OSSP-PTA: An Online Stochastic Stepping Policy for PTA on Reinforcement Learning

Dan Niu, *Member, IEEE,* Yichao Dong, *Graduate Student Member, IEEE,* Zhou Jin, *Member, IEEE,* Chuan Zhang, *Senior Member, IEEE,* Qi Li, *Member, IEEE,* Changyin Sun, *Senior Member, IEEE,*

***Abstract*—DC analysis is essential and still quite challenging in large-scale nonlinear circuit simulation. Pseudo transient analysis (PTA) is a widely-used and has great potential solver in the industry. However, the PTA convergence and simulation efficiency is still seriously affected by its stepping policy. This paper proposes an online stochastic stepping policy (OSSP) for PTA based on deep reinforcement learning (DRL). To achieve better policy evaluation and stronger stepping exploration ability, the dual soft Actor-Critic agents work with the proposed valuation splitting and online momental scaling, enabling our OSSP to intelligently encode PTA iteration status and online further adjust forward and backward time-step size for unseen test circuits without human intervention and domain knowledge, trained solely by RL from self-search. Our public sample buffer and priority sampling are also introduced to overcome the sparsity and imbalance of sample data. Numerical examples demonstrate that the proposed OSSP achieves a significant efficiency speedup (up to 47.0X less NR iterations) and convergence enhancement on unseen test circuits compared with the previous iter-based and SER-based stepping methods, in just one stepping iteration.**

***Index Terms*—DC analysis, pseudo transient analysis, stochastic stepping, valuation splitting, momental scaling**

I. Introduction

In circuit simulation, robust computation of DC operating point of large-scale nonlinear circuit is still a fundamental and difficult task [1], [2]. DC analysis is not only a first basic check of circuit operating, but also a precondition with influential repercussions for subsequent analyses in SPICE-like transistor-level simulators, which include small-signal AC analysis, transient analysis and sensitivity analysis [3], [4]. In essence, DC analysis is to solve a large set of nonlinear algebraic equations established by modified nodal analysis (MNA) [1], [5], [6].

Early circuit simulators, which relied mainly on the Newton-Raphson (NR) method or one of its variants to calculate the operating point, are considered unreliable [1], [7]. These methods are robust and quadratically convergent if supplying a starting point sufficiently close to a solution. But NR-based methods may fail if no such point is selected [8], [9]. To solve the convergence issue, a variety of continuation methods are further proposed, including Gmin stepping methods [10], [11], Source stepping methods [12], [13], Homotopy methods [14], [15], Pseudo-transient analysis (PTA) [16]–[19]. Among these, Gmin stepping and Source stepping methods are relatively simple and mature. Homotopy methods are highly device model-dependent and difficult to be implemented in actual simulators, though they are globally convergent [13], [20]. During the homotopy implementation, a complex modification of the SPICE source program is required.

In contrast, PTA and its variants (e.g. Pure PTA (PPTA) [21], [22], Damped PTA (DPTA) [23], [24] and Compound element PTA (CEPTA) [25]) do not depend on the device model. They have been used as the most practical and dominant solver in the industry due to no discontinuity issues and ease of implementation [26], [27]. The idea in PTA is to first modify the circuit network by adding some pseudo elements, in such a way that a prespecified state x_0 can become a valid initial state. Then a transient analysis is carried out, starting from x_0, until reaching a steady state [25]. In this case, DC analysis is simplified to work out the steady state of a system of ordinary differential equations (ODEs), which can be solved iteratively through numerical integration stepping.

Obviously, the stepping policy is quite important for PTA convergence and simulation efficiency, since it determines the number and the difficulty level of nonlinear equations to be solved at discrete timepoints. However, choosing the stepping policy is nontrival and still the bottleneck in the PTA application [20]. Currently some heuristic stepping control methods have been proposed [28], [29]. But these stepping techniques rely on heuristics and manually-setting formulas that are too general and do not consider the specificities of each circuit netlist. In addition, the enhancement of device nonlinearity and exponential increase of the parasitic parameters pose bigger challenges for the PTA convergence. In this situation, it has emerged as a promising and hot research topic that whether and how to design a "intelligent" and "adaptive" PTA stepping policy by blending machine learning (ML) methods [4], [30].

Reinforcement learning (RL), as an important branch of machine learning, has resulted in the creation of self-learning agents achieving superhuman performance at the games of Go, Shogi and Chess [31], [32]. In electronic design automation (EDA) field, deep RL is also used with large success in many

Manuscript received July 29, 2022; revised October 16, 2022. This work was supported by the Natural Science Foundation of Jiangsu Province of China under Grant BK20202006, the Commercialization of Scientific and Research findings of Jiangsu Province under Grant BA2021012. (*Corresponding author: Zhou Jin and Dan Niu.*)

Dan Niu are with the Key Laboratory of Measurement and Control of CSE, Ministry of Education, Nanjing 210096, China.

Dan Niu, Yichao Dong and Qi Li are with the School of Automation, Southeast University, Nanjing 210096, China.

Changyin Sun is with the School of Artificial Intelligence, Anhui University, Hefei 230601, China.

Zhou Jin is with the Super Scientific Software Laboratory, China University of Petroleum-Beijing, Beijing 102249, China.

Chuan Zhang is with the LEADS of Southeast University, the National Mobile Communications Research Laboratory of Southeast University, and the Purple Mountain Laboratories, Nanjing 211189, China.

applications such as chip placement, logic synthesis and device sizing automation [33]–[35]. In the PTA for DC analysis, the optimal step size at each PTA time-step is unknown and thus the supervised learning is not applicable. But the circuit simulation states at current PTA time-step can provide a wealth of effective information to determine the step size for the next PTA time-step iteration. The PTA iteration process can be approximately treated as a Markov decision process that RL owns a great benefit to solve [30], [36].

In this paper, dual autonomous RL agents working seamlessly with valuation splitting and online momental scaling for online stochastic PTA stepping policy (OSSP-PTA) are proposed to adaptively tune the PTA forward stepping and backward stepping, which will accelerate PTA convergence remarkably without human intervention and domain knowledge, trained solely by RL from self-search. Moreover, for the unseen test circuit simulations, the proposed method can still continuously learn and online further adjust the stepping policy compared with the supervised learning with fixed network. The contributions of this paper are as follows.

1) We achieves a significant efficiency speedup and convergence enhancement for DC analysis on unseen test circuits by applying the proposed online stochastic stepping policy (OSSP) for PTA methods, without any additional training and in just one PTA iteration process. It intelligently encodes PTA iteration status and online adaptively tune forward and backward time-step size without supervised samples.

2) We introduce dual soft Actor-Critic agents and propose an online stochastic stepping policy (OSSP) based on maximum entropy. Compared with the deterministic stepping policy in the previous PTAs, OSSP achieves stronger stepping space exploration ability and helps damp out the oscillations to obtain better convergence.

3) Continuous valuation splitting architecture with three streams is proposed to produce separate estimates of the state value function and advantage function. It can achieve better policy evaluation in the presence of redundant or similar actions to accelerate simulation.

4) An adaptive and momental step scaling along with online stepping policy update are put forward to deal with the circuit differences between offline training and online prediction. Biased first and second moment estimates are introduced to obtain online momental step scaling by encoding the previous step sizes of the unseen test circuit. It breaks through the gain limitations to further improve the simulation efficiency.

In addition, the public sample buffer and priority sampling are introduced to solve insufficient and imbalance sample data.

The proposed OSSP method is compatible to almost all kinds of PTA solvers and easy to be implemented in the SPICE-like simulators.

II. Preliminary

In this section, PTA as well as its typical time-step control methods are reviewed.

A. PTA Method

Consider a nonlinear circuit to be solved. Let the number of nodes in a circuit be N+1, the number of independent voltage sources be M. Such a circuit can be described by modified nodal equations to obtain nonlinear algebraic equations [37], that is

$$\mathbf{F}(\boldsymbol{x})=0, \tag{1}$$

where $\boldsymbol{x}=(\boldsymbol{v},\boldsymbol{i})^T\in\boldsymbol{R}^n$, and $n=N+M$. The variable vector $\boldsymbol{v}\in\boldsymbol{R}^N$ denotes the node voltages to the datum node and the variable vector $\boldsymbol{i}\in\boldsymbol{R}^M$ represents the branch currents of the independent voltage sources [38]. When the target circuit is modified to the PTA case, the set of ordinary differential equations (ODEs) can be obtained as

$$\left\{\begin{array}{l}\mathbf{D}\dot{\boldsymbol{x}}(t)=-\mathbf{F}(\boldsymbol{x}(t),t),\\ \boldsymbol{x}\left(t_0\right)=\boldsymbol{x}_0,\end{array}\right. \tag{2}$$

where $\dot{\boldsymbol{x}}(t)=(\dot{\boldsymbol{v}}(t),\dot{\boldsymbol{i}}(t))$ is the derivative of $\boldsymbol{x}$ with respect to the time t, and $\mathbf{D}$ is the incidence matrix that represents the inserted pseudo elements [25]. Then the implicit DDF-k numerical integration [20], shown in Eq. (3), is employed to Eq. (2) at the discrete time point t_{n+1}, where n≥k-1.

$$\left.\dot{\boldsymbol{x}}(t)\right|_{t=t_{n+1}}=\frac{\boldsymbol{x}_{n+1}-\boldsymbol{x}_{n+1-k}}{\sum\limits_{j=0}^{k-1}\left(\boldsymbol{h}_{n+1-j}\right)}. \tag{3}$$

By Eq. (3), Eq. (2) is numerically integrated to a steady state using some variable time-step schemes. The time interval $[0,t_f]$ is replaced with discrete time points $t_0,t_1,\ldots,t_n,t_{n+1},\ldots,t_f$ through numerical integration, and the time-step size is $h_n\equiv t_n-t_{n-1}$. t_f is the time point when the pseudo circuit is settled down. Obviously, a suitable stepping policy is essential for whether and how fast the PTA iterations reach steady state.

