nature communications

Article

https://doi.org/10.1038/s41467-025-60085-w

Memristive Bellman solver for decision-making

Received: 29 September 2024

Accepted: 14 May 2025

Published online: 27 May 2025

Check for updates

Zhe Feng[1,6], Zuheng Wu[1,6] ✉, Jianxun Zou[1,6], Lingli Cheng[2], Xiaolong Zhao[3], Xumeng Zhang[2], Jian Lu[4], Cong Wang[5], Yilin Wang[3], Haochen Wang[1], Wenbin Guo[1], Zhibin Qian[1], Yunlai Zhu[1], Zuyu Xu[1], Yuehua Dai[1] ✉ & Qi Liu[2] ✉

The Bellman equation, with a resource-consuming solving process, plays a fundamental role in formulating and solving dynamic optimization problems. The realization of the Bellman solver with memristive computing-in-memory (MCIM) technology, is significant for implementing efficient dynamic decision-making. However, the iterative nature of the Bellman equation solving process poses a challenge for efficient implementation on MCIM systems, which excel at vector-matrix multiplication (VMM) operations but are less suited for iterative algorithms. In this work, by incorporating the temporal dimension and transforming the solution into recurrent dot product operations, a memristive Bellman solver (MBS) is proposed, facilitating the implementation of the Bellman equation solving process with efficient MCIM technology. The MBS effectively reduces the iteration numbers and which further enhanced by approximated solutions leveraging memristor noise. Finally, the path planning tasks are used to verify the feasibility of the proposed MBS. The theoretical derivation and experimental results demonstrate that the MBS effectively reduces the iteration cycles, facilitating the solving efficiency. This work could be a sound of choice for developing high-efficiency decision-making systems.

The optimal decision-making system, minimizing or maximizing certain performance metrics by adjusting the control variables of the system, is essential for various fields of application, such as aircraft flight path planning, robot motion control, financial portfolio optimization, etc.[1–3]. Dynamic programming, a thought that divide a complex problem into simple subproblems, is a common method to solve the optimization problem of multi-stage decision process[4,5]. As the core foundation of dynamic programming, the Bellman equation plays a fundamental role in formulating and solving dynamic optimization problems to decision-making[6,7].

The Bellman equation describes the relationship between the value of a state or action and the expected immediate reward plus the value of the subsequent state or action[8–10]. This iterative process makes solving of Bellman equation a computationally intensive task. Hence, realization of Bellman solver with efficient computing technology is significant for developing efficient dynamic decision-making system. Fortunately, computing-in-memory (CIM) technologies are promising for advancing the computing efficiency. Memristors, including resistive[11–13], phase change[14–16], ferroelectric[17–19] and magnetism memristors[20–22] with advantages of non-volatile memory properties[23–25], low power consumption[26–28], have garnered significant attention for CIM technologies realization (MCIM). The crossbar structure can efficiently complete the vector matrix multiplication (VMM) operations by using Ohm's law and Kirchhoff's law[29–31]. Hence, implementing memristive Bellman solver (MBS) would have the potential for facilitating efficient dynamic decision-making process.

[1]School of Integrated Circuits, Anhui University, Hefei, Anhui, China. [2]Frontier Institute of Chip and System, Fudan University, Shanghai, China. [3]School of Microelectronics, University of Science and Technology of China, Hefei, China. [4]Research Center for Intelligent Computing Hardware, Zhejiang Laboratory, Hangzhou, China. [5]Institute of Brain-inspired Intelligence, National Laboratory of Solid State Microstructures, School of Physics, Collaborative Innovation Center of Advanced Microstructures, Nanjing University, Nanjing, China. [6]These authors contributed equally: Zhe Feng, Zuheng Wu, Jianxun Zou.
✉e-mail: wuzuheng@ahu.edu.cn; daiyuehua@ahu.edu.cn; qi_liu@fudan.edu.cn

In previous works, researchers have demonstrated MCIM for accelerating dynamic optimization and decision-making tasks, such as reinforcement learning and Markov decision problem.[12,32]. However, in these works, the Bellman equation solving process still relies on Von-Neumann computing systems. The Bellman equation is an iterative double expectation equation, and its iterative solving process is inherently difficult to implement efficiently on MCIM systems due to the mismatch with the VMM operation principle. Furthermore, the process of solving the Bellman equation is a precise process, which is different from the approximate solution of the neural network. The Bellman equation is fundamentally a precise mathematical formulation to compute exact value functions or optimal policies, while neural networks inherently rely on approximation due to their function approximation nature. Therefore, as the complexity of the task being solved increase, the number of iterations of Bellman's solution process will increase sharply[33–35]. In addition, the intrinsic noise in memristive devices further restricts the precise solution of the Bellman equation based on MCIM. Whereas, the researchers prove that the recursive memristive VMM operation and the intrinsic noise of the memristor can accelerate the convergence rate and reduce the complexity of the iterative algorithms[36–38]. Hence, the software and hardware co-optimization efforts have to be paid for developing efficient MBS to further enhance the efficiency of dynamic decision-making system.

In this work, we propose an MBS with software and hardware co-optimization efforts. Software-wise, we have incorporated the temporal dimension and transformed the solution of iterative double expectations into a recurrent dot product operation, which facilitates the implementation of the Bellman equation solving process with MCIM technology. The MBS effectively reduces the iteration numbers (see **Methods**). Furthermore, hardware-wise, we found that the inherent noise characteristics of memristors can contribute to finding an approximate solution, further reducing the iterations (see **Methods**). Finally, the path planning tasks are used to verify the feasibility of the proposed MBS. The results indicate that the MBS shows potential improvements in computation complexity (from $O(k|S|^2)$ to $O(k'|S|)$, $k' < k$, here, k and k' denotes the number of iterations required for the traditional Bellman equation and MBS to converge, respectively), energy consumption (~10^3×) and computing speed (see Supplementary Table S1, S2 and Note 2 for details). This work offers a reliable framework for building efficient dynamic decision-making systems.

Results and Discussion

Challenges and solutions for realizing a memristive Bellman solver

As shown in Fig. 1a, dynamic programming, an idea that breaks down a complex problem into simple ones, has been widely applied in various dynamic optimization fields such as reinforcement learning, auto-driving and path planning etc. The dynamic optimization problem aims to find the optimal decision from the initial state to the end state. The core foundation is to solve Bellman equation to find the most valuable decision. Figure 1b illustrates a scene with three states (S_1, S_2, S_3) where each state can transition to any other state (including itself) and S_1 set as Start and S_3 set as End. The decision could be made by the optimized value of each state, which could be obtained by solving the Bellman equation. It should be noted that, here the presentation form of Bellman equation, which is the backward induction form, is different from that of standard reinforcement learning[12,39], but their mathematical essence is the same (see Supplementary Note 1). Moreover, the action space is implicitly incorporated into the state transition probabilities. Figure 1c (left panel) shows the action implicit backward induction Bellman equation for optimizing each state value in Fig. 1b, which features an iterative solving process. It should be noted that in the action implicit backward induction Bellman equation (without time dimension), $\mathbf{V}(S_n)$ and $\mathbf{V}(S_{n\text{-}1})$ represents the current state value and the previous state value, respectively. The current state and the previous state can be any one in the state space **S**. That is to say, in Fig. 1c (left panel), the green-marked $V(S_1)$, $V(S_2)$ and $V(S_3)$ can represent the previous values of states S_1, S_2, and S_3 respectively (because the $\sum$ has been unfolded). However, the current state value marked in red is uncertain. It can be any one of the states S_1, S_2, and S_3. Therefore, although formally it seems that the equation can be compatible with MCIM. However, in the actual operation process, there is no time dimension, and we cannot determine how the current output correspond to next input. Therefore, the action implicit backward induction Bellman equation cannot be compatible with MCIM. To facilitate the Bellman solving process with efficient MCIM operation principle, the time dimension has been introduced. The introduced time dimension could transform the iterative solution process into a recurrent dot product process (Fig. 1c, right panel) to reduce the computation complexity (reduce iteration numbers) (see Supplementary Note 2 and **Methods**), and without influencing the convergency properties of Bellman equation solving process. The value of current state ($\mathbf{V}(S_t)$) could be obtained by the dot product operation between the sum of current state's reward and value of previous state multiplied by an attenuation constant ($\mathbf{R}(S_t)+\gamma\mathbf{V}(S_{t-1})$) and the state transition probabilities ($\sum_{S_{t-1}\in\mathbf{S}}\mathbf{P}(S_{t-1}|S_t)$). The dot product operation can be implemented by the memristor array efficiently, by mapping the $\mathbf{R}(S_t)+\gamma\mathbf{V}(S_{t-1})$ as input and mapping the $\sum_{S_{t-1}\in\mathbf{S}}\mathbf{P}(S_{t-1}|S_t)$ as conductances (weights).

In addition, in digital computing systems, the solution process of the Bellman equation is highly precise. However, if the transition probabilities are the same, more iterations are needed to achieve the distinction (Fig. 1d). As the complexity of the problem increases, the likelihood of this scenario will also rise. That is to say, as the problem grows more complex, the number of iterations needed to solve it will increase. When the time dimension is introduced, solving the Bellman equation using MCIM will face challenges. Due to the inherent noise of memristors (such as write noise and read noise), the memristor cells cannot accurately represent the state transition probabilities. Fortunately, we found that the intrinsic noise of memristors could serve as a significant computational resource to improve the efficiency of the Bellman solving process. The intrinsic noise of memristors facilitates the distinction of the same state transition probabilities, enabling an approximate solution process and reducing iterations (Fig. 1e). Even though solving the Bellman equation with MCIM involves approximation, it does not affect the convergence of the Bellman equation (see **Methods**). That is to say, by leveraging approximate solutions, an approximate optimal solution can still be found. In fact, in various application scenarios, an approximately optimal solution is sufficient.

Memristive Bellman solver for decision-making

Here, the workflow of the MBS for decision-making is introduced. First, we take the scenario in Fig. 1b as an example to describe the process of solving the Bellman equation using the MBS. To clarify the MBS implementation, the two operators, that is, memristive Bellman dot operator (MB_{dot} operator) and memristive Bellman recurrent operator (MB_r operator), are defined (Fig. 2a). The MB_{dot} operator indicates that the dot product between the reward vector ($[R(S_1), R(S_2), R(S_3)]$) and the state transition probabilities. The reward vector is mapped to the input of the memristor array, while the state transition probabilities are mapped to the conductance states (weight matrix) of the memristor array. In this MB_{dot} operation process, the output result from the previous time step is multiplied by the attenuation parameter γ and returned to the input terminal, then combined with the input reward signal to form the total input at the next time step. It is worth noting that the number of MB_{dot} operations correlates with the time parameter t. Additionally, within a single time step, only one state value is updated. Hence, at the next time step, only the rewards corresponding to the updated state change. After completing MB_{dot} operations (one solving round, that is, one MB_r operation), the obtained reward vector

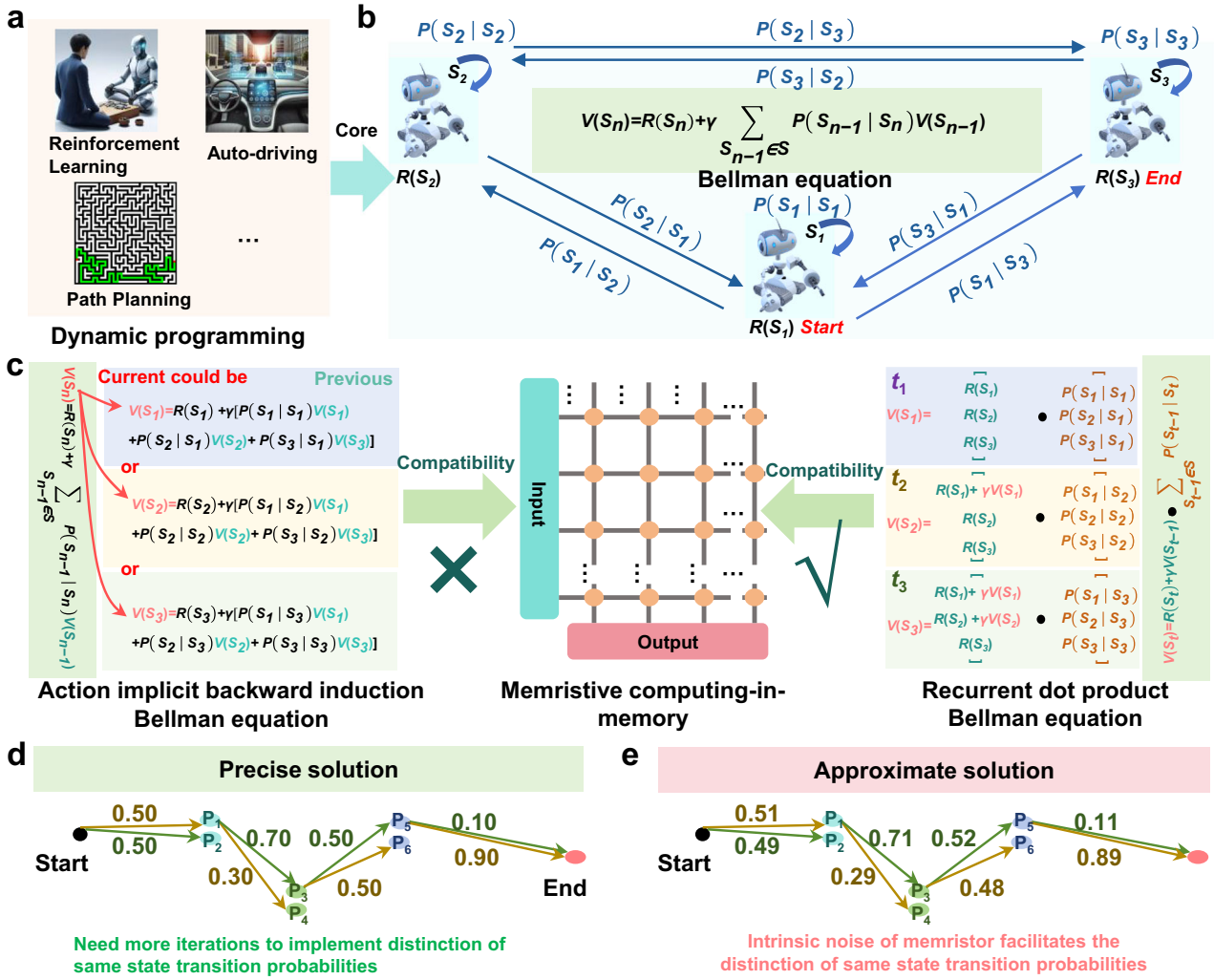

Fig. 1 | Challenges and solutions for realizing memristive Bellman solver. **a** Schematic of dynamic programming idea for various fields application, such as reinforcement learning, auto-driving and path planning etc. **b** A scene with three states (S_1, S_2, S_3) where each state can transition to any other state (including itself). The optimized value of each state could be obtained by solving the Bellman equation for decision-making. **c** The action implicit backward induction Bellman equation (without time dimension) for optimizing each state value in (**b**). Due to the absence of time dimension, $V(S_n)$ (red marked) may be indeterminate for any state. Therefore, although it seems compatible with MCIM, it cannot be determined at the time of hardware deployment whether the output corresponds to the subsequent input. A MBS for optimizing each state value in **b** realized by incorporating the temporal dimension and transforming the iterative solving process into recurrent dot product operations, facilitating the compatibility with MCIM (right panel). **d** In digital computing system, the Bellman solution process is a precise process. However, when there are same state transition probabilities, it will be necessary to conduct more iterations to achieve distinction. **e** The intrinsic noise of memristor would facilitates the distinction of same state transition probabilities, featuring approximate solution process and reducing iterations.

($[R(S_1)+\gamma V(S_1), R(S_2)+\gamma V(S_2), R(S_3)+\gamma V(S_3)]$) will be used as the initial reward for the next round solving process (iteration of MB_r operation to optimize the value of each state).

As illustrated in Fig. 2b, the results of adjacent MB_r operations are compared to identify whether the MBS has completed the solving process. The MBS completes the solving process, until the difference between two adjacent MB_r operation results is lower than a specific threshold (ε, 0.1). Otherwise, the previous MB_r operation result is used as the next initial input of MB_{dot} operation. The recurrent MB_{dot} operations and MB_r operations are performed to solve the Bellman equation. Furthermore, after the MBS solves the Bellman equation, the weight matrix (decision) is updated (optimized) by the ε-greedy rule[40]. Subsequently, the adjacent weight matrices are compared to determine whether the weight matrix has converged to a stable state (the decision is optimized to the optimal). The final weight matrix (the optimal decision) is obtained once the difference between two adjacent weight matrices (decisions) is lower than a specific threshold (τ, 0.1). Otherwise, the whole process is repeated. (The pseudo-code of the MBS is shown in Supplementary Note 3.).

As shown in Figs. 2c, S1 and S2, the simulation results indicate that the MBS can effectively reduce the number of iterations for value calculation across problems with varying state spaces, even when memristors operate at different precision levels. Additionally, the iteration number ($MB_{dot} \times MB_r$) could be further reduced by leveraging an approximate solution, as shown in Fig. 2d, e. During simulations, the approximate solution scheme assigns read noise to each weight state mapped to state transition probabilities, with this noise following a Gaussian distribution. This read noise can acts as a "random perturbation", introducing an approximate and sampling mechanism to the traditional precise solution scheme. This concept shares similarities with simulated annealing or stochastic gradient methods[37]. In each solving round, the read noise can help the state value achieve a random jump, prematurely exiting the repetitive and precise iterative process of detailed calculations, thereby achieving approximate convergence results with fewer iterations. Hence,

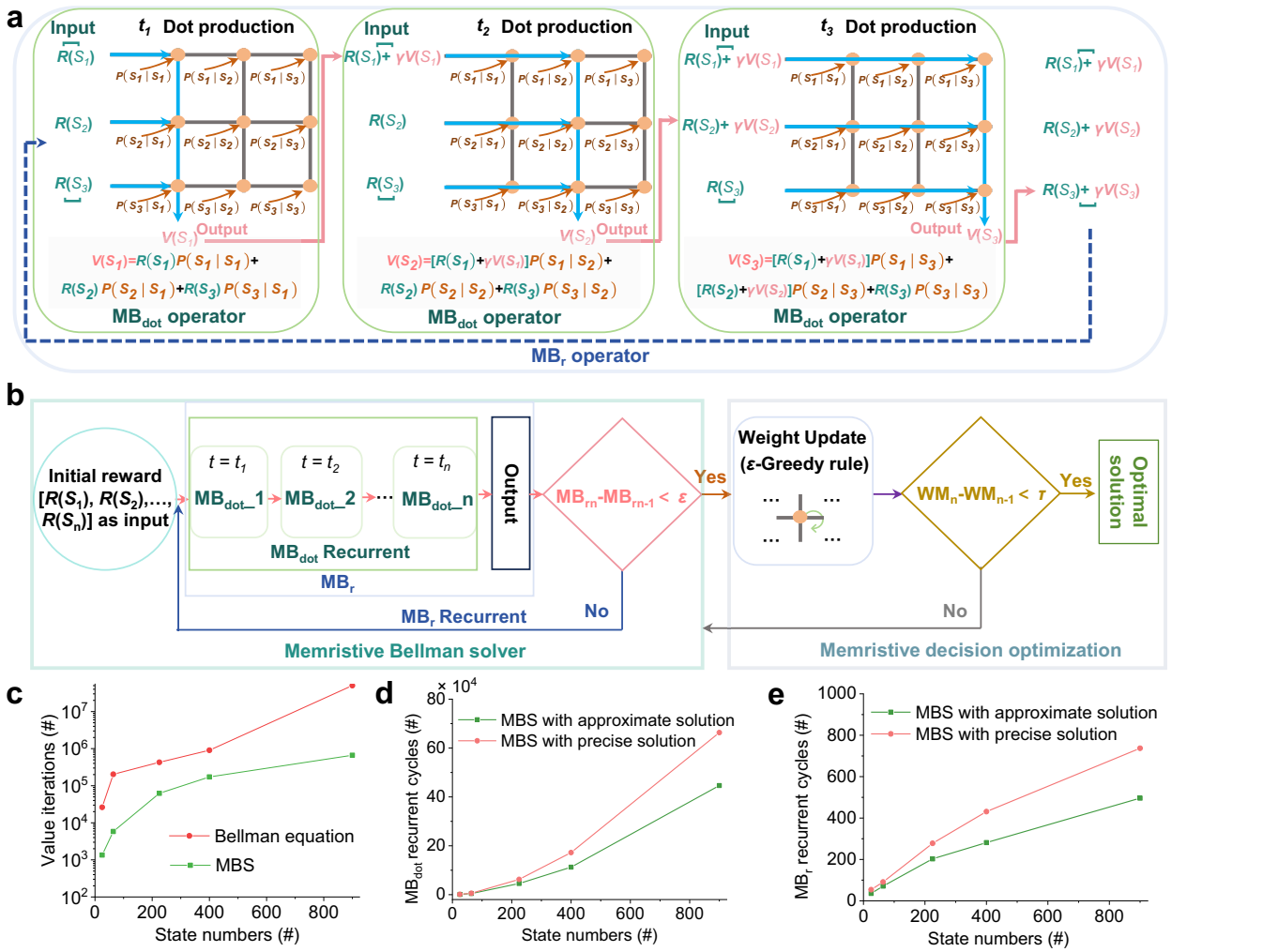

Fig. 2 | The work flows of memristive Bellman solver for decision-making. a The solving process of recurrent dot product Bellman equation for the application scene in Fig. 1b. Here, two operators are defined, that is, memristive Bellman dot operator (MB_{dot} operator) and memristive Bellman recurrent operator (MB_r operator). The blue arrows indicate that the current flows at each time step. **b** Memristive Bellman solver and memristive decision optimization. The Bellman solved by performance recurrent MB_{dot} and MB_r operations, until the difference between two adjacent MB_r operation results lower than a specific threshold (ε). After the Bellman equation solved by MBS, the weights (conductance states of memristor) would be updated according to ε-greedy rule to optimize the decision. The updated weights would be compared with previous weights until the difference is less than the threshold (τ), namely, the weights are approaching stability, meaning the decision optimization process finished. **c** The value iteration numbers variation with the state space. The comparison of the (**d**), MB_{dot} recurrent cycles and (**e**), MB_r recurrent cycles with approximate solution and precise solution.

considering the intrinsic read noise of the memristor, it is expected that the iteration number could be further reduced during hardware implementation. Additionally, it is worth noting that the proposed MBS is mainly suitable for scenarios where the state transition probability can be determined. When making decisions for scenarios where state transition strategies are difficult to determine, we can solve the problem based on the Q-network[12]. Furthermore, we can solve the loss value by embedding MBS into its training process and accelerating the training process.

Memristor for Bellman solver hardware implementation

For hardware implementation of the MBS (Fig. S3), a 1 kb resistive memristor array with a structure of 1 transistor 1 resistive memristor (1T1R) is adopted. The fabrication process of the memristor array aligns with our previous work[41]. Figure 3a_i shows the optical image of the fabricated memristor array. The 1T1R structure is confirmed with scanning transmission electron microscopy (STEM), as indicated in Fig. 3a_ii. Furthermore, the memristor cell marked by the red box in Fig. 3a_ii is zoomed, as shown in the Fig. 3a_iii. The memristor cell shows a TiN/TaO_x/HfO_x/TiN stack structure, confirmed by the energy dispersive spectrum (EDS) (Fig. S4). The memristor exhibits typical bipolar resistive switching characteristics (Figs. 3b, S5). When the top electrode is applied with a positive voltage sweep (0 → 1.75 V) and the gate terminal is biased at 1.55 V, the memristor switches from a high resistance state (HRS) to a low resistance state (LRS), i.e., SET process. Conversely, when the negative voltage sweep (0 → −1.85 V) stimulus is applied to the top electrode and the gate terminal biases with 4 V, the memristor switches from LRS to HRS, i.e., RESET process. Here, the conductance of the memristor is used to map the state transition probabilities. The transition probabilities between different states are different. Hence, the memristor needs to be programmed to different state levels. To verify the multi-level programmability, the memristor was programmed with a fixed positive voltage sweep (0 → 2.5 V) on the top electrode, varying the gate electrode bias (1.4 to 3 V, with a step of 0.05 V). The results indicate that the memristor can be reliably programmed to different state levels (Fig. 3c), which meets the demand for mapping the state transition probabilities. Furthermore, the typical long-term potentiation (LTP) and long-term depression (LTD) properties also verify the multi-level programmability of the memristor (Fig. S6).

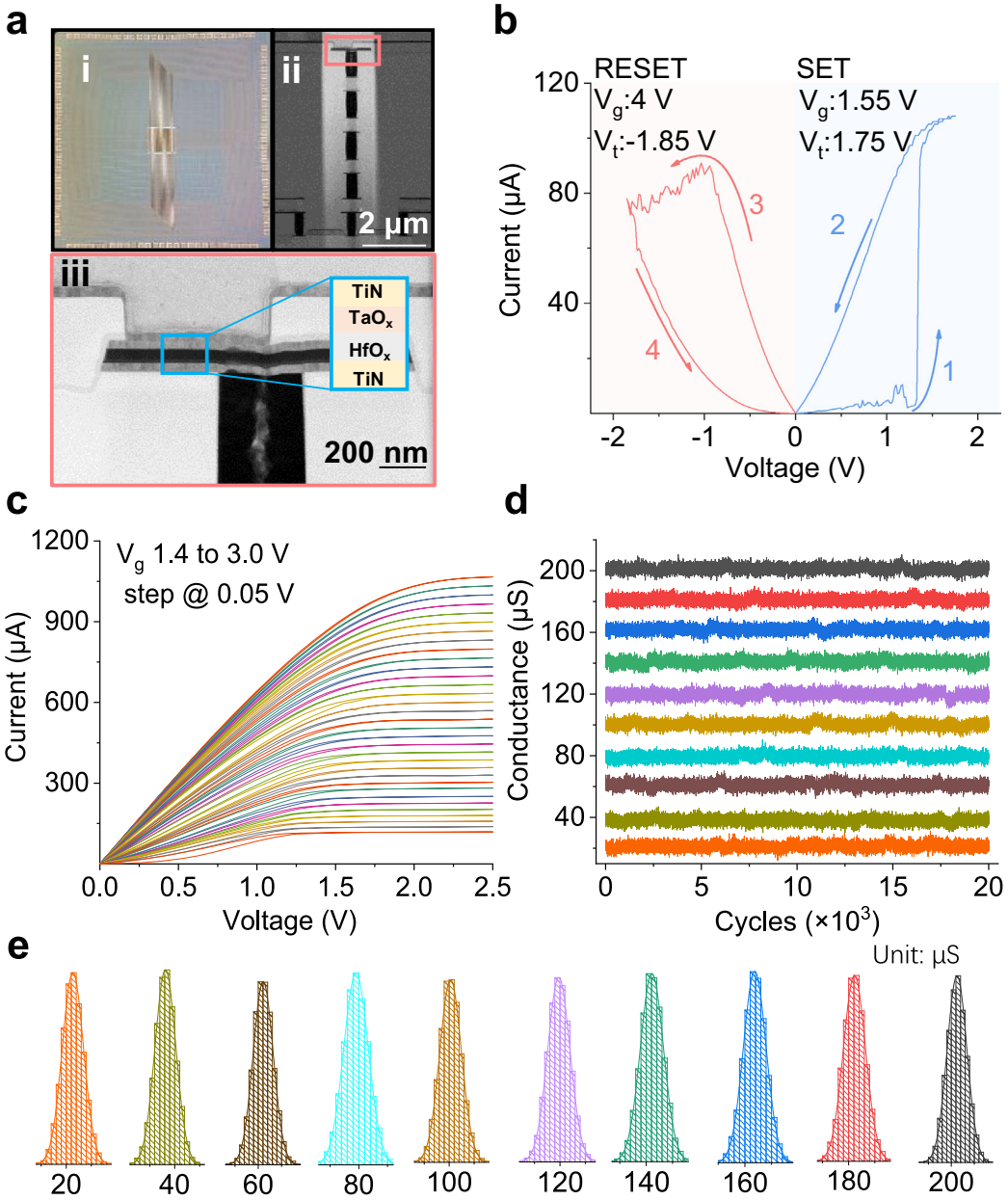

Fig. 3 | The characteristics of the TiN/TaO_x/HfO_x/TiN memristor for memristive Bellman solver implementation. a (i) The optical image of 1 kb memristor array. (ii) The STEM image of the 1T1R cell. (iii) The STEM image of the memristor cell. **b** The typical I-V curve of the memristor. **c** The multi-level program properties of the memristor, enabled by adjusting the gate voltage from 1.4 V to 3.0 V with a step of 0.05 V. **d** The read noise of memristor with 10 different conductance states. Each conductance is read with 20,000 times. **e** The distribution of the conductance obeys the Gaussian distribution.

It should be noted that, as illustrated in Fig. 2a, the MBS working process needs to perform recurrent MB_{dot} operations (i.e., read operations). Hence, the read noise would influence the solving effect, making the Bellman equation cannot be precisely solved. Whereas, theoretical derivation and simulation results indicate that the intrinsic noise ($\delta_{intrinsic}$) facilitates efficient convergence of the Bellman equation to an approximate optimal solution (see **Methods**). The intrinsic noise needs to meet the Gaussian distribution, and the Gaussian distribution of memristor conductance states has been demonstrated in various previous works[37,42]. Here, to investigate the read noise properties of the fabricated TiN/TaO_x/HfO_x/TiN memristor, the memristors were programmed to different conductance level and read for 20000 cycles (Fig. 3d). As shown in Fig. 3e, the read noise of each conductance level obeys a Gaussian distribution. The results indicated that the fabricated TiN/TaO_x/HfO_x/TiN memristor can be effectively utilized for hardware implementation of the MBS.

Memristive Bellman solver for path planning tasks

To investigate the feasibility of the MBS for practical decision-making applications, the path planning tasks are implemented with the MBS. In these tasks, the cost function is $Cost = \sum_{t=1}^{T} \gamma^{t-1} \boldsymbol{R}(S_t)$. Where $\boldsymbol{R}(S_t)$ is the reward (or cost) at state S_t, γ is the discount factor, and T is the total number of steps taken to reach the goal. First, a 5 × 5 maze path planning task is implemented (Fig. 4a). In this maze, the state_1, state_25 and state_12 is set as Start, End and Bonus, respectively. The state_13 and state_16 is set as Trap. Each state can perform four actions, namely, up, down, left, and right. The aim of this task is to plan the path from state_1 to state_25 through state_12. The path is realized by the transition between states. The initial decision (weight matrix) of the

https://doi.org/10.1038/s41467-025-60085-w

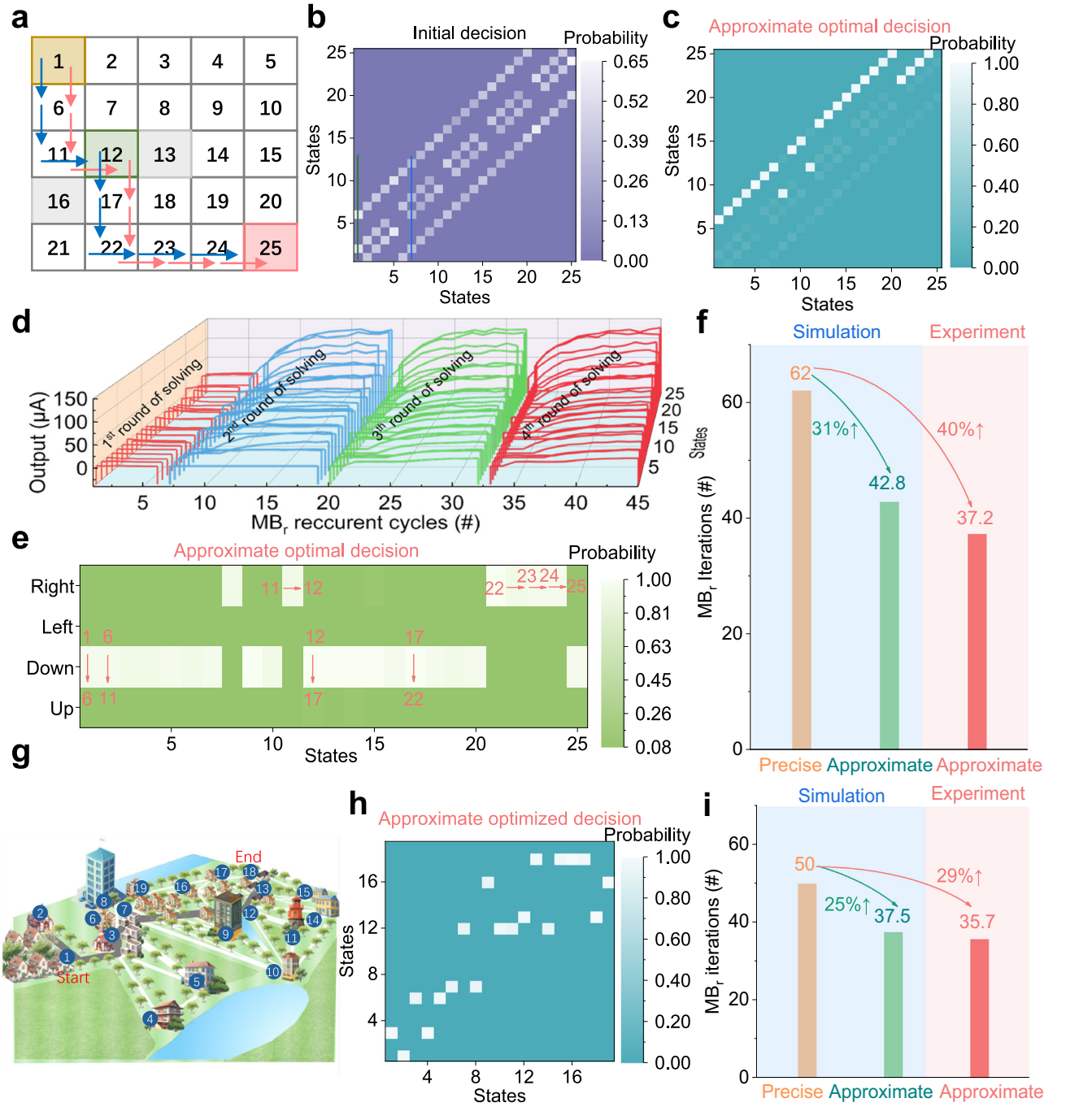

Fig. 4 | Memristive Bellman solver for path planning tasks. a Schematic of a 5 × 5 maze path planning scene, containing 25 states. In this maze, the state_1, state_25, and state_12 is set as Start, End, and Bonus, respectively. The state_13 and state_16 is set as Trap. Each state can perform four actions, namely, up, down, left, and right. **b** The initial decision (weight matrix) of the probability of transitions between states. **c** The approximate optimal decision (weight matrix) of the probability of transitions between states. **d** The value evolution tendency under the four times Bellman equation solving process. **e** The approximate optimal decision (weight matrix) for taking appropriate action (up, down, right, or left) of each state. **f** The comparison of the iteration (recurrent) times of the Bellman equation precise and approximate solving process. **g** Schematic of a constructed road mapping scene, containing 19 states. In this road map, the state_1 and state 18 is set as Start and End, respectively. Each state can only transmit to an adjacent state without obstacles (such as lake). **h** The approximate optimal decision (weight matrix) of the probability of transitions between states. **i** The comparison of the iteration (recurrent) times of the Bellman equation precise and approximate solving process.

probability of transitions between states is shown in Fig. 4b. The higher conductance of memristor represents the higher transition probability between the states. The results indicate that the path is hard to determine from the initial decision, owing to the transition probabilities between states are close. For example, the blue line in Fig. 4b indicates that the probability of state_7 transiting to state_2, state_6, state_8 and state_12 is almost same. Hence, the decision needs to be optimized by MBS. It should be noted that due to the intrinsic read noise of the memristor, for complex problem, the decision may cannot be optimized to the optimal decision. Fortunately, the read noise of the memristor obeys Gaussian distribution (Fig. 3e), which enables the MBS to obtain the approximate optimal decision (see **Methods**).

The results show that the decision needs to be optimized 4 times by the MBS to obtain the approximate optimal decision (Fig. 4c

and S7). The approximate optimal decision (weight matrix) of the probability of transitions between states is shown in Fig. 4c. The results indicate that each state has only one significant maximum probability of moving to the next state, which benefits the decision making. According to the approximate optimal decision, the path could be planned as the red arrows in Fig. 4a, which is same as the precise optimal decision (blue arrows in Fig. 4a). It should be noted that, as shown in Fig. S7, the path could be planned by only 2 times optimization. However, it still needs to be determined whether this path (strategy) is the optimal one or not. As shown in Fig. S8, after going through different rounds of optimization, the strategy keeps converging (the differences among the weights keep decreasing). It is not until the fourth round of optimization that the weights remained basically unchanged (i.e., reached convergence). Hence, the decision was optimized 4 times during the experiment. The reward and value evolution tendency under the optimization process is shown in Fig. 4d and S9–S10. In addition, the decision of each state to choose appropriate action is also optimized by the optimization process (Fig. S11). The approximate optimal decision (weight matrix) for taking appropriate action (up, down, right or left) of each state is shown in Fig. 4e. According to the state-action pairs decision, the path planning result is accordance with the state transition map, as the red arrows and state numbers illustrated in Fig. 4e. For the maze path planning task, the MBS shows obviously efficiency than traditional solving scheme. The (average) iteration numbers of 10 times the Bellman equation solving process for this path planning task is shown in Figs. 4f and S12. The results indicate that, by introducing noise, the approximated solving process could effectively reduce the iteration cycles, facilitating the decision-making efficiency.

In addition, a road mapping task (containing 19 states) is constructed to verify the effectiveness of MBS for decision-making (Fig. 4g). In this road map, the state_1 and state 18 is set as Start and End, respectively. Different from the maze path planning task, each state can take four fixed actions. In this road map, each state can only transmit to an adjacent state without obstacles (such as lake). After 4 times optimization, the approximate optimal decision is made by the MBS (Figs. 4h, S13–S15). The decision-making efficiency is also improved by adopting approximative solving process (Figs. 4i, S16). The results indicate that the MBS is suitable for various path planning scenes to make decisions. In addition, the estimation results indicate that the MBS shows obviously energy consumption advantages than GPU for both the tasks (Supplementary Note 2).

The simulation results indicate that the MBS effectively reduces the total number of value iterations for both tasks (Fig. S17). Furthermore, the experimental results based on the approximate solution are better than the simulation results. In the simulation process, only the read noise of the memristor has been considered. Whereas, during the whole decision-making process, the updating of weights is involved in the experiment, which introduces the write error (Fig. S18). We further simulated the write error influence on the iteration speed (Fig. S19). The results indicate that the write error could facilitate the iteration speed in a certain range. Hence, this may be due to the existence of the write error in the weight update process of the experiment, which makes the difference in weights more obvious and achieves the optimal decision faster. For example, according to the initial decision, the probability (weight) of state_1 transitioning to state_2 and state_6 is almost the same (0.5), as the green line indicated in Fig. 4b). During the decision optimization process, the probability (weight) updates according to ε-greedy rule. However, the transition of state_1 to state_2 or state_6 can both satisfy the demand of decision optimization. Therefore, in the process of simulation, the probability will be equal. However, there is a write error during the experiment, and the transfer of state_1 to state_2 or state_6 will cause a distinction, which can speed up the whole optimization process, facilitating the decision-making efficiency.

In summary, we propose an MBS that benefits from both software and hardware co-optimization. On the software side, we have integrated the temporal dimension and converted the iterative double expectations into recurrent dot product operations, which streamline the implementation of the Bellman equation using MCIM technology and reduce its computation complexity (reduced iterations). In fact, it should be noted that the operation of the recurrent dot product can be adapted to any type of memory device based on CIM technologies. On the hardware side, we leverage the inherent noise characteristics of memristors to enable approximate solutions instead of precise ones, further reducing iterations. We validated the proposed MBS through path planning tasks. Theoretical analysis and experimental results show that the MBS significantly reduces the number of iterations required, enhancing solving efficiency. This approach offers a promising solution for developing more efficient dynamic decision-making systems.

Methods

Recurrent dot product Bellman equation

The Bellman equation is a fundamental equation in dynamic programming and reinforcement learning, used to compute the value function, which represents the expected cumulative reward starting from state S. The Bellman equation is expressed as:

$$\mathbf{V}(S_n) = \mathbf{R}(S_n) + \gamma \sum_{S_{n-1} \in \mathbf{S}} \mathbf{P}(S_{n-1}|S_n)\mathbf{V}(S_{n-1}) \tag{1}$$

Here, the $\mathbf{V}(S_n)$ is the value function of the current state S_n, $\mathbf{R}(S_n)$ is current reward function, γ is a discount constant, $\mathbf{P}(S_{n-1}|S_n)$ is the state transition probability from previous states to current state, $\mathbf{S}$ is the set of all possible states (state space), S_n represents the current state and S_{n-1} represents previous states.

This equation is iterative, meaning it requires repeated calculations to converge to the optimal value function. However, this iterative nature makes it challenging to implement efficiently on MCIM systems. To make the Bellman equation compatible with MCIM systems, we introduce a time dimension to describe the relationship between the current state and previous states. Let S_t and S_{t-1} represent the state at time t and t–1, respectively. The Bellman equation can then be rewritten as:

$$\mathbf{V}(S_t) = \mathbf{R}(S_t) + \gamma \sum_{S_{n-1} \in \mathbf{S}} \mathbf{P}(S_{n-1}|S_t)\mathbf{V}(S_{n-1}) \tag{2}$$

Here, S_{n-1} represents all possible previous states, while S_{t-1} represents the state at the previous time t–1. Hence, in equation (2), S_{n-1} cannot be directly replaced by S_{t-1} because S_{n-1} refers to all previous states, whereas S_{t-1} refers to a single state at time $t-1$.

To simplify the equation, we make two key assumptions based on the temporal difference (TD) algorithm and the local property of the state space[43,44].

Temporal smoothness assumption. When the learning process is close to convergence, the value function at two adjacent time steps t and $t-1$ is approximately equal. That is:

$$\mathbf{V}(S_t) \approx \mathbf{V}(S_{t-1}) \tag{3}$$

Local consistency assumption. In a local region of the state space, the value function is approximately constant. This means:

$$\mathbf{V}(S_t) \approx \mathbf{V}(S_{n-1}) \tag{4}$$

for all S_{n-1} in the neighborhood of S_t.

Combining these two assumptions, we obtain that:

$$\mathbf{V}(S_{t-1}) \approx \mathbf{V}(S_{n-1}) \tag{5}$$

Hence, the (2) could be written as:

$$\mathbf{V}(S_t) = \mathbf{R}(S_t) + \gamma \sum_{S_{n-1} \in \mathbf{S}} \mathbf{P}(S_{n-1}|S_t)\mathbf{V}(S_{t-1}) \tag{6}$$

Since the sum of probabilities over all possible transitions is equal to 1 (i.e., $\sum_{S_{n-1} \in \mathbf{S}} \mathbf{P}(S_{n-1}|S_t) = 1$, the equation simplifies to:

$$\mathbf{V}(S_t) = \mathbf{R}(S_t) + \gamma \mathbf{V}(S_{t-1}) \tag{7}$$

To reintroduce the state transition probabilities, we express (7) as:

$$\mathbf{V}(S_t) = [\mathbf{R}(S_t) + \gamma \mathbf{V}(S_{t-1})] \sum_{S_{t-1} \in \mathbf{S}} \mathbf{P}(S_{t-1}|\mathbf{S}) \tag{8}$$

Here, the **S** represents the whole state space. Each time, the state could be one state of the whole state space. Hence, this can be further rewritten using the dot product notation:

$$\mathbf{V}(S_t) = [\mathbf{R}(S_t) + \gamma \mathbf{V}(S_{t-1})] \sum_{S_{t-1} \in \mathbf{S}} \mathbf{P}(S_{t-1}|S_t) \tag{9}$$

In summary, by introducing the time dimension and making the above approximations, we transform the original iterative Bellman equation into a recurrent dot product form, which facilitates its solving by MCIM technology.

Additionally, the assumptions (a) and (b) would introduce error in the original Bellman equation, we need to further discuss the applicable conditions of the recurrent dot product Bellman equation.

The error from assumption (a) must be bounded:

$$\left|\mathbf{V}(S_t) - \mathbf{V}(S_{t-1})\right| \le \delta_t \tag{10}$$

Where δ_t is a small positive constant (tolerance for temporal variation).

The error from assumption (b) must be bounded:

$$\left|\mathbf{V}(S_{n-1}) - \mathbf{V}(S_{t-1})\right| \le \delta_s, \forall S_{n-1} \text{ where } \mathbf{P}(S_{n-1}|S_t) > 0, \text{ and } \sum_{S_{n-1} \in \mathbf{S}} \mathbf{P}(S_{n-1}|S_t) = 1 \tag{11}$$

Where δ_s is a small positive constant (tolerance for temporal variation).

Hence, the error introduced by these assumptions is $|(2)-(7)|$:

$$|\mathrm{E}| = \left|\gamma[\sum_{S_{n-1} \in \mathbf{S}} \mathbf{P}(S_{n-1}|S_t)\mathbf{V}(S_{n-1}) - \mathbf{V}(S_{t-1})]\right| = \left|\gamma[\mathbf{V}(S_{n-1}) - \mathbf{V}(S_{t-1})]\right| \le \gamma\delta_s \tag{12}$$

In order to ensure that (2) and (7) are basically equivalent, then:

$$|\mathrm{E}| \ll \left|\mathbf{V}(S_t)\right| \tag{13}$$

That is:

$$\gamma\delta_s \ll \left|\mathbf{V}(S_t)\right| \tag{14}$$

The (14) ensures that the error induced by local consistency assumption is negligible. The error induced by temporal smoothness assumption should also be negligible. Hence, the applicable conditions of recurrent dot product Bellman equation are:

$$\max(\gamma\delta_s, \delta_t) \ll \left|\mathbf{V}(S_t)\right| \tag{15}$$

Proof the convergence of recurrent dot Bellman equation

To proof the convergence of recurrent dot Bellman equation, we want to introduce the Banach fixed-point theorem firstly. The Banach fixed-point theorem (also known as the contraction mapping theorem) states that if a function **G** is a contraction mapping on a complete metric space, then **G** has a unique fixed point[45]. This fixed point can be found by iteratively applying **G** to any initial point in the space. According to the Banach fixed-point theorem, we can proof that the recurrent dot Bellman equation converges to a unique solution, provided that the mapping **G** defined by the recurrent dot Bellman equation is a contraction.

We define the mapping **G** as follows:

$$\mathbf{G}(\mathbf{V}(S_{t-1})) = \mathbf{V}(S_t) \tag{16}$$

Here, **G** takes the value function $\mathbf{V}(S_{t-1})$ at time $t-1$ and maps it to the value function $\mathbf{V}(S_t)$ at time t. Using the recurrent dot Bellman equation, **G** can be expressed as:

$$\mathbf{G}(\mathbf{V}(S_{t-1})) = [\mathbf{R}(S_t) + \gamma \mathbf{V}(S_{t-1})] \cdot \sum_{S_{t-1} \in \mathbf{S}} \mathbf{P}(S_{t-1}|S_t) \tag{17}$$

The goal is to prove that **G** is a contraction mapping, which will ensure that the recurrent dot Bellman equation converges to a unique solution. The mapping **G** is contraction if there exists a constant γ (with $0 < \gamma < 1$) such that for any two value functions **V** and **W**:

$$\left|\left|\mathbf{G}(\mathbf{V}) - \mathbf{G}(\mathbf{W})\right|\right|_\infty \le \gamma ||\mathbf{V} - \mathbf{W}||_\infty \tag{18}$$

Here, the $||\cdots||_\infty$ is infinity norm. The infinity norm of a vector $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ is defined as:

$$||\mathbf{X}||_\infty = \max\{|x_1|, |x_2|, \ldots, |x_n|\} \tag{19}$$

To prove that **G** is a contraction mapping, we start by considering two value functions **V** and **W**. Using the definition of **G** in Eq. (16), we compute **G**(**V**)-**G**(**W**)

$$\mathbf{G}(\mathbf{V}) - \mathbf{G}(\mathbf{W}) = [\mathbf{R}(S_t) + \gamma \mathbf{V}(S_{t-1})] \cdot \sum_{S_{t-1} \in \mathbf{S}} \mathbf{P}(S_{t-1}|S_t) - [\mathbf{R}(S_t) + \gamma \mathbf{W}(S_{t-1})] \cdot \sum_{S_{t-1} \in \mathbf{S}} \mathbf{P}(S_{t-1}|S_t) \tag{20}$$

Simplifying, we get:

$$\mathbf{G}(\mathbf{V}) - \mathbf{G}(\mathbf{W}) = \gamma[\mathbf{V}(S_{t-1})] - \mathbf{W}(S_{t-1})] \cdot \sum_{S_{t-1} \in \mathbf{S}} \mathbf{P}(S_{t-1}|S_t) \tag{21}$$

Similarly, for traditional Bellman equation, there is:

$$\mathbf{G}(\mathbf{V}) - \mathbf{G}(\mathbf{W}) = \gamma \sum_{S_{n-1} \in \mathbf{S}} \mathbf{P}(S_{n-1}|S_n)[\mathbf{V}(S_{n-1})] - \mathbf{W}(S_{n-1})] \tag{22}$$

Next, we apply the infinity norm $||\cdots||_\infty$ to both sides of (21) and (22). we obtain:

$$\begin{aligned} \left|\left|\mathbf{G}(\mathbf{V}) - \mathbf{G}(\mathbf{W})\right|\right|_\infty &= \left|\left|\gamma[\mathbf{V}(S_{t-1})] - \mathbf{W}(S_{t-1})] \cdot \sum_{S_{t-1} \in \mathbf{S}} \mathbf{P}(S_{t-1}|S_t)\right|\right|_\infty \\ &\le \gamma \left|\left|\sum_{S_{t-1} \in \mathbf{S}} \mathbf{P}(S_{t-1}|S_t)\right|\right|_\infty \cdot ||\mathbf{V} - \mathbf{W}||_\infty \end{aligned} \tag{23}$$

$$\left|\left|\mathbf{G}(\mathbf{V})-\mathbf{G}(\mathbf{W})\right|\right|_\infty = \left|\left|\gamma \sum_{S_{n-1}\in\mathbf{S}} \mathbf{P}(S_{n-1}|S_n)[\mathbf{V}(S_{n-1})]-\mathbf{W}(S_{n-1})\right]\right|\right|_\infty \le \gamma||\mathbf{V}-\mathbf{W}||_\infty \quad (24)$$

Here, $||\sum_{S_{t-1}\in\mathbf{S}}\mathbf{P}(S_{t-1}|S_t)||_\infty \le 1$, Hence, for both of recurrent dot Bellman equation and traditional Bellman equation, the **G** is a contraction mapping.

To apply the Banach fixed-point theorem, we must also show that the state space **S** is a complete metric space. A metric space is complete if every Cauchy sequence in the space converges to a point within the space.

Let $\mathbf{S} = \{S_1, S_2, \ldots, S_t\}$ be a finite set of states.

Define a metric **d** on **S** as:

$$\mathbf{d}(S_i, S_j) = \left|\left|\mathbf{V}(S_i)-\mathbf{V}(S_j)\right|\right|_\infty \quad (25)$$

Since **S** is finite, any Cauchy sequence $\{S_n\}$ in **S** must eventually repeat some state S^* infinitely often. This ensures that $\{S_n\}$converges to S^*[46]. That is the state space **S** is a complete metric space.

Since **G** is a contraction mapping and the state space **S** is complete, the Banach fixed-point theorem guarantees that **G** has a unissque fixed point V^*, such that $\mathbf{G}(V^*) = V^*$. This proves that the recurrent dot Bellman equation converges to a unique solution. Furthermore, it should be noted that, here

$$\gamma\left|\left|\sum_{S_{t-1}\in\mathbf{S}}\mathbf{P}(S_{t-1}|S_t)\right|\right|_\infty \cdot ||\mathbf{V}-\mathbf{W}||_\infty \le \gamma||\mathbf{V}-\mathbf{W}||_\infty \quad (26)$$

Hence, the recurrent dot Bellman equation convergency speed is faster than traditional Bellman equation.

Recurrent dot product Bellman equation approximate solution

In the memristive recurrent dot Bellman equation, the state transition probability $\mathbf{P}(S_{t-1}|S_t)$ is mapped to the conductance states of memristors. However, memristors exhibit intrinsic read noise ($\delta_{intrinsic}$), making it difficult to achieve precise solutions. Instead of treating this noise as a drawback, we leverage it to find an approximate solution to the Bellman equation. By introducing the $\delta_{\text{intrinsic}}$, the state transition probability could be replaced by $\boldsymbol{\sigma}(S_{t-1}|S_t)$:

$$\boldsymbol{\sigma}(S_{t-1}|S_t) = \mathbf{P}(S_{t-1}|S_t) + \delta_{intrinsic} \quad (27)$$

Then the recurrent dot product Bellman equation becomes:

$$\mathbf{V}(S_t) = [\mathbf{R}(S_t) + \gamma\mathbf{V}(S_{t-1})] \cdot \sum_{S_{t-1}\in\mathbf{S}} \boldsymbol{\sigma}(S_{t-1}|S_t) \quad (28)$$

The sum $\sum_{S_{t-1}\in\mathbf{S}}\boldsymbol{\sigma}(S_{t-1}|S_t)$ includes the intrinsic noise, that is:

$$\sum_{S_{t-1}\in\mathbf{S}} \boldsymbol{\sigma}(S_{t-1}|S_t) = \sum_{S_{t-1}\in\mathbf{S}} \mathbf{P}(S_{t-1}|S_t) + \sum_{S_{t-1}\in\mathbf{S}} \delta_{intrinsic} \quad (29)$$

Since $\sum_{S_{t-1}\in\mathbf{S}}\boldsymbol{\sigma}(S_{t-1}|S_t) = 1$ (by definition of probability), we have:

$$\sum_{S_{t-1}\in\mathbf{S}} \boldsymbol{\sigma}(S_{t-1}|S_t) = 1 + \sum_{S_{t-1}\in\mathbf{S}} \delta_{intrinsic} \quad (30)$$

The intrinsic read noise of memristor follows Gaussian distribution, which has been reported in various previous works[37,42], and tested in our measurement results. That is:

$$\delta_{intrinsic} \sim N(0, \varphi^2) \quad (31)$$

Here, the φ^2 is the read noise variance. We set $|S|$ as the number of states. Therefore, $\sum_{S_{t-1}\in\mathbf{S}}\delta_{intrinsic}$ is equivalent to adding $\delta intrinsic\,|\,S|$ times. When summing $|S|$ independent Gaussian variables, the resulting sum is also Gaussian. That is:

$$\sum_{S_{t-1}\in\mathbf{S}} \delta_{intrinsic} \sim N(0, |S|\varphi^2) \quad (32)$$

Here, $|S|\varphi^2$ is the total variance of the sum, which scales linearly with the number of terms $|S|$.

Hence,

$$\sum_{S_{t-1}\in\mathbf{S}} \boldsymbol{\sigma}(S_{t-1}|S_t) \sim N(1, |S|\varphi^2) \quad (33)$$

Using *Chebyshev's* inequality, we can show that the noise is bounded with high probability. Specifically, there exists a constant K (with $0 < K \le 1$) such that:

$$\left|\left|\sum_{S_{t-1}\in\mathbf{S}} \boldsymbol{\sigma}(S_{t-1}|S_t)\right|\right|_\infty \le \mathrm{K} \quad (34)$$

Then, we define a mapping **F** as:

$$\mathbf{F}(\mathbf{V}(S_{t-1})) = \mathbf{V}(S_t) \quad (35)$$

Similarly, we could obtain:

$$\left|\left|\mathbf{F}(\mathbf{V})-\mathbf{F}(\mathbf{W})\right|\right|_\infty \le \gamma K\left|\left|\sum_{S_{t-1}\in\mathbf{S}}\mathbf{P}(S_{t-1}|S_t)\right|\right|_\infty \cdot ||\mathbf{V}-\mathbf{W}||_\infty \quad (36)$$

Causing $0 < \gamma < 1$ and $0 < K \le 1$, $0 < \gamma K < 1$. That is, **F** is a compression mapping, and the recurrent dot product Bellman equation converges to an approximate solution.

Furthermore, during the approximate solution process, the contraction factor is reduced from γ to γK. This indicates that the approximate solution is faster (at least not slower) than the precise solution.

Programming protocol of the write-and-verify strategy for weight update

The programming protocol of the write-and-verify strategy for weight update is shown in Fig S20, schematically. When update the weights, the device is reset to high resistance state firstly. Then the SET pulse is applied on the device according to the target. If the device resistance wrote to the acceptive range, the write operation is success. If the resistance is high, then the gate voltage is increased and then another SET pulse is applied to decrease its resistance. If the resistance is lower than the lowest resistance of the acceptance range, or the write-and-verify operation reaches a limit, then the write is a failure.

Data availability

The data that support the findings of this study are provided as a Source Data file with this paper. Source data are provided with this paper.

Code availability

The codes are available at https://github.com/495008566/MBS.git.

References

1. Roozegar, M., Mahjoob, M. J. & Jahromi, M. Optimal motion planning and control of a nonholonomic spherical robot using dynamic programming approach: simulation and experimental results. *Mechatronics* **39**, 174–184 (2016).

2. Zhou, B., Gao, F., Wang, L., Liu, C. & Shen, S. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robot. Autom. Lett.* **4**, 3529–3536 (2019).
3. Mitten, L. Preