-tuple (S,A,R,P,μ). Where S is the state space; A is the action space; $R : S \times A \to \mathbb{R}$ is the reward function; $P : S \times A \times S \to [0, 1]$ is the transition kernel and $\mu : S \to [0, 1]$ is the initial state distribution. Let $s_0 \sim \mu$ denote that the initial state s_0 depends on μ, then $a_t \sim \pi(\cdot \mid s_t)$ and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ are similar. This can set a simple trajectory $\tau = (s_0, a_0, s_1, \ldots)$. Consider a policy denoted by $\pi = \{\pi(a \mid s) : s \in S, a \in A\}$ and aim to find a stationary policy that maximizes the expected discounted return, i.e., objective function:

$$J_R(\pi) = \mathbb{E}^{\pi}_{s\sim\mu}[\sum_{t=0}^{\infty} \gamma^t r_t] \,,$$

where γ is the discount factor, and r_t is the reward at time t. Therefore, the update and improvement of π is based on the comprehensive judgment of each reward. If adopting the deterministic policy, $a = \pi(s)$, else the stochastic policy $a \sim \pi(a \mid s)$.

3.2 EQUALITY CONSTRAINT ACTION SPACE EXPLORATION

The method we propose is based on the dynamic knowledge. Therefore, in order to highlight the actual effectiveness of the method and the scalability applicable to any continuous-control reinforcement learning method, dynamic knowledge only affects action selection stage. We first formulate the constraints and dynamics following the notation used in Laine & Tomlin (2019). Without loss of generality, the constraint occurs at $t = 0, 1, \ldots, T-1, T$. As a matter of convenience, let $s \in \mathbb{R}^n$,

$a \in \mathbb{R}^m$, and we address the following policy problem:

$$
\begin{aligned}
& a \sim \pi(a \mid s) \\
& dynamics : s_{t+1} - (F_{s_t} s_t + F_{a_t} a_t + f_{1_t}) = 0, t = 0, 1, \ldots, T-1 \\
s.t. \quad & initial\, condition : s_0 \sim \mu \\
& constraint\, at\, t : G_{s_t} s_t + G_{a_t} a_t + g_{1_t} = 0, t = 0, 1, \ldots, T-1 \\
& constraint\, at\, T : G_{s_T} s_T + g_{1_T} = 0
\end{aligned}
$$

Where F_{s_t}, F_{a_t} and f_{1_t} define the agent dynamics at time $t = 0, 1, \ldots, T-1, T$. G_{s_t}, G_{a_t} and g_{1_t} define the constraints at $t = 0, 1, \ldots, T-1$. G_{s_T} and g_{1_T} define the constraint at $t = T$. The deterministic policy is similar. In addition, we introduce the function $C(s_t)$ called 'constraint-to-go' used in Laine & Tomlin (2019):

$$C(s_t) = H_{s_t} s_t + h_{1t}, t = 0, 1, \ldots, T \,,$$

which is similar to the value function and representing the stacking of values that the residual constraint from the beginning of s_t to the back. So at time T there is:

$$C(s_T) = G_{s_T} s_T + g_{1T} \,.$$

4 Action decomposition regular

4.1 Action decomposition

We first explain the idea of action decomposition. For a simple example, as shown in the speed coordinate system in the Fig. 2, when a constraint that requires $u_x = u_y$ occurs, we can linearly combine u_x and u_y into $w = \frac{\sqrt{2}}{2} u_x + \frac{\sqrt{2}}{2} u_y$ and $y = -\frac{\sqrt{2}}{2} u_x + \frac{\sqrt{2}}{2} u_y$, so that we only need to keep $y = 0$ to satisfy the constraint , and the w dimension will be completely free.

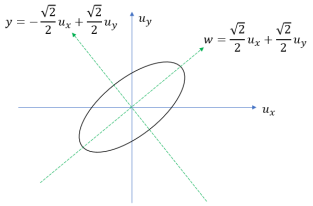


Figure 2: The original velocity coordinate system re-linearly combines two new dimensions y and w according to the constraints.

4.2 Safety regular based on action decomposition

We solve the problem of safety exploration in the action space under the linear equality constraint based on the above ideas. In our method, the solving technology of the constraint dimension matches the programming technology in Laine & Tomlin (2019), but the solution of the free dimension is expanded according to the property of the policy exploration. The solution process firstly goes backwards from $t = T - 1$:

$$
\begin{aligned}
& a \sim \pi(a \mid s) \\
& s_T - (F_{s_{T-1}} s_{T-1} + F_{a_{T-1}} a_{T-1} + f_{1_{T-1}}) = 0 \\
s.t. \quad & a \in \arg\min_a \| \begin{pmatrix} G_{s_{T-1}} s_{T-1} + G_{a_{T-1}} a_{T-1} + g_{1_{T-1}} = 0 \\ H_{s_T} s_T + h_{1_T} = 0 \end{pmatrix} \|_2
\end{aligned}
$$

and use the dynamic equation to eliminate s_T. In this way, only s_{T-1} and a_{T-1} exist in the problem. Organize the formula and rewrite the above question as:

$$
\begin{aligned}
& a \sim \pi(a \mid s) \\
s.t. \quad & a \in \arg\min_a \| N_{s_{T-1}} s_{T-1} + N_{a_{T-1}} a_{T-1} + n_{1_{T-1}} \|_2
\end{aligned}
$$

where we define as follows: $N_{s_{T-1}} = \begin{pmatrix} G_{s_{T-1}} \\ H_{s_T} F_{s_{T-1}} \end{pmatrix}$, $N_{a_{T-1}} = \begin{pmatrix} G_{a_{T-1}} \\ H_{s_T} F_{a_{T-1}} \end{pmatrix}$, $n_{1_{T-1}} = \begin{pmatrix} g_{1_{T-1}} \\ H_{s_T} f_{1_{T-1}} + h_{1_T} \end{pmatrix}$. Obviously, at this step, a of the constraint item is only related to $N_{a_{T-1}}$, that is, all the information of the constraint item is contained in $N_{a_{T-1}}$.Perform SVD on $N_{a_{T-1}}$ to get $N_{a_{T-1}} = U_{T-1}\Sigma_{T-1}V_{T-1}^T$, and define $V_{T-1}^T = \begin{pmatrix} P_{T-1}^T \\ Z_{T-1}^T \end{pmatrix}$, where the first r rows of the V_{T-1}^T are denoted as P_{T-1}^T, the last $(m-r)$ rows are denoted as Z_{T-1}^T. And r is the rank of $N_{a_{T-1}}$. Then we make use of the following result:

Corollary 4.1. *The action a formulated at time t can be decomposed in the following form:*

$$\hat{a}_t = P_t y_t + Z_t w_t$$

Proof. The proof is provided in Appendix D. □

We can regard y_t as a constraint dimension and w_t as a free dimension. Since the learning of the policy also includes the punishment feedback caused by the violation of constraints, so the original problem is transformed into:

$$\begin{aligned} w_{T-1} &= Z^T \cdot a, a \sim \pi(a \mid s) \\ y_{T-1} &= \arg\min_a \|N_{s_{T-1}} s_{T-1} + N_{a_{T-1}} P_{T-1} y_{T-1} + n_{1_{T-1}}\|_2 \\ &= -(N_{a_{T-1}} P_{T-1})^\dagger (N_{s_{T-1}} s_{T-1} + n_{1_{T-1}}) \end{aligned}$$

Through the above steps, the solution $\hat{a}_{T-1}$ will be easily obtained:

$$\hat{a}_{T-1} = P_{T-1} y_{T-1} + Z_{T-1} w_{T-1}.$$

And update $C(s_{T-1})$ by combining $\hat{a}_{T-1}$ and $(N_{s_{T-1}} s_{T-1} + N_{a_{T-1}} \hat{a}_{T-1} + n_{1_{T-1}})$:

$$\begin{gathered} C(s_{T-1}) = H_{s_{T-1}} s_{T-1} + h_{1_{T-1}} = \\ (I - N_{a_{T-1}} P_{T-1} (N_{a_{T-1}} P_{T-1})^\dagger) N_{s_{T-1}} s_{T-1} + (I - N_{a_{T-1}} P_{T-1} (N_{a_{T-1}} P_{T-1})^\dagger) n_{1_{T-1}} \end{gathered}$$

Where $(N_{a_{T-1}} P_{T-1})^\dagger$ is the pseudo inverse of $N_{a_{T-1}} P_{T-1}$. We can show that $C(s_t) = 0$, if $N_{a_{T-1}} P_{T-1}$ is an invertible matrix.

5 PRACTICAL IMPLEMENTATION

5.1 IMPLEMENTATION DETAIL

We divide the use of ADR into two types, which is shown in the Fig. 3. The first type: When the agent does not receive any constraint signals, the policy generated by the policy network directly obtains executable actions in the form of deterministic policy or stochastic policy. The second type: When the agent receives the constraint signals that it needs to comply within a period of time($t = 0, 1, \ldots, T-1, T$) in the future, we might as well start counting the time from receiving the constraint signals. We require that the constraint signals of this period of time be processed through ADR to obtain the constraint dimension action and the free dimension projection matrix, and the output action of the RL also needs to be corrected by the above result for obtaining the actual execution. In addition, we give the method to deal with situations where the selected action violates convex action space. The details are provided in Subsection 5.2.

A detailed pseudo code is provided in Appendix A of the supplementary materials.

5.2 CONVEX ACTION SPACE

Various physical limitations of action will appear in real-world applications. And physical limits can lead to the limited action space. In this section, we discuss the most common convex action space. In fact, the physical limitation is also a constraint when the selected action exceeds it. In

order to satisfy the hard constraint(Chen et al., 2021) as much as possible, we suggest that when the action exceeds the physical limitation, first program the constraint dimension in the action space to find the constraint dimension action closest to ADR's recommendation, and then find the closest free dimension action suggested by the RL. This ensures that actions get higher rewards under conditions that satisfy the constraints as much as possible. In fact, this is a multi-objective optimization problem(MOO)(Miettinen, 2012; Lin et al., 2019). The above method (also called ϵ-constraint method or main objective method) is widely used, and its optimal solution is the effective solution of MOO(also called the Pareto optimum solving) when the limited action space is a convex set. We define the following problems:

Problem 1.

$$\begin{aligned} min \quad & \begin{pmatrix} f_1(a) \\ f_2(a) \end{pmatrix} = \begin{pmatrix} \|P^T a - P^T \hat{a}\|_2 \\ \|Z^T a - Z^T \hat{a}\|_2 \end{pmatrix} \\ s.t. \quad & a \in \mathbb{D} \end{aligned}$$

Problem 2.

$$\begin{aligned} min \quad & \left(\|P^T a - P^T \hat{a}\|_2 \right) \\ s.t. \quad & a \in \mathbb{D} \end{aligned}$$

Problem 3.

$$\begin{aligned} min \quad & \left(\|Z^T a - Z^T \hat{a}\|_2 \right) \\ s.t. \quad & a \in \mathbb{H} \end{aligned}$$

where $\mathbb{H}$ is the efficient solution set of Problem. 2.

This result can be demonstrated by the following Theorem 5.1.

Theorem 5.1. *Suppose to exist $\bar{a} \in \mathbb{D}$, $\mathbb{D}$ is a convex set, subject to $\bar{a}$ is the optimal solution of Problem. 3, then $\bar{a}$ is not only the weakly effective solution of Problem. 1, but also the Pareto optimal solution of Problem. 1, and it is unique.*

Proof. See Appendix E of the supplementary materials. □

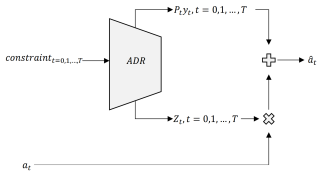
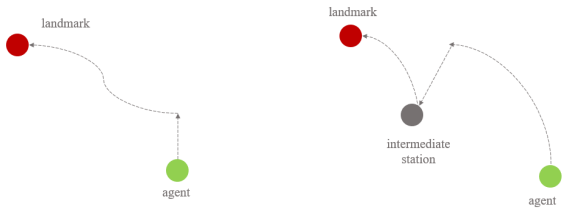


Figure 3: When the constraints in $t = 0, 1, \ldots, T$ in the future becomes effective, $\hat{a_t}$ will be corrected by action decomposition regular,otherwise $\hat{a_t} = a_t$.

6 EXPERIMENTS

Although we expect to show benefits from combining ADR with any continuous-control RL method, for the following experiments, we use the Deep Deterministic Policy Gradient (DDPG)(Lillicrap et al., 2015). Although DDPG (Lillicrap et al., 2015) is a deterministic policy that can directly output actions, in fact, our method is not only suitable for reinforcement learning algorithms for deterministic policy, but also has applicability for stochastic policy. Our experiments are based on the current popular multi-agent particle world (Lowe et al., 2017) with continuous observation and action space and some basic simulated physics. We design two new sets of simulation experiments based on physical constraints to test the effectiveness of ADR as shown in Fig 4. It is worth mentioning that no new hyperparameters are introduced in the process of our experiment.

We provide exact details about experiments in Appendix B and hyperparameters about the method in Appendix C.



(a) (b)

Figure 4: Two tasks we consider, including: (a)Keep it straight; (b)Passing the intermediate station

6.1 KEEP IT STRAIGHT

6.1.1 EXPERIMENT DESCRIPTION

The agent starts from a random starting point to a random final landmark. But we require the agent to maintain a straight line movement as accurately as possible in a certain direction during the first period of time. Although this task seems simple, it is not easy to satisfy the accuracy requirements for RL. That is because the larger learning rate of the algorithm leads the faster convergence and the poorer stability, and the smaller learning rate of the algorithm leads to slow convergence and waste of time(Smith, 2017).

In this experiment, the reward is set based on the negative Euclidean distance from the final landmark at each moment. At each step, the agent also obtains the reward for minimizing energy consumption based on the negative two-norm of action. The penalty is set based on the two-norm of the velocity deviating from the current motion direction. Finally, the violated constraint is equal to the accumulation of the two-norm of the distance from the original straight line at each time step when the constraint occurs. In fact, this will require the agent to learn to approach the landmark more quickly while keeping the direction of motion stable in the early stage.

6.1.2 EXPERIMENT ANALYSIS

Learning curves are provided in the Fig. 5. For the reward curve, DDPG needs a lot of episodes of training to obtain higher rewards, but DDPG+ADR gets higher rewards at the beginning and is always higher than DDPG in the whole training process. For the violated constraint curve, DDPG seriously violates the constraints at the beginning of training, and can not strictly satisfy the constraints in the whole training process. In fact, the minimum value of constraint violation in a single round of DDPG is 7.4×10^{-8}. But DDPG+ADR can keep the violation of constraints in the order of 10^{-16} in the whole process, which can be considered negligible.

The experiments show that, on the one hand, DDPG+ADR can indeed make the actions output by RL's policy strictly satsify the linear equality constraints, even in the training process. On the other hand, compared with DDPG, DDPG+ADR shows better performance in obtaining rewards.

6.2 PASSING THE INTERMEDIATE STATION

6.2.1 EXPERIMENT DESCRIPTION

The agent is still required to go from a random starting point to a random final landmark. And the agent will suddenly receive a constraint signal to go to an intermediate station at the intermediate moment. Note that since the agent is constrained only at the intermediate moment, the agent will exceed its physical limitations due to the distance of the intermediate station, which is too far away. In this case, the agent can only approach as close as possible and never satisfy the constraint. In fact, this experiment requires the algorithm to be robust when the agent encounters a sudden constraint that exceeds its physical limit.

In this experiment, the reward is set based on the negative Euclidean distance from the final landmark at each moment. At the same time, the agent also obtains the negative two-norm of action as the

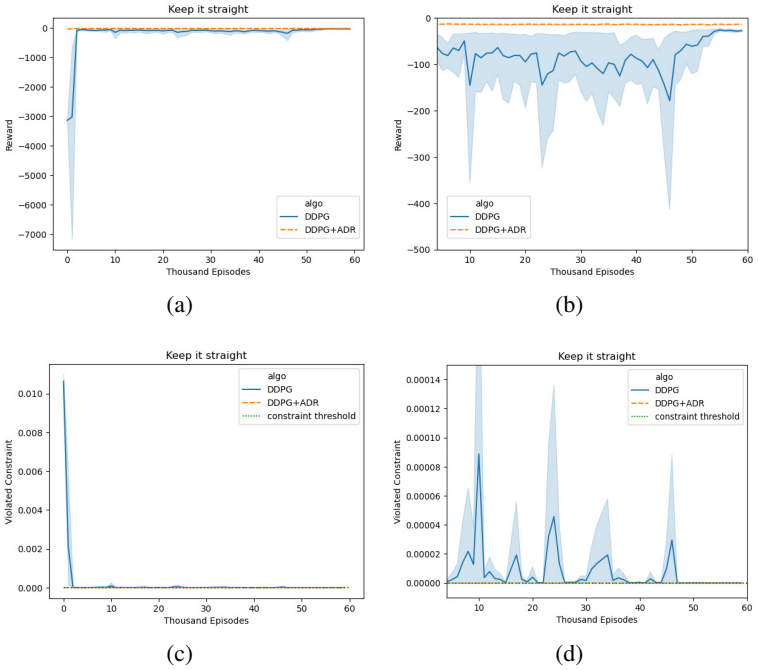


Figure 5: Reward refers to the mean of accumulated return in a certain number of episodes. Violated constraints refers to the accumulation of the agent's violation of the constraint during the time when the constraint occurs. (a)The reward curve from starting to train until it converges. (b)The part of (a) after 4000 episodes. (c)The violated constraint curve. (d)The part of (c) after 4000 episodes. The green dotted line equal to 0 indicates the accumulation of 0 violated constraint. Plotted are medians with upper and lower quantiles of 5 seeds.

reward for minimizing energy consumption. The penalty for the agent receives and the violated constraint in each episode are set based on the Euclidean distance from the intermediate station.

6.2.2 Reward Shaping

For comparison, we also conduct reward shaping experiments on the DDPG algorithm. At each time step before the end of the constraint, we set the modified reward function(Ng et al., 1999) to the same scale as the original reward, which is set by the following formula:

$$r_F = \phi(s_t) - \phi_(s_{t-1}), \phi_(s_0) = 0$$

Where ϕ is set based on the distance from the intermediate station, see Appendix B for details.

6.2.3 Experiment Analysis

The experimental results are shown in the Fig. 6. Compared with DDPG, DDPG+ADR has demonstrated superior performance, not only in terms of cumulative rewards much higher, but also much smaller in violation of constraints. Surprisingly, the design of reward shaping does not make DDPG run better but have an adverse effect. It means that the value function of this task is complicated, and the reward shaping that only relies on constraints is quite different from the value function.

This shows that at the moment when the constraint occurs, DDPG+ADR really shows robustness. It helps the agent make the action that satisfies the constraint as much as possible and minimizes the missed reward.

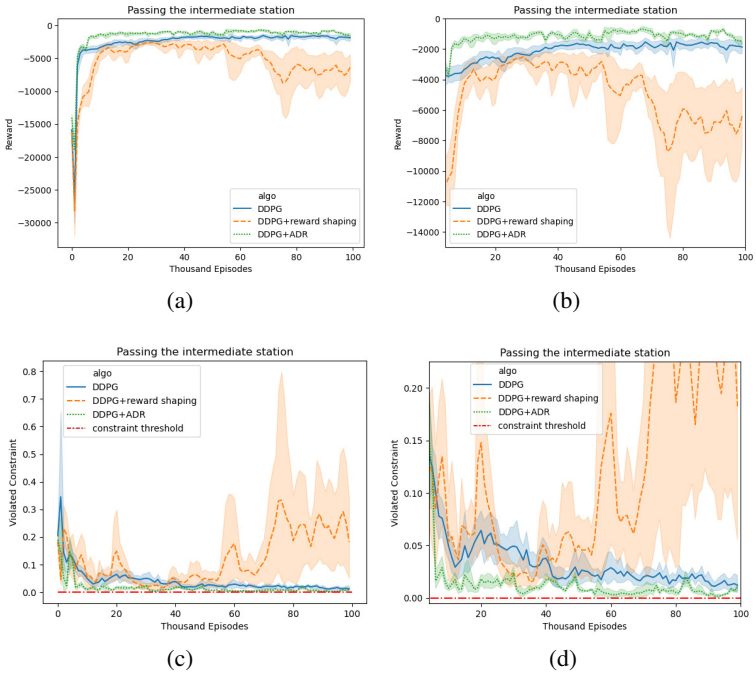


Figure 6: (a)The learning curve of the reward in this task. (b)The part of (a) after 10000 episodes. (c)The violated constraint curve. (d)The part of (c) after 10000 episodes. The green dotted line equal to 0 indicates the accumulation of 0 violated constraint. Plotted are medians with upper and lower quantiles of 6 seeds.

7 Discussion

In this paper, we propose a simple and practical approach that can effectively solve the problem of action exploration in reinforcement learning under the linear equality constraints. Our method ADR is based on the linear dynamics model and uses the idea of SVD to decompose the action space into constrained dimension and free dimension to control separately. At the same time, we propose feasible solutions to the situation that constraints exceed convex action space, and ensure that actions satisfy the constraints as much as possible within a single time step, and the loss of rewards can be minimized. In the experiment, compared with DDPG, DDPG+ADR can obtain more rewards and stricter constraints satisfaction in both tasks. At the same time, DDPG+ADR shows its robustness in sudden constrained tasks. It is worth mentioning that our method has the advantages of no training and does not need to make assumptions about the dimensions of constraints.

An exciting feature is that our method can be combined with any continuous-control RL method. In addition, there are many promising ideas for future work: the use of interior point methods to improve the equality constraints; the deeper integration of SVD ideas with reinforcement learning(Gemp et al., 2020). And in the real world, some dynamic models are too complicated to be researched. In future work, we plan to use Piecewise Linear Neural Networks(PLNN) which can explain the non-linear dynamic model of an object(Nagabandi et al., 2018; Chu et al., 2018) to extend the applicability of our method.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. *arXiv preprint arXiv:1705.10528*, 2017.

Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pp. 136–145. PMLR, 2017.

Anil Aswani, Patrick Bouffard, and Claire Tomlin. Extensions of learning-based model predictive control for real-time application to a quadrotor helicopter. In *2012 American Control Conference (ACC)*, pp. 4661–4666, 2012. doi: 10.1109/ACC.2012.6315483.

Anil Aswani, Humberto Gonzalez, S Shankar Sastry, and Claire Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

Guillaume Bellegarda and Katie Byl. An online training method for augmenting mpc with deep reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5453–5459, 2020.

Felix Berkenkamp, Riccardo Moriconi, Angela P Schoellig, and Andreas Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 4661–4666. IEEE, 2016.

Felix Berkenkamp, Matteo Turchetta, Angela P Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *arXiv preprint arXiv:1705.08551*, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Yuntian Chen, Dou Huang, Dongxiao Zhang, Junsheng Zeng, Nanzhe Wang, Haoran Zhang, and Jinyue Yan. Theory-guided hard constraint projection (hcp): A knowledge-based data-driven scientific machine learning method. *Journal of Computational Physics*, 445:110624, 2021. ISSN 0021-9991.

Lingyang Chu, Xia Hu, Juhua Hu, Lanjun Wang, and Jian Pei. Exact and consistent interpretation for piecewise linear neural networks: A closed form solution. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1244–1253, 2018.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Jaime F. Fisac, Anayo K. Akametalu, Melanie N. Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2019.

Javier García, Fern, and o Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015.

Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. Eigengame: Pca as a nash equilibrium. *arXiv preprint arXiv:2010.00554*, 2020.

Markus Giftthaler, Michael Neunert, Markus Stäuble, Jonas Buchli, and Moritz Diehl. A family of iterative gauss-newton shooting methods for nonlinear optimal control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9, 2018.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. 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.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Computer ence*, 2015.

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

L.C.M. Kallenberg. *Linear programming and finite Markovian control problems*. MC Tracts. Centrum Voor Wiskunde en Informatica, 1983.

Forrest Laine and Claire Tomlin. Efficient computation of feedback control for equality-constrained LQR. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6748–6754. IEEE, 2019.

Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pp. 3703–3712. PMLR, 2019.

Sergey Levine and Vladlen Koltun. Guided policy search. In *International conference on machine learning*, pp. 1–9. PMLR, 2013.

Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Sören Kammel, J. Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, Michael Sokolsky, Ganymed Stanek, David Stavens, Alex Teichman, Moritz Werling, and Sebastian Thrun. Towards fully autonomous driving: Systems and algorithms. pp. 163 – 168, 07 2011.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *Computer ence*, 2015.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. *Advances in neural information processing systems*, 32:12060–12070, 2019.

Ryan Lowe, Yi Wu, Aviv Tamar, and Jean Harb. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

Michael Lutter, Boris Belousov, Kim Listmann, Debora Clever, and Jan Peters. Hjb optimal feedback control with deep differential value functions and action constraints. In *Conference on Robot Learning*, pp. 640–650. PMLR, 2020.

Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

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

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566. IEEE, 2018.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287, 1999.

Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, volume 98, pp. 463–471. Citeseer, 1998.

Keith W Ross and Ravi Varadarajan. Markov decision processes with sample path constraints: the communicating case. *Operations Research*, 37(5):780–790, 1989.

Keith Wimberly Ross. *Constrained Markov decision processes with queueing applications*. PhD thesis, University of Michigan, 1985.

Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. In *In AISTATS*. Citeseer, 2011.

Tedrake Russ. Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation(course notes for mit 6.832). `http://underactuated.mit.edu/`, 2021.

S. Sadraddini and C. Belta. A provably correct mpc approach to safety control of urban traffic networks. In *2016 American Control Conference (ACC)*, 2016.

Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

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

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

L. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

Mark W Spong. The swing up control problem for the acrobot. *IEEE control systems magazine*, 15 (1):49–55, 1995.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pp. 9133–9143. PMLR, 2020.

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

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2019.

Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

Chao Yu, Jiming Liu, and Shamim Nemati. Reinforcement learning in healthcare: A survey. *arXiv preprint arXiv:1908.08796*, 2019.

Yiming Zhang, Quan Vuong, and Keith W Ross. First order constrained optimization in policy space. *arXiv preprint arXiv:2002.06506*, 2020.

A PSEUDO CODE

Algorithm 1: Action Decomposition Regular

Input: constraint$G_{s_t}, G_{a_t}, g_{1_t}, G_{s_T}, g_{1_T}$; policy network π_θ ; dynamics $F_{s_t}, F_{a_t}, f_{1_t}$; $t = 0, 1, \ldots, T-1$

Output: action $a_t; t = 0, 1, \ldots, T-1$

if $T > 0$ **then**
  $H_{s_T} \leftarrow G_{s_T}$
  $h_{s_T} \leftarrow g_{s_T}$
  for $t = T-1, T-2, \ldots, 0$ **do**
    $N_{a_t} \leftarrow \begin{pmatrix} G_{a_t} \\ H_{s_{t+1}} F_{a_t} \end{pmatrix}$
    $N_{s_t} \leftarrow \begin{pmatrix} G_{s_t} \\ H_{s_{t+1}} F_{s_t} \end{pmatrix}$
    $n_{1_t} \leftarrow \begin{pmatrix} g_{1_t} \\ H_{s_{t+1}} f_{1_t} + h_{1_{t+1}} \end{pmatrix}$
    $V_t^T \leftarrow \text{SVD}(N_{a_t})$
    $P_t, Z_t \leftarrow V_t^T$
    $H_{s_t} \leftarrow (I - N_{a_t} P_t (N_{a_t} P_t)^\dagger) N_{s_t}$
    $h_{1_t} \leftarrow (I - N_{a_t} P_t (N_{a_t} P_t)^\dagger) n_{1_t}$
  end for
end if
if $T > 0$ **then**
  for $t = 0, 1, \ldots, T-1$ **do**
    $a_t \leftarrow \pi_\theta$
    Receive s_t
    $y_t \leftarrow -(N_{a_t} P_t)^\dagger (N_{s_t} s_t + n_{1_t})$
    $w_t \leftarrow Z_t^T a_t$
    $a_t \leftarrow P_t y_t + Z_t w_t$
  end for
else
  $a_t \leftarrow \pi_\theta$
end if

B EXPERIMENT DETAILS

All the experiments we conducted are built on Python(3.6) and Tensorflow (1.8.0) in Intel i7-10875H CPU.

B.1 KEEP IT STRAIGHT

We used the multi-agent particle environment (Mordatch & Abbeel, 2017) provided by OpenAI Gym(Brockman et al., 2016) for this set of tasks. The agent moves on a two-dimensional plane and travels from a random starting point to a random goal point. At the beginning of each episode, we require the agent to accurately move in a straight line in the y-axis direction, similar to walking out of a parking space or crossing a narrow road. In our setting, the step length of an episode is 26 steps, so the duration of this straight-going phase should not be too long, and our setting is 5 steps. For the reward of the agent in the experiment, we set the following:

1. **Reward for the agent to go to the goal:**

$$r_{goal} = -\|p_{agent} - p_{goal}\|_2^2$$

2. **Reward for the agent to keep moving in a straight line:**

$$r_{keep} = -10000|v_y|^2$$

3. **Reward for the agent about control Effort Penalty:**

$$r_{control} = -0.01\|a\|_2^2$$

Usually when we face such a multi-objective optimization problem(MOO), we always impose a large weight on the hard constraint. In order to let DDPG and DDPG+ADR learn to keep the straight line as hard as possible, we both set the weight to 10000. This has no effect on the comparison of our method ADR.

where p_{agent}, p_{goal} are the positions of the agent and goal point. And v_y is the velocity of the agent in y-axis.

And the constraint setting is:

$$constraint : v_y = 0$$

In the multi-agent particle environment (Mordatch & Abbeel, 2017), $a \in \mathbb{R}^5$ represents the join forces of the agent, and $s \in \mathbb{R}^4$ is composed of the speed of the agent and the distance to the goal point. Regardless of noise, and let the mass m of the agent be 1, we fully follow the dynamic equation in multi-agent particle environment(Mordatch & Abbeel, 2017):

$$\begin{aligned} v &= \tfrac{A}{m} a dt + (1-d)v \\ x &= vdt + x \\ A &= \begin{pmatrix} 0, 1, -1, 0, 0 \\ 0, 0, 0, 1, -1 \end{pmatrix} \end{aligned}$$

where A is the matrix that turn the resultant force into the driving force of agent. And $dt = 0.1$ is the step size of a single time step. The physical damping coefficient $d = 0.25$.

B.2 PASSING THE INTERMEDIATE STATION

The agent is also on a two-dimensional plane, going from a random starting point to a random goal point. But unlike before, there is an intermediate station at a distance of $(0.3, 0.3)$ from the goal point. The agent will receive the constraint of going to the intermediate station as much as possible in the middle moment. The time step of each episode is also 26 steps, this time we chose the intermediate time $t = 12$. And it only takes effect at this moment. This requires the agent to learn to take corresponding actions in emergency situations. We set the agent's reward in this task as follows:

1. **Reward for the agent to go to the goal:**

$$r_{goal} = -\|p_{agent} - p_{goal}\|_2^2$$

2. **Reward for the agent to move towards the intermediate station:**

$$r_{pass} = -10000\|(0.3, 0.3) - (p_{agent} - p_{goal})\|_2^2$$

3. **Reward for the agent about control Effort Penalty:**

$$r_{control} = -0.01\|a\|_2^2$$

Similarly, in order for the policy in DDPG and DDPG+ADR to learn to satisfy the hard constraints as much as possible, we set a larger weight for the second reward.

And the constraint setting is:

$$constraint : p_{agent} - p_{goal} = (0.3, 0.3)$$

As for the dynamic equation, it is exactly the same as the task setting above. And in reward shaping, r_{pass} is modified to r_{Ft}. The effective time of r_{F_t} is modified from t=1 to t=12. The formula of r_F is modified to:

$$r_{F_t} = r_{pass_t} - r_{pass_{t-1}}, t = 1, 2 \ldots, 12,$$

where r_{pass_t} means that the argument of the function r_{pass} is the current state, and the argument of $r_{pass_{t-1}}$ is the state at the previous moment. And $r_{pass_0} = \phi(s_0) = 0$ (Ng et al., 1999).

Table 2: Hyperparameters for experiments

Hyperparameter	**DDPG+ADR**
maximum episode length	26
learning rate for Adam optimizer	10^{-2}
discount factor	0.95
batch size	1024
number of units in the MLP	64
number of hidden layers	2
size of the replay buffer	10^6

C HYPERPARAMETERS FOR EXPERIMENTS

The hyperparameter settings of DDPG and DDPG+ADR are exactly the same, and there are no additional parameters introduced. And in fact, there is no need to adjust the parameters in our experiment. Activation function for MLP is ReLU. Table 2 shows the hyperparameters used in the experiment.

D PROOF OF COROLLARY 4.1

Proof. Since V is composed of normal orthogonal basis, then we have $\hat{a}_t = V_t \cdot V_t^T \cdot \hat{a}_t$, where $V_t^T = \begin{pmatrix} P_t^T \\ Z_t^T \end{pmatrix}$. We can therefore derive

$$\hat{a}_t = \begin{pmatrix} P_t, Z_t \end{pmatrix} \cdot V_t^T \cdot \hat{a}_t = P_t y_t + Z_t w_t.$$

□

E PROOF OF THEOREM 5.1

Proof. Since the objective functions of Problem. 2 and Problem. 3 are both convex functions, $\mathbb{D}$ is a convex set, and the local minimum of the convex function is the global minimum, so Problem. 2 and Problem. 3 always have optimal solutions.

If $\bar{a}$ is not the Pareto optimal solution of Problem. 1, then $\exists a \in \mathbb{D}$, which satisfies one of the following two cases: either $f_1(a) \leq f_1(\bar{a})$ and $f_2(a) < f_2(\bar{a})$, or $f_1(a) < f_1(\bar{a})$ and $f_2(a) \leq f_2(\bar{a})$. But the first case contradicts Problem. 3, and the second case contradicts Problem. 2.

Uniqueness is obvious, because $V^T = \begin{pmatrix} P^T \\ Z^T \end{pmatrix}$ constitutes a set of Orthonormal basis in the action space. □