B. Time-Step Control Methods

In the commercial EDA tools, a simple NR iteration counting stepping method called **"iter-based method"** here, is widely utilized in PTA methods [28], [29]. It compares the number of NR iterations at each PTA time-step with two options (IMAX and IMIN) to determine next time-step size. Generally, when the number of NR iterations is less than IMIN, the next step size will be enlarged by 2 times. If the number is more than IMAX, the simulator rolls back to the previous time-step and the new size will be set as 1/8 of previous size. Otherwise, the step size keeps the same. This policy is simple and fast. However, it only considers the NR iteration number and employs the fixed size change rates, which are too rigid to achieve good simulation performance. Moreover, selecting approximate parameters (e.g. IMAX, IMIN, initial time-step size) for different circuits are also difficult [20].

Moreover, a adaptive time-step control method based on switched evolution/relaxation (SER) was proposed in [39] (called **"SER-based method"** here). In this method, the residual $\mathbf{F}(x)$, the relative change of $\mathbf{x}$ and the iteration count Nitr_n are considered to optimize the time-step size. The forward stepping algorithm, in its simplest, is

$$\begin{aligned}h_{n+1}&=E\left(h_n,\mathrm{Nitr}_n,\mathbf{x},\mathbf{F}(\mathbf{x})\right)\\&=h_n\cdot MAX\left(1,\delta\cdot\gamma\cdot\left\|\mathbf{F}\left(\mathbf{x}_{n-1}\right)\right\|/\left\|\mathbf{F}\left(\mathbf{x}_n\right)\right\|\right),\end{aligned} \tag{4}$$

where δ denotes the relative change of x per time-step, and γ assesses the difficulty of NR convergence, defined as $\mathrm{IMIN}\,/\,\mathrm{Nitr}_n$.

In the backward stepping, cautious strategy is adopted to reduce the time-step size gradually, which is expected to improve the backward stepping efficiency. The parameter G (usually initialized to 10) is set to control the decrement ratio of step size, that is

$$h_{n+1} = \frac{G}{1+G} h_n, \tag{5}$$

It is a manually set formula. G will be reduced by half for each roll-back step.

While larger backward step size can be obtained, the conservative backward strategy reduces the step size too slow and usually continuous rollbacks are required to achieve NR convergence again, which will waste too much NR iterations. Besides, It is also difficult to optimize parameters for various circuits. It is still a heuristic method relying on artificial expertise and has weak generalization ability. However, it has been demonstrated that intelligent and flexible time-step control methods have great potential to enhance PTA convergence and accelerate simulation efficiency.

III. Proposed Online Stochastic Stepping policy

A. Overview

In this section, the PTA iterative solution tracing process is treated as a MDP problem and an online stochastic stepping policy is proposed to interact with the circuit simulation environment. We build dual RL agents that tune the forward and backward step size of PTA iterations autonomously, with the objective of minimizing total NR iterations while achieving PTA convergence. Our RL-based stepping problem consists of the following four key elements:

States: the set of PTA iteration state parameters from the EDA tools. A single state s_t consists of a state parameter set at one time-step t_n.

Actions: the set of actions that the agent can use to compute the PTA iteration solution of pseudo circuit. An action a_t is to generate proper next time-step size h_{n+1} for transient analysis.

State transition: given a state s_t and an action a_t, the next state is the iteration state parameters of the same pseudo circuit with next time-step size h_{n+1}.

Reward: the reward increases if the action generates the time-step size to make PTA iterations tend to converge.

In RL, our two agents learns from interacting with transient iteration of pseudo circuit over a number of discrete time-steps. At each time-step t_n, the agents receive a state s_t, and output a normalized action a_t ((-1,1) range) according to its policy π. In return, the agents receive a reward signal r_t and transition to the next state s_{t+1}. An optimal policy is one that maximizes the expected returns or values [31].

B. Our RL stepping settings

In this work, PTA stepping policy is a combinatorial optimization problem where state set is very large and exhaustive search is infeasible. For the agents adaptively selecting a proper action, a good representation of PTA iteration status must be firstly defined.

Our States The selection of RL states is quite important to transfer the knowledge across very different circuit netlists so that our agents generalize stepping tuning policy to unseen circuit netlists. It should reflect the PTA iteration status and evaluate whether and how difficult the iteration tends toward convergence. Considering that circuit topologies and circuit scales vary, node voltages or branch currents and even their normalized values are not appropriate and are not considered as states. In this work, our state s_t is written as a concatenation of the following parameters **NR_iters**, **Res** and **RC_rate**.

- **NR_iters:** is the NR iteration number to get the transient solution at each time-step t_n and it indirectly translates the difficulty of NR iterations.
- **Res:** is the residual, defined as

$$Res = C_R \frac{\|\boldsymbol{x}_n - \boldsymbol{x}_{n-1}\|}{\|t_n - t_{n-1}\|}, \tag{6}$$

where C_R is the residual coefficient. Res evaluates whether the equations are close to final solution.

- **RC_rate:** is the relative change rate of solutions, which is given by

$$RC\_rate = C_{RC} \frac{\|\boldsymbol{x}_n - \boldsymbol{x}_{n-1}\|}{\|\boldsymbol{x}_{n-1} - \boldsymbol{x}_{n-2}\|}, (n \geq 2), \tag{7}$$

where C_{RC} is the coefficient of relative change rate. RC_rate is served to indicate whether the iteration solution tends toward the steady state or still changes drastically.

In addition, two flags (Conv_NR and Conv_PTA) are used to represent the convergence status and will be used in reward functions. **Conv_NR** denotes whether the NR iterations converge or not. It is obtained by comparing the NR iteration number with a constant parameter. It is important to determine whether backward stepping works. **Conv_PTA** as a successful ending flag indicates whether the PTA iterations reach the steady state.

The fore-mentioned state inputs are normalized and encodered by our agents. As shown in Fig. 1, two actor networks in the proposed dual Actor-Critic agents will online adaptively output an stochastic action a to generate the next time-step size h_{n+1} for forward stepping or h^b_{n+1} for backward stepping.

Our Dual Agents As shown Fig. 1, our dual agents include forward agent and backward agent. According to the input state s_n at t_n timepoint, the dual agent network will firstly determine which agent (Forward or Backward) works. Then the Actor module in Forward agent or Backward agent outputs the stochastic action a_n, which is used to generate the next step size for PTA iterations. For DC analysis by PTA iterations, conventional single agent structure of RL is unsuitable due to two main reasons. First, not only forward stepping operations but also backward stepping operations may occur in circuit simulation, especially solving DC analysis of "difficult" circuits. When NR iterations converge, forward stepping operation is expected to take large enough step size to enhance simulation efficiency. Otherwise, backward stepping operation rollbacks to the previous time-step and a smaller step size

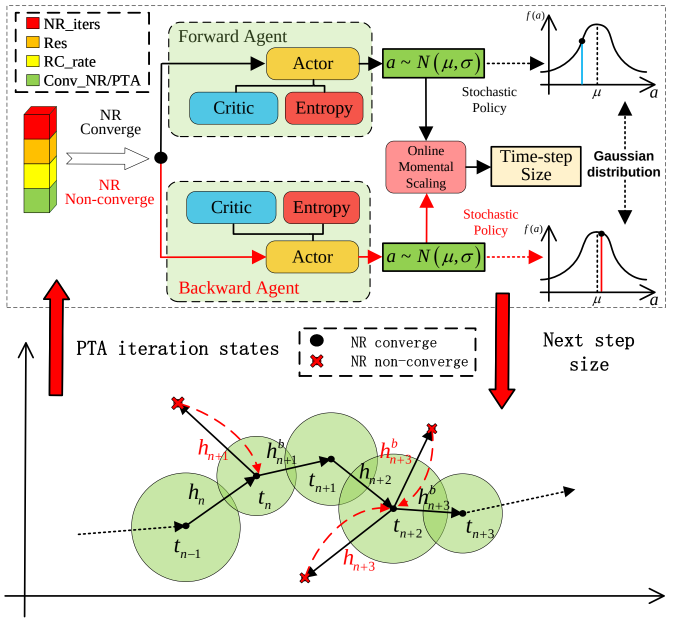

Fig. 1. Proposed dual soft Actor-Critic agents with stochastic stepping policy

should be chosen carefully to solve NR non-convergence. This differs from many existing RL game tasks where "Undo" does not exist. Besides, when NR non-convergence occurs, solution $\boldsymbol{x}_{n+1}$ at time-step t_{n+1} is non-existent and the states (e.g. Res and RC_rate) for RL can not be obtained, which is also different from many RL tasks. Secondly, the training samples at forward steppings are much more than those at backward steppings, which results in sample imbalance. In this case, it is hard for single agent structure to learn valuable backward stepping information and the network is easily unstable on account of the imbalanced samples. This forms an obstacle for actor network online update. In this work, dual agents called forward agent and backward agent, are introduced to cope with the NR convergence and non-convergence situation separately and output the optimal stepping policy.

Online Stochastic Stepping Policy As shown in Fig. 1, the proposed online stochastic stepping policy will give the next step size by sampling from a probability distribution. An online stochastic stepping policy rather than a deterministic stepping policy [28], [30], [39] in previous PTAs is proposed to realize stronger stepping space exploration ability, which is helpful to damp out the oscillation to obtain better PTA convergence performance. Moreover, the stochastic stepping policy is also be online updated to deal with the circuit differences between offline training and online prediction. The output action a is a Gaussian distribution whose mean and covariance is obtained by the Actor networks, where the improved soft Actor-Critic [40] with valuation splitting is proposed (shown in subsection C). A soft Q value function $Q_\theta(s_t, a_t)$, and a tractable policy $\pi_\phi(a_t|s_t)$ is employed. Three streams are proposed to explicitly separate the representation of state values and (state-dependent) action advantages. They are combined to produce the estimate of the soft Q value function, which can effectively stabilize modal training.

The soft Q value function is trained by minimizing the soft Bellman residual. It is convenient to train simultaneously with the policy networks. The update of Q value network is:

$$\nabla_\theta J_Q(\theta) = \nabla_\theta \frac{1}{|B|} \sum_{(s,a,r,s)\in B} \left(Q_\theta(s,a) - y\left(r,s'\right)\right)^2, \tag{8}$$

where B is the set of samples whose quantity is batch-size. $y\left(r,s'\right)$ is defined as:

$$y\left(r,s'\right) = r + \xi\left(Q_\theta\left(s',a'\right) - \kappa \log \pi_\theta\left(a' \mid s'\right)\right), \tag{9}$$

where ξ is the decay factor, κ is the entropy temperature factor. Note that $Q_\theta\left(s,a\right) - y\left(r,s'\right)$ is the TD error [31]. The TD error is proportional to the gradient in the update of Q function network.

The policy parameters are learned by directly minimizing the expected KL-divergence. Therefore, we derive the update gradient of policy:

$$\nabla_\phi J_\pi(\phi) = \nabla_\phi \frac{1}{|B|} \sum_{s\in B} \left(\kappa \log \pi_\phi(\pi(s) \mid s) - Q_\theta\left(s, \pi_\phi(s)\right)\right). \tag{10}$$

Policy network outputs the Gaussian mean me and covariance var:

$$[me, var] = \pi_\phi\left(s_t\right). \tag{11}$$

Then the output Gaussian action a is firstly normalized to the interval (-1,1) by the activation function $tanh$.

$$a = \tanh(me + \varepsilon_t * var), \tag{12}$$

where ε_t is sampled by standard normal distribution. Next, two exponential equations are proposed to generate the next time-step size separately from the action a. The equation in the forward stepping is expressed as

$$h_{n+1} = m_f e^{a+n_f} * h_n, \tag{13}$$

In the backward agent, however, continuous multiplication operations are employed by

$$h^b_{n+1} = (m_b)^o \prod_{i=1}^{o} e^{-a_i+n_b} * h_{n+1} = k_b * h_{n+1}, \tag{14}$$

where k_b is defined as the backward size reduction coefficient. h^b_{n+1} is the reselected time-step size due to NR non-convergence. Constant parameters m_f, n_f, m_b, n_b are set by the desired maximum and minimum change rates of time-step size. o is the number of continuous NR non-convergence. In this work, action a_i is not the output of a explicit mathematical formula like in iter-based or SER-based stepping methods, but the output of dynamic actor network. This actor network is online updated by batch sampling and batch gradient descent. In this case, when continuous NR non-convergences occur, it can not be guaranteed that size reduction coefficient $m_b e^{-a_{i+1}+n_b}$ for the next time-step must be smaller than the previous size reduction coefficient $m_b e^{-a_i+n_b}$. However, it is confirmable that $m_b e^{-a_{i+1}+n_b} < 1$ holds and then continuous multiplication is proposed to obtain a certain smaller and smaller step size $(m_b)^o \prod_{i=1}^{o} e^{-a_i+n_b} * h_{n+1}$ when continuous NR non-convergences occur.

Reward Functions Lastly, the weighted sum of iteration states and additional bias are utilized to design the forward reward function r_f and backward reward function r_b, respectively. They are expressed as

$$r_f = \sum_{i=0}^{2} c_{fi} \left\| \tilde{S}_{RLi} \right\| + R_{fend} + B_f, \tag{15}$$

$$r_b = \sum_{i=0}^{2} c_{bi} \left\| \tilde{S}_{RLi} \right\| + R_{bend} + B_b, \tag{16}$$

where $\tilde{S}_{RLi}$ denotes the normalized states, which are obtained by using the min-max normalization for the states **NR_iters**, **Res** and **RC_rate**. c_{fi} and c_{bi} are the weight coefficients. R_{fend} and R_{bend} are constant rewards added when the current PTA iteration converges. B_f and B_b are constant biases, which aim to keep the total reward value be negative.

Algorithm 1 presents the detailed pseudo-code for the online stochastic stepping policy.

C. Proposed Valuation Splitting Architecture

In this part, continuous valuation splitting architecture with three streams is proposed to separately estimate the state value function and advantage function, which achieves better policy evaluation to accelerate simulation. In the simulation of unseen test circuits, the pre-trained OSSP by the offline training circuit set will further learn and online update using the simulation data that it has just experienced in the unseen test circuits. The data are usually insufficient and imbalance, especially at the early stage of online learning. Therefore, it is vital to improve the RL model stability and fast convergence capability.

In the conventional soft Actor-Critic model, the input of the critic network is state-action pair (s, a). Updating the critic network by TD error only improves the Q value of the current action at the current state. It means that only the value for one of the actions is updated, the values for other actions remain untouched. The state value can not be better approximated, which is important to more quickly identify the correct action during policy evaluation for faster convergence. Inspired by advantage function in policy gradient [41], [42], valuation splitting architecture is proposed for soft Actor-Critic model in this work, where the representation of state values and (state-dependent) action advantages are explicitly separated by the proposed double-Actor architecture. As shown in Fig. 2, the valuation splitting architecture has three streams to separately estimate state value $V(s)$ and the advantages $A(s, a)$ for each action. The three streams are combined via a special aggregating layer to produce an estimate of the state-action value function Q, which can replace the current single-stream Q network of conventional soft Actor-Critic model without any extra supervision. The state value $V(s)$ measures how good a particular state s is. The Q function, however, measures the value of choosing a particular action a in this state s. The advantage function $A(s, a; \beta)$, whose network parameters are β, obtains a relative measure of the importance for each action a. The stream $V(s; \alpha)$ where α are the value network parameters, learns a general value that is shared across many similar actions at s, hence leads to faster convergence.

Algorithm 1 Proposed online stochastic Stepping Policy

```
Require: Important parameters setting:
 1: Reward function $r_f$, $r_b$, step function $\tau_f$, $\tau_b$;
 2: Learning rate $\lambda_Q$, $\lambda_\phi$, $\lambda_\kappa$;
Ensure:
 3: Build algorithm structure:
 4:         Critic network $Q_{\theta_f}$, $Q_{\theta_b}$, actor network $\pi_{\phi_f}$, $\pi_{\phi_b}$;
 5:         Target actor networks $\pi^*_{\phi_f}$, $\pi^*_{\phi_b}$;
 6:         Independent buffer $\mathcal{B}_f$, $\mathcal{B}_b$, public buffer $\mathcal{B}_p$;
 7:         Temperature coefficient $\kappa_f$, $\kappa_b$;
 8:         Priority set $\mathcal{P}_f$, $\mathcal{P}_b$, $\mathcal{P}$;
 9: Connect SPICE software;
10: Simulate with initial time step $h$;
11: Record state $s$, $Conv\_NR$, $Conv\_PTA$, $Conv\_NR'$;
12: while $Conv\_PTA \neq$ True do
13:     if $Conv\_NR \neq$ False then
14:         get action $a \sim \pi_{\phi_f}(a \mid \mathbf{s})$;          ▷ forward agent
15:         update momental scaling $K_m$;          ▷ online scaling
16:         calculate next time step $h' \leftarrow K_m \tau_f(a) h$;
17:         get next state $s'$, update $Conv\_NR'$, $Conv\_PTA$;
18:         calculate $r \leftarrow r_f(s, a)$,
19:         store sample $\mathcal{B}_f \leftarrow \mathcal{B}_f \cup \{(s, a, r, s')\}$;
20:     else
21:         get action $a \sim \pi_{\phi_b}(a \mid \mathbf{s})$;          ▷ backward agent
22:         update momental scaling $K_m$;          ▷ online scaling
23:         calculate next time step $h' \leftarrow K_m \tau_b(a) h$;
24:         get next state $s'$, update $Conv\_NR'$, $Conv\_PTA$;
25:         calculate $r \leftarrow r_b(s, a)$, update priority set $\mathcal{P}_b$;
26:         store sample $\mathcal{B}_b \leftarrow \mathcal{B}_b \cup \{(s, a, r, s')\}$;
27:     end if
28:     if $Conv\_NR \oplus Conv\_NR'$ then
29:         update priority set $\mathcal{P}$;          ▷ public sample buffer
30:         store sample $\mathcal{B}_p \leftarrow \mathcal{B}_p \cup \{(s, a, r, s')\}$;
31:         update time step $h$;
32:     end if
33:     Extracted in public and independence buffer:
34:     take $\mathcal{N}$ samples according to priority set;
35:     $\Lambda_f \sim \mathcal{P}_f$, $\Lambda_b \sim \mathcal{P}_b$;          ▷ priority sampling
36:     Calculate the $Q$ value:          ▷ valuation splitting
37:     for $i$ in $\{f, b\}$ do
38:         $Q_{\theta_i}(s, a) = V_{\alpha_i}(s) + (\tilde{A}_{\beta_i}(s, \pi_{\phi_i}(s)) - \tilde{A}_{\beta_i}(s, \pi^*_{\phi_i}(s)))$
39:     end for
40:     gradient update:
41:     $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i)$ for $i \in \{f, b\}$;
42:     $\phi_i \leftarrow \phi_i - \lambda_\pi \nabla_{\phi_i} J_\pi(\phi_i)$ for $i \in \{f, b\}$;
43:     $\kappa_i \leftarrow \kappa_i - \lambda_{\kappa_i} \nabla_{\kappa_i} J(\kappa_i)$ for $i \in \{f, b\}$;
44:     Iterate to the next step:
45:     $s \leftarrow s'$, $h \leftarrow h'$, $Conv\_NR \leftarrow Conv\_NR'$;
46: end while
```

The module where the three streams of fully connected layers are combined to output a Q estimate, requires very thoughtful design. From Fig. 2,

$$Q(s, a; \alpha, \beta) = V(s; \alpha) + A(s, a; \beta), \tag{17}$$

where $Q(s, a; \alpha, \beta)$ is only a parameterized estimate of the true Q value function. Eq. (17) is unidentifiable in the sense that

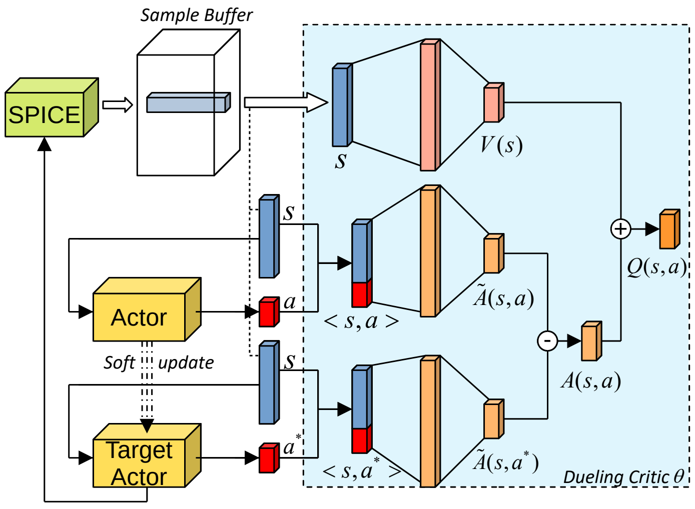

Fig. 2. Proposed valuation splitting of the critic network

$V(s)$ and $A(s,a)$ cannot be recovered uniquely from given Q, which gives poor practical performance. To solve it, the advantage function estimator is forced to have zero advantage at the optimal action, that is

$$\left\{ \begin{array}{l} Q(s,a;\theta) = V(s;\alpha) + A(s,a;\beta), \\ A(s,a;\beta) = \tilde{A}(s,a;\beta) - \max\limits_{a^* \in |\mathcal{A}|} \tilde{A}\left(s,a^*;\beta\right), \end{array} \right. \tag{18}$$

where θ represents the total parameters of the critic network which fits the Q value function. In deep Q-learning network (DQN) [43], the relative rank of the A (and hence Q) values does not be changed and any greedy or ϵ-greedy policy based on Q values is preserved. From Eq. (18), the maximum action advantage $\max\limits_{a^* \in |\mathcal{A}|} \tilde{A}\left(s,a^*;\beta\right)$ is required to obtain the identifiable advantage value.

Different from the traditional DQN method, the soft Actor-Critic model outputs a continuous action space and the maximum advantage value can not be obtained directly by the traversal method on the discrete action space. In this work, double-Actor architecture using actor and target actor is proposed to obtain the maximum advantage value. For the actor, it will converge to output an action to get the maximum Q value, which depends on different actions at the same state, namely the advantage function. Therefore, we take the output action of the target actor as the optimal action a^* with the maximum advantage value. Moreover, the update of the actor uses the policy gradient under the current policy. To distinguish the actor that provides optimal policy from that needs to update policy, a target actor network is introduced to avoid the gradient disappearance problem of advantage network updating in the same actor. It copies from the actor at regular intervals of the updating process, which makes a network difference to produce the gradient.

$$Q(s,a;\theta) = V(s;\alpha) + (\tilde{A}(s,\pi(s);\beta) - \tilde{A}(s,\pi^*(s);\beta)), \tag{19}$$

where the $\pi^*(s)$ represents the target actor. This optimization can better fit the true Q value, and improve the stability of model training and simulation efficiency.

Note that Eq. (19) is implemented as part of the network and not as a separate algorithmic step. Training of the valuation splitting architecture requires only back-propagation as standard Q networks. The estimates $V(s;\alpha)$ and $A(s,a;\beta)$ are computed automatically without any extra supervision.

D. Online Adaptive and momental step scaling

In this part, an adaptive and momental step scaling method is put forward to break through the size gain limitations to further improve the simulation efficiency. For the unseen test circuits, online learning and continuously adjusting the stepping policy is quite important to enhance the simulation performance for the new circuit structures and scales. It can be seen that the scaling bound of step size in Eqs. (13) and (14) is limited by the range (-1, 1) of action a. When continuous NR convergences occur, the limited maximum growth rate $m_f e^{1+n_f}$ may not be large enough and more aggressive step size can be adopted to further improve simulation efficiency. A similar situation also exists at the limited reduction rate for backward stepping. In this work, online adaptive scaling parameter K_m is proposed based on the momentum concept [44]. Eqs. (13) and (14) are modified to

$$h_{n+1} = K_m * m_f e^{a_n + n_f} h_n, \tag{20}$$

$$h^b_{n+1} = K_m * (m_b)^o \prod_{i=1}^{o} e^{a_{n,i} + n_b} h_n, \tag{21}$$

$$K_m = \varrho_s M_n \min(\frac{1}{\sqrt{V_n} + \epsilon}, g_n), \tag{22}$$

where ϱ_s is a scale factor. M_n is the first moment estimate of the maximum magnification and V_n is the weighted mean of second raw moment. ϵ is added to make the denominator be a positive value. g_n is designed to restrict the dynamic bound of K_m. It is based on the exponential moving averages of the history g_n themselves to smooth out unexpected large K_m, especially at the beginning of PTA iterations. The minimum bounding operation can be seen as clipping the K_m element-wisely so that the output is constrained by the current smoothed value. The detailed formulas are given by

$$\left\{ \begin{array}{l} M_n = \eta_1 M_{n-1} + (1-\eta_1) K_{n-1}, \\ V_n = \eta_2 V_{n-1} + (1-\eta_2) \frac{1}{T} \sum\limits_{j=n-T}^{n-1} \left(\widetilde{K}_j - \bar{K}_T\right)^2, \\ g_n = \eta_3 g_{n-1} + (1-\eta_3) \frac{1}{\sqrt{V_n}+\epsilon}, \end{array} \right. \tag{23}$$

where the hyper-parameters η_1, η_2 and η_3 control the exponential decay rates for the moment estimates M_n, V_n and g_n. T is the filtering period. The discrete degree of the actions is calculated. K_n is set as $m_f e^{a_n + n_f}$ and $(m_b)^o \prod_{i=1}^{o} e^{a_{n,i} + n_b}$ for the forward and backward stepping, respectively. Similarly, $\widetilde{K}_n$ are $m_f e^{\mu_n + n_f}$ and $(m_b)^o \prod_{i=1}^{o} e^{\mu_{n,i} + n_b}$ for two stepping cases, respectively. μ_n is the mean of the action a_n. $\bar{K}_T$ is the mean of the $\widetilde{K}_j$ at the last T sample time points.

M_n, V_n and g_n are used to online learn the previous time-step size experiences in the unseen circuit simulation. They can further increase step size scaling when previous step size gains are relatively uniform large and further reduces step size when previous step size gains are not stable. Meanwhile, they

TABLE I
SIMULATION EFFICIENCY COMPARISONS UNDER DPTA WITH THREE STEPPING METHODS

Circuits	OSSP		iter-based		SER-based		Speedup (vs iter-based)		Speedup (vs SER-based)	
	step	iter	step	iter	step	iter	step	iter	step	iter
UA733	21	71	39	141	37	152	1.86	1.99	1.76	2.14
UA709	24	137	532	2960	50	339	22.17	21.61	2.08	2.47
mux8	35	101	64	156	46	118	1.83	1.54	1.31	1.17
rca	19	62	29	104	28	119	1.53	1.68	1.47	1.92
fadd32	39	129	939	1968	917	1859	24.08	15.26	23.51	14.41
gm1	20	41	42	85	21	47	2.10	2.07	1.05	1.15
todd3	49	199	4648	9353	4645	9341	94.86	47.00	94.80	46.94
DCOSC	22	90	34	116	34	130	1.55	1.29	1.55	1.44
HVREF	20	73	30	96	22	93	1.50	1.32	1.10	1.27
mosamp	30	140	99	248	97	239	3.30	1.77	3.23	1.71
ab_integ	35	158	2232	4540	2167	4406	63.77	28.73	61.91	27.89
e1480	42	207	2760	5553	2741	5514	65.71	26.83	65.26	26.64
mosrect	27	83	412	838	407	826	15.26	10.10	15.07	9.95
RCA3040	19	62	29	104	28	119	1.53	1.68	1.47	1.92
TADEGLOW6TR	21	85	39	145	27	102	1.86	1.71	1.29	1.20
THM5	36	116	2661	5331	2660	5324	73.92	45.96	73.89	45.90
voter25	34	117	72	192	45	124	2.12	1.64	1.32	1.06
cram	23	62	57	130	36	87	2.48	2.10	1.57	1.40
Square_root	48	252	95	364	67	259	1.98	1.44	1.40	1.03
Suntraction	24	79	43	112	34	93	1.79	1.42	1.42	1.18
Average							19.26X	10.86X	17.82X	9.64X

help dampen oscillations by adding fractions η_1, η_2 and η_3 at the past update vector to the current update vector. As a result, faster PTA convergence can be achieved.

E. Public Sample Buffer and Priority Sampling

In this part, public sample buffer and priority sampling policy are introduced to deal with insufficient and imbalance sample data. For the online update of RL model on the unseen test circuits, the experience samples are usually insufficient and imbalance, especially at the early stage of online simulation. Moreover, dual agent structure further aggravates the insufficient sample situation, where the online training samples are divided into forward sample buffer and backward sample buffer. In this work, the public sample buffer is also introduced to improve the sampling efficiency and enhance the training stability. For forward agent, the experience where the output action causes the non-convergence of NR iterations, will be stored at the public sample buffer, since the backward agent can also learn from this experience to more sharply shorten the step size to avoid this low reward action. The similar thing also happens for backward agent. The public sample buffer can effectively alleviate the sample insufficiency situation and accelerate the convergence speed of online learning.

In this work, RL sample Sam consists of the current state s_t, action a_t, the next state s_{t+1}, calculated reward r_t, completion flag d_t (NOT **Conv_NR** OR **Conv_PTA**) and sample value Sv. It is stored using the following format:

$$Sam = < s_t, a_t, s_{t+1}, r_t, d_t, Sv >, \tag{24}$$

where sample value Sv is the node value of the binary tree in the following priority sampling section. As for action a_t, it need to be recalculated by the size reduction coefficient in Eq. (14) back to the value range (-1, 1) when continuous backward steppings occur. The recalculation of a_t is not needed for forward stepping operation.

The public sample buffer effectively increases the utilization rate of online training samples. In addition, how to select the samples to update the network is also important for enhancing the convergence performance of online learning. In this work, priority sampling [45] is employed and the samples with larger TD error (Temporal-Difference error) will be selected with higher probability. The sampling probability P_k in the k-th sample can be calculated by the following formula:

$$P_k = \frac{Sv_k}{\sum_{i=1}^{N} Sv_i}, \tag{25}$$

where $Sv_k = |\Delta TD| + \delta$. ΔTD is TD error and δ is a positive bias coefficient.

It is known that larger TD error will contribute a greater amount to the update of the critic and actor network [31]. In the implementation, the absolute value of the TD error and the sample value Sv of each sample are stored in the summation binary tree. In the random sampling process, the selected probability of the leaf node will be proportional to the stored value. Note that the samples with small TD error do not be eliminated. They are beneficial for the sample richness (diversity) and network learning. It is demonstrated that public sample buffer and priority sampling policy can remarkably enhance the PTA convergence and simulation efficiency in circuit simulation.

IV. EXPERIMENTS AND RESULTS

A. Experimental Environment and Offline Pre-training

The PTA methods with the proposed online stochastic stepping policy (OSSP) are implemented in a Spice-like circuit simulator on a server with RTX 2060 6GB GPU and i7-10750H CPU. The performance is evaluated by dozens of

TABLE II
SIMULATION EFFICIENCY COMPARISONS UNDER CEPTA WITH THREE STEPPING METHODS

Circuits	**OSSP**		**iter-based**		**SER-based**		**Speedup (vs iter-based)**		**Speedup (vs SER-based)**	
	step	**iter**	**step**	**iter**	**step**	**iter**	**step**	**iter**	**step**	**iter**
UA733	15	75	28	121	31	122	1.87	1.61	2.07	1.63
UA709	48	296	83	407	70	480	1.73	1.38	1.46	1.62
mux8	21	65	48	122	31	103	2.29	1.88	1.48	1.58
rca	15	64	21	82	16	64	1.40	1.28	1.07	1.00
slowlatch	31	154	50	169	38	250	1.61	1.10	1.23	1.62
fadd32	19	76	46	131	28	115	2.42	1.72	1.47	1.51
gm1	15	31	36	74	17	39	2.40	2.39	1.13	1.26
todd3	34	195	131	689	98	777	3.85	3.53	2.88	3.98
DCOSC	17	81	28	126	20	100	1.65	1.56	1.18	1.23
HVREF	17	73	20	77	18	106	1.18	1.05	1.06	1.45
mosamp	21	108	26	106	26	124	1.24	0.98	1.24	1.15
ab_integ	38	189	214	499	147	416	5.63	2.64	3.87	2.20
e1480	33	159	51	179	40	252	1.55	1.13	1.21	1.58
mosrect	19	57	20	65	20	64	1.05	1.14	1.05	1.12
RCA3040	15	64	21	82	24	64	1.40	1.28	1.60	1.00
TADEGLOW6TR	15	79	25	110	22	101	1.67	1.39	1.47	1.28
THM5	27	120	33	118	27	135	1.22	0.98	1.00	1.13
bjtff	23	106	35	138	25	142	1.52	1.30	1.09	1.34
toronto	42	239	70	277	44	310	1.67	1.16	1.05	1.30
add32	19	78	58	173	36	119	3.05	2.22	1.89	1.53
pchip	39	230	78	333	56	357	2.00	1.45	1.44	1.55
UA741PFBVINNEG	39	234	57	231	61	340	1.46	0.99	1.56	1.45
Average							1.99X	1.55X	1.52X	1.52X

benchmark transistor circuits. For fair comparisons, the performance is evaluated by dozens of benchmark transistor circuits, whose details can be easily found and checked in [46].

At first, we select seven typical circuits (two MOS circuits and five BJT circuits) as the training dataset to achieve the offline pre-training of stepping model. The PTA with the proposed OSSP conducts the DC analysis for the seven circuits and the obtained samples are utilized to continuously update the actor and critic network of the forward agent and backward agent. After 24 training epochs, the stepping model tends to converge and the pre-training OSSP is obtained.

Then, for unseen test circuits, the pre-training OSSP will be online updated by the new samples obtained from the unseen test circuits and outputs the adaptive and intelligent step size for PTA iterations to accelerate the DC analysis convergence.

In this work, PTA with the online stochastic stepping policy (OSSP-PTA) is put forward to enhance the convergence and computational efficiency of DC analysis. For comparisons, PTA with other two typical stepping policies (iter-based and SER-based) are also implemented. As for the convergence, whether or not finding the dc operating points of test circuits is illustrated. Considering the computational efficiency, the number of PTA steps and NR iterations, as two widely-used evaluation metrics are employed [9], [15], [24].

B. Simulation Efficiency Comparisons

To comprehensively evaluate the performance of the proposed OSSP, it is implemented in two typical PTA methods (DPTA [23] and CEPTA [25]) and compared with two widely-used and effective time-step control methods (iter-based stepping method [29] and SER-based stepping method [39]). Both convergence and simulation efficiency performances are compared and analyzed.

For the simulation efficiency, firstly the DPTA with the proposed OSSP, conventional iter-based as well as SER-based stepping methods are implemented and compared. The number of NR iterations and PTA steps with the three stepping methods for 20 test circuits are shown in Table I. For more distinct comparisons, the speedup between different stepping methods are also presented. From this table, it is clear that the DPTA with the proposed OSSP has better simulation efficiency performance, which shows the average 10.86X reduction of NR iterations and 19.26X reduction of PTA steps compared with iter-based stepping method. Meanwhile, it shows average 9.64X reduction of NR iterations and 17.82X reduction of PTA steps compared with SER-based adaptive stepping method.

Figure 3 gives the simulation process of a example circuit UA709 with the three stepping methods. The bar charts in the left represent the change rate of step size, which is in logarithmic form so that the forward and backward steppings are symmetric with respect to 0-axis. The bar charts in the right give the NR iteration number taken in each time-step. The red horizontal lines in the left bar charts represents the average change rate of the forward step sizes with the three stepping methods. In the right bar charts, the orange horizontal lines give the average NR iteration number of total time-steps (TA: Total Average), and the green lines represent the average NR iteration number in the forward steppings (FA: Forward Average). In summary, it is clear that the proposed OSSP requires the least PTA steps and NR iterations, for two main reasons. On the one hand, in the forward stepping part, the proposed method has the highest average size change rate ($2^{1.22}$=2.329X), but the average NR iteration number at each

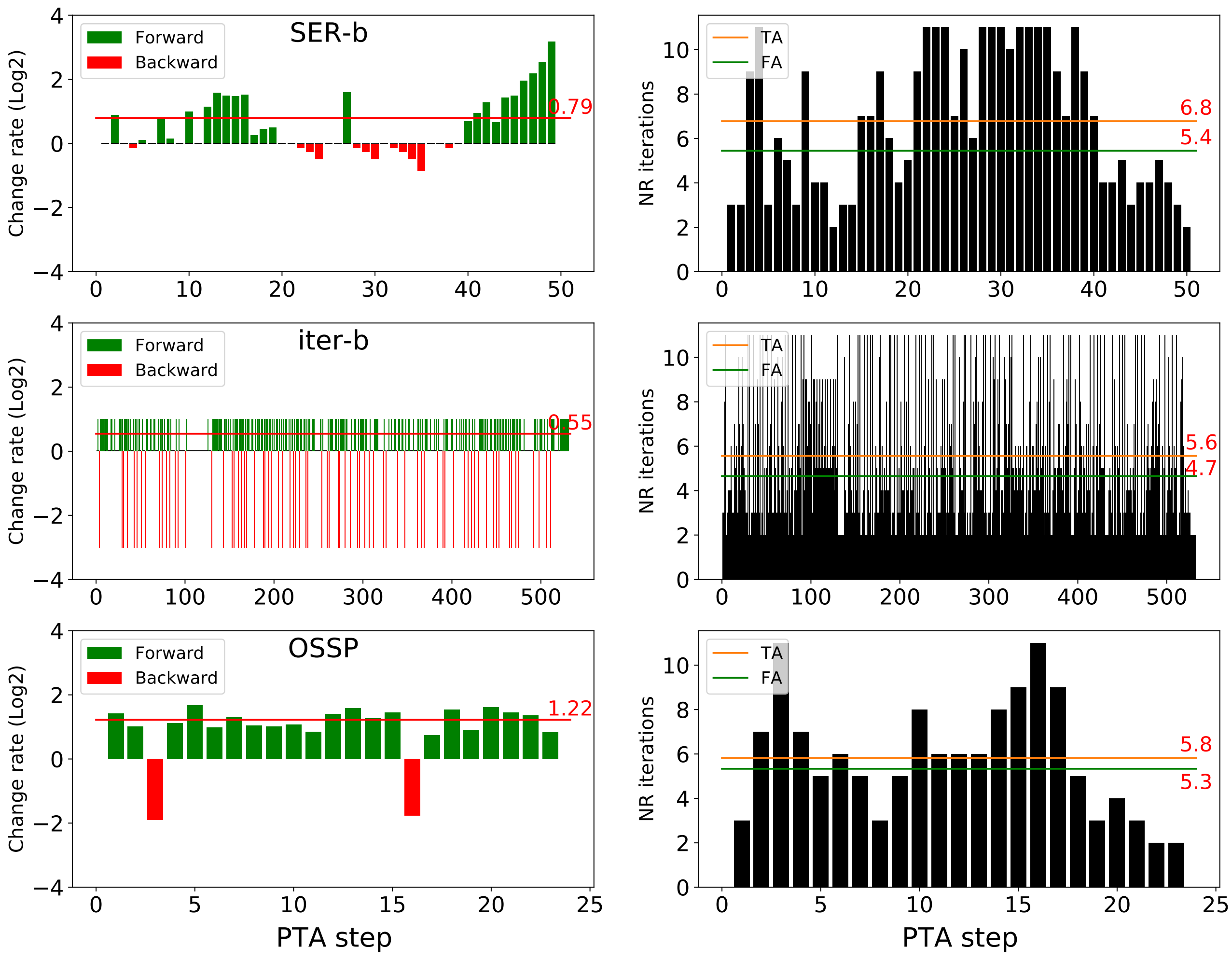

Fig. 3. Simulation process of circuit UA709 with three stepping methods

forward time step (FA=5.3) is almost the same as those of other two methods (5.4 and 4.7), which means that the OSSP makes the PTA iterations converge with larger step size while without consuming more NR iterations at each time step. On the other hand, in the backward stepping part (negative red parts in the left charts), our method not only usually has less backward stepping operations, but also has lower and distinct shrink rate to adaptively solve non-convergences. Compared with the iter-based stepping method in this circuit, too many backward steppings seriously affect the simulation efficiency (large speedup 21.61X of NR iterations is obtained). Moreover, its shrink rate is fixed at 1/8 (2^{-3}). The sharp backward stepping policy in iter-based method may reduce the probability of continuous backward steppings, but also depresses the step size and also leads to low convergence speed. Meanwhile, the SER-based method adopts a conservative and slow backward stepping policy (G/(G+1) in Eq. (5)) and brings about many continuous backward steppings. As a result, a large number of PTA steps and NR iterations are wasted on the non-convergent time-steps.

In addition, the proposed OSSP is compatible to kinds of PTA solvers. Similarly, the CEPTA with the three stepping methods are implemented. The corresponding number of NR iterations, PTA steps, and speedups for many test circuits are compared in Table II. From Table II, the simulation efficiency can also be enhanced remarkably with the OSSP. Compared with the iter-based stepping method, the acceleration ratio of NR iterations/PTA steps are 3.53X/5.63X in maximum and 1.55X/1.99X in average, respectively. Compared with the SER-based adaptive stepping method, the corresponding speedups are 3.98X/3.87X in maximum and 1.52X/1.52X in average.

To sum up, the proposed OSSP can not only achieve larger average step size at forward steppings, but also adaptively give as low as shrink rate at backward steppings while minimizing the occurrence of continuous backward steppings. Therefore, higher simulation efficiency is obtained.

C. PTA Convergence Comparisons

Apart from simulation efficiency, PTA convergence guarantee is actually more important and promising especially for large-scale circuits and "difficult" circuits. It is quite frustrating to take several hours or even days for DC simulation but non-convergence occurs. It is highly desirable to make non-convergence cases converge.

First, the test results of ten "difficult" circuits under the DPTA method with three stepping methods are shown in Table III. "-" denotes non-convergence. From this table, it can be seen that the proposed OSSP can achieve PTA convergence for some circuits that other two stepping methods fail to converge.

TABLE III
CONVERGENCE COMPARISONS UNDER DPTA

Circuits	iter-b	SER-b	OSSP
bjtff	-	-	207
add20	-	-	307
add32	-	-	174
gm19	-	-	379
toronto	-	-	241
verg	-	-	22
sram	-	-	372
mem_plus	-	-	544
pchip	1049	-	399
UA741PFBVIN	-	-	293

Similar conclusions can also be obtained for CEPTA case. Table IV presents the test results of CEPTA with the three stepping methods. From Tables. III and IV, it is demonstrated that the proposed OSSP can effectively improve the convergence performance of the DPTA and CEPTA (widely-used PTAs) solvers compared with other two stepping methods.

TABLE IV
CONVERGENCE COMPARISONS UNDER CEPTA

Circuits	iter-b	SER-b	OSSP
verg	22	-	22
optrans	-	-	4573
pump	22	-	22
mem_plus	-	-	230
jge	-	-	1517

Figure 4 presents the simulation process of a example circuit jge with two stepping methods, including step sizes and node voltage curves (V150 node). In the step size subfigure, the red cross and blue dot represent non-convergence and convergence steps, respectively. It can be seen that even the voltage solution curve oscillates between two obvious different voltages, the iter-based stepping method just judges NR iteration number and outputs a fixed same step size, rather than comprehensively considers the relative change of solution x so that a more "intelligent" step size can be generated to jump out of the local oscillation. In contrast, the OSSP does not fall into the local non-convergence point and the node voltage curve approaches the steady state around 1.7V easily.

Similar oscillation non-convergence phenomenon also appears in the SER-based stepping method. In Fig. 5, the simulation process of another example circuit add20 with two stepping methods is shown. We can see that the heuristic SER-based stepping policy relies on the artificial empirical formula to deal with the oscillatory and finally the output step size changes cyclically between keeping same and slightly decreasing due to non-convergence. Compared with it, the proposed stochastic stepping policy has stronger discrete size distribution and stepping space exploration ability. It helps the PTA damp out the local oscillation to achieve convergence.

In addition, aside from the oscillation, another common non-convergence situation is called "Time step too small", where PTA iterations still fail to converge even the step size hits the lowest bound. The simulation process of a "difficult" example circuit optrans with the proposed OSSP and SER-based stepping methods is shown in Fig. 6. Red cross and blue dot also represent non-convergence step and convergence step. From this figure, the SER-based stepping method just keeps decreasing the step size to struggle for NR convergence, but the PTA convergence condition can not be satisfied even the step size reaches the lowest limit (1E-9 s). In contrast, the step size by the proposed OSSP can maintain in certain range with strong discrete distribution and does not decrease continuously to achieve convergence. Thus the proposed OSSP performs well even in some "difficult" circuits where "time step too small" will occur by conventional stepping methods.

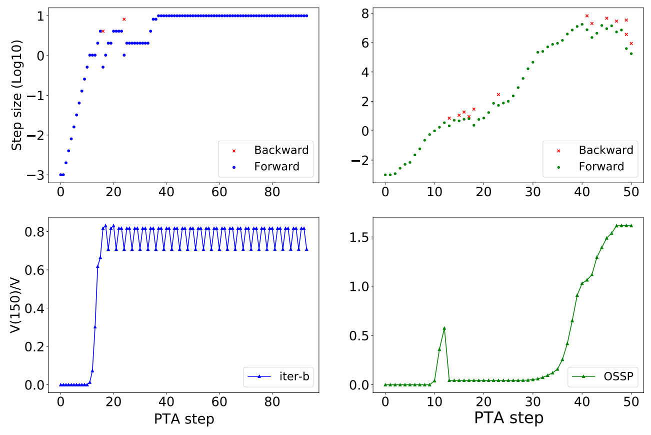

Fig. 4. Simulation process of circuit jge with iter-based and OSSP

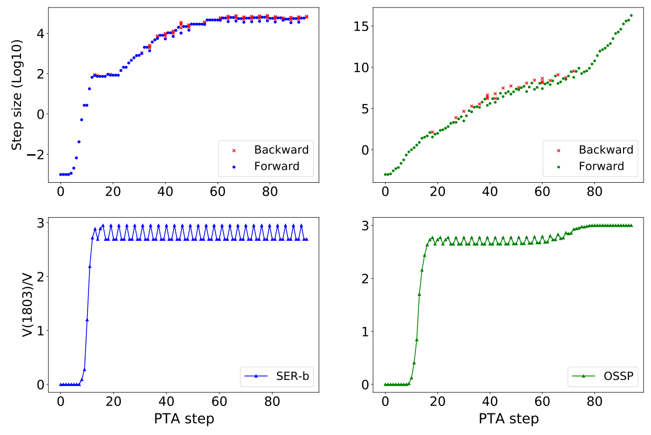

Fig. 5. Simulation process of circuit add20 with SER-based and OSSP

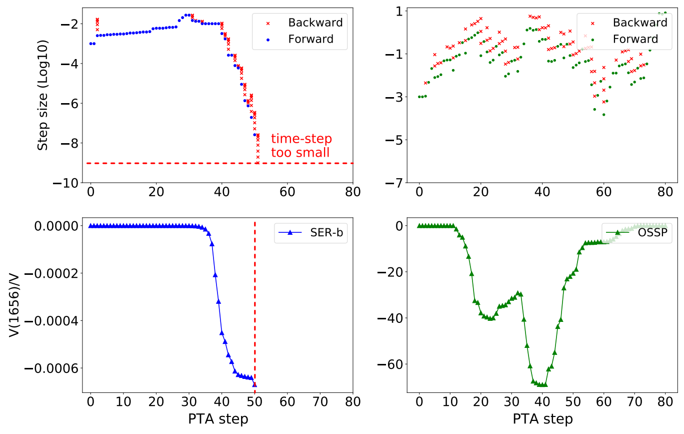

Fig. 6. Simulation process of circuit optrans with SER-based and OSSP

D. Performance Comparisons at Different Pseudo Capacitors

In PTA methods, a set of nonlinear algebraic equations are turned into a system of ordinary differential equations (ODEs) by inserting pseudo-elements into the circuit. To obtain an easy-to-solve ODE system rather than an ill-defined system (e.g., containing bifurcation, fold), the required pseudo-elements (mainly pseudo capacitors) can be totally different for different circuits. It is common that various sub-circuits in a large-scale circuit require different or even incompatible pseudo-elements to achieve good convergence. Therefore, a robustness stepping strategy which has a wide applicability for different pseudo-elements is crucial to enhance PTA simulation efficiency and convergence in the large-scale circuits.

TABLE V
PERFORMANCE COMPARISONS AT DIFFERENT PSEUDO CAPACITORS

Citcuits	Algorithms	C	iter	step	rejected-step
jge	iter-b	1.0E-04	-	-	-
		1.0E-02	-	-	-
		1.0E+04	-	-	-
	OSSP	1.0E-08	910	125	43
		1.0E-04	454	67	14
		1.0E-02	643	112	23
		1.0E+04	661	123	24
		1.0E+08	655	136	21
	SER-b	1.0E-04	6044	1025	57
		1.0E-02	-	-	-
		1.0E+04	-	-	-
votor	iter-b	1.0E-08	14695	3080	47
		1.0E-02	-	-	-
		1.0E+04	-	-	-
	OSSP	1.0E-08	728	102	29
		1.0E-04	10115	1653	630
		1.0E-02	5722	903	353
		1.0E+04	11328	1875	697
		1.0E+08	10859	1823	683
	SER-b	1.0E-08	1225	158	52
		1.0E-02	-	-	-
		1.0E+04	-	-	-

In Table V, the test results of two example “difficult” circuits with different pseudo capacitors under the three stepping methods are shown. “-” also represents non-convergence. From this table, SER-based method takes more NR iterations and even fails to converge under some pseudo capacitors, while simple iter-based method fails to converge in most cases. Compared with other two stepping methods, it can be seen that the proposed OSSP always converges, even the value of pseudo capacitors change in a wide range. Therefore, the proposed OSSP can effectively enhance the robustness of PTA methods for large-scale circuits.

E. Ablation Experiments

In this section, ablation experiments for online stochastic stepping policy, value splitting and online momental scaling are conducted to verify their effectiveness for improving PTA convergence performance, respectively.

First, Table VI presents the comparison results of the proposed OSSP with and without the value splitting, which can better approximate the state value and more quickly identify the correct action during policy evaluation. From this table, the NR iterations of test circuits can be effectively decreased by an average of 1.49 times (3.67X in maximum).

TABLE VI
PERFORMANCE COMPARISONS UNDER PROPOSED STEPPING POLICY WITH AND WITHOUT THE VALUE SPLITTING

Circuits	OSSP		OSSP (non-value)		Speedup
	step	iter	step	iter	iter
DIFFPAIR	16	93	24	107	1.15
jge	249	1517	316	2208	1.46
UA741	42	286	46	353	1.23
UA741PFBVIP	36	230	38	269	1.17
DCOSC	17	81	21	84	1.04
UA709	48	296	65	481	1.63
UA733	15	75	15	86	1.15
bjtff	23	106	28	163	1.54
pchip	39	230	45	323	1.40
mike2	24	89	24	116	1.30
bias	40	260	129	953	3.67
mosamp	21	108	30	173	1.60
add20	34	204	40	277	1.36
slowlatch	31	154	29	193	1.25
schmitfast	20	82	21	111	1.35
Average	-	-	-	-	1.49X

Second, the proposed online stochastic stepping policy and conventional deterministic policy is compared in Table VII. From the results on test circuits, the NR iterations can be achieved 1.56 times acceleration in average by the proposed online stochastic policy. More importantly, it can be seen that the conventional deterministic policy fail to converge on six “difficult” circuits, but the proposed online stochastic stepping policy possesses stronger stepping space exploration ability and still converges to dc solutions.

TABLE VII
PERFORMANCE COMPARISONS UNDER PROPOSED STEPPING METHOD WITH ONLINE STOCHASTIC OR CONVENTIONAL DETERMINISTIC STEPPING POLICY

Circuits	OSSP		OSSP (non-stocha)		Speedup
	step	iter	step	iter	iter
DIFFPAIR	16	93	-	-	-
jge	249	1517	-	-	-
UA741	42	286	438	54	1.53
UA741PFBVIP	36	230	287	40	1.25
DCOSC	17	81	142	25	1.75
UA709	48	296	468	60	1.58
UA733	15	75	98	22	1.31
bjtff	23	106	184	29	1.74
pchip	39	230	-	-	-
mike2	24	89	-	-	-
bias	40	260	309	39	1.19
mosamp	21	108	199	27	1.84
add20	34	204	362	48	1.77
schmitfast	20	82	-	-	-
toronto	42	239	385	50	1.61
schmitslow	28	137	-	-	-
TRCKTorig	15	47	91	19	1.94
Average	-	-	-	-	1.56X

At last, the effectiveness of the proposed online momental step scaling is verified in Table VIII. It can break through the gain limitation from Eqs. (13) and (14) and online further adjusts the step size scaling by encodering the newly-experienced size samples of the unseen test circuits. From this table, the online momental step scaling can further speed up the NR iterations by average 1.64X (5.41X in maximum) to enhance the simulation efficiency.

TABLE VIII
PERFORMANCE COMPARISONS UNDER PROPOSED STEPPING METHOD WITH AND WITHOUT ONLINE MOMENTAL STEP SCALING

Circuits	**OSSP**		**OSSP (non-scaling)**		**Speedup**
	step	**iter**	**step**	**iter**	**iter**
DIFFPAIR	16	93	18	106	1.14
jge	249	1517	316	2208	1.46
UA741	42	286	57	448	1.57
UA741PFBVIP	36	230	51	367	1.60
DCOSC	17	81	15	83	1.02
UA709	48	296	49	398	1.34
UA733	15	75	17	80	1.07
bjtff	23	106	24	135	1.27
pchip	39	230	45	343	1.49
mike2	24	89	24	93	1.04
bias	40	260	167	1406	5.41
mosamp	21	108	27	181	1.68
add20	34	204	34	229	1.12
toronto	42	239	68	473	1.98
ring11	15	56	16	80	1.43
Average	-	-	-	-	1.64X

V. CONCLUSIONS

In this paper, we propose an online stochastic stepping policy (OSSP) for PTA, which employs dual autonomous RL agents to online adaptively adjust the forward and backward step sizes of the unseen test circuits for accelerating PTA convergence. To achieve better policy evaluation, we design a continuous valuation splitting architecture with three streams to produce separate estimates of the state value function and advantage function. To deal with the circuit differences between offline training and online prediction, online momental size scaling based on first and second moment estimates was proposed to work with online stepping policy update for further enhancing simulation efficiency. By evaluating OSSP on some banchmark circuits, we demonstrate that the proposed method is able to online further adjust the step size with strong stepping space exploration ability and obtain remarkable PTA convergence and simulation efficiency enhancement (up to 94.86X less PTA steps and 47.00X less NR iterations).

REFERENCES

[1] T. Nakura, "SPICE Simulation", *Essential Knowledge for Transistor-Level LSI Circuit Design*. Springer, Singapore, pp. 19-47, 2016.

[2] J. Zhao, Y. Wen, Y. Luo, Z. Jin, W. Liu, and Z. Zhou, "Sflu: Synchronization-free sparse lu factorization for fast circuit simulation on gpus," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 37–42, 2021.

[3] S. Uatrongjit, B. Kaewkham-Ai, and K.Prakobwaitayakitt, "Finding all dc operating points of nonlinear circuits based on interval linearization and coordinate transformation," in *2022 International Electrical Engineering Congress (iEECON)*, pp. 1–4, 2022.

[4] Y. Chen, H. Pei, X. Dong, Z. Jin, and C. Zhuo, "Application of deep learning in back-end simulation: Challenges and opportunities," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 641–646, 2022.

[5] I. N. Hajj, "Circuit theory in circuit simulation," *IEEE Circuits and Systems Magazine*, vol. 16, no. 2, pp. 6–10, 2016.

[6] K. S. Kundert and P. Gray, *The Designer's Guide to Spice and Spectre*. USA: Kluwer Academic Publishers, 1995.

[7] S. Hamedi-Hagh, *Computational Electronic Circuits: Simulation and Analysis with MATLAB®*. Springer International Publishing, 2022.

[8] C. Lemke, "Pathways to solutions, fixed points, and equilibria (cb garcia and wj zangwill)," *SIAM Review*, pp. 445–446, 1984.

[9] D. Niu, K. Sako, G. Hu, and Y. Inoue, "A globally convergent nonlinear homotopy method for mos transistor circuits," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 95, no. 12, pp. 2251–2260, 2012.

[10] E. Yilmaz and M. M. Green, "Some standard spice dc algorithms revisited: Why does spice still not converge?," in *1999 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 6, pp. 286–289, IEEE, 1999.

[11] M. Bhattacharya and P. Mazumder, "Augmentation of spice for simulation of circuits containing resonant tunneling diodes,"*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 1, pp. 39–50, 2001.

[12] E. J. W. ter Maten, T. G. Beelen, A. de Vries, and M. van Beurden, "Robust time-domain source stepping for dc-solution of circuit equations," in *Scientific Computing in Electrical Engineering (SCEE 2012), September 11-14, 2012, Zurich, Switzerland*, pp. 39–40, 2012.

[13] T. Najibi, "Continuation methods as applied to circuit simulation," *IEEE Circuits and Devices Magazine*, vol. 5, no. 5, pp. 48–49, 1989.

[14] L. T. Watson, "Globally convergent homotopy methods: a tutorial," *Applied Mathematics and Computation*, vol. 31, pp. 369–396, 1989.

[15] Y. INOUE, S. KUSANOBU, K. YAMAMURA, and M. ANDO, "An initial solution algorithm for globally convergent homotopy methods," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 87, no. 4, pp. 780–786, 2004.

[16] R. Wilton, "Supplementary algorithms for dc convergence (circuit analysis)," in *IEE Colloquium on SPICE: Surviving Problems in Circuit Evaluation*, pp. 3–1, IET, 1993.

[17] H. Yu, Y. Inoue, Y. Matsuya, and Z. Huang, "An effective pseudo-transient algorithm for finding dc solutions of nonlinear circuits," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 89, no. 10, pp. 2724–2731, 2006.

[18] J. Deng, K. Batselier, Y. Zhang, and N. Wong, "An efficient two-level dc operating points finder for transistor circuits," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2014.

[19] Z. Jin, T. Feng, Y. Duan, X. Wu, M. Cheng, Z. Zhou, and W. Liu, "Palbbd: A parallel arclength method using bordered block diagonal form for dc analysis," in *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, pp. 327–332, 2021.

[20] F. N. Najm, *Circuit simulation*. John Wiley Sons, 2010.

[21] W. Weeks, A. Jimenez, G. Mahoney, D. Mehta, H. Qassemzadeh, and T. Scott, "Algorithms for astap–a network-analysis program," *IEEE Transactions on Circuit Theory*, vol. 20, no. 6, pp. 628–634, 1973.

[22] L. Goldgeisser, E. Christen, M. Vlach, and J. Langenwalter, "Open ended dynamic ramping simulation of multi-discipline systems," in *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems(Cat. No. 01CH37196)*, vol. 5, pp. 307–310, IEEE, 2001.

[23] X. Wu, Z. Jin, D. Niu, and Y. Inoue, "A pta method using numerical integration algorithms with artificial damping for solving nonlinear dc circuits," *Nonlinear Theory and Its Applications, IEICE*, vol. 5, no. 4, pp. 512–522, 2014.

[24] Z. Jin, X. Wu, D. Niu, X. Guan, and Y. Inoue, "Effective ramping algorithm and restart algorithm in the spice3 implementation for dpta method," *Nonlinear Theory and Its Applications, IEICE*, vol. 6, no. 4, pp. 499–511, 2015.

[25] H. Yu, Y. Inoue, K. Sako, X. Hu, and Z. Huang, "An effective spice3 implementation of the compound element pseudo-transient algorithm," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 90, no. 10, pp. 2124–2131, 2007.

[26] F. Dhiabi, M. L. Megherbi, A. Saadoune, R. Carotenuto, and F. Pezzimenti, "Pyams: A new software for modeling analog elements and circuit simulations," *INTERNATIONAL JOURNAL*, vol. 10, no. 4, pp. 233–242, 2021.

[27] S.-H. Manual, "Release 1999.2," *Avanti Corporation*, 1999.

[28] K. S. Kundert and P. C. Gray, "The designer's guide to spice and spectre," 1995.

[29] H. R. Pota, "Inside spice," 2010.

[30] Z. Jin, H. Pei, Y. Dong, X. Jin, X. Wu, W. W. Xing, and D. Niu, "Accelerating nonlinear dc circuit simulation with reinforcement learning," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 619–624, 2022.
[31] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[32] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *arXiv preprint arXiv:1811.12560*, 2018.
[33] A. Hosny, S. Hashemi, M. Shalan, and S. Reda, "Drills: Deep reinforcement learning for logic synthesis," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 581–586, IEEE, 2020.
[34] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2020
[35] J. Liang, V. Ganesh, P. Poupart, and K. Czarnecki, "Exponential recency weighted average branching heuristic for sat solvers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.
[36] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, pp. 679–684, 1957.
[37] C.-W. Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Transactions on circuits and systems*, vol. 22, no. 6, pp. 504–509, 1975.
[38] D. Niu, X. Wu, Z. Jin, and Y. Inoue, "An effective and globally convergent newton fixed-point homotopy method for mos transistor circuits," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 96, no. 9, pp. 1848–1856, 2013.
[39] X. Wu, Z. Jin, D. Niu, and Y. Inoue, "An adaptive time-step control method in damped pseudo-transient analysis for solving nonlinear dc circuit equations," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 100, no. 2, pp. 619–628, 2017.
[40] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al., "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.
[41] M. E. Harmon, L. Baird, and A. H. Klopf, "Advantage updating applied to a differential game," *Advances in neural information processing systems*, vol. 7, 1994.
[42] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.
[43] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[44] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Computer Science*, 2014.
[45] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015. cite *arxiv:1511.05952*, Comment: Published at *ICLR* 2016.
[46] J. Barby and R. Guindi, "Circuitsim93: A circuit simulator benchmarking methodology case study," in *Sixth Annual IEEE International ASIC Conference and Exhibit*, pp. 531–535, IEEE, 1993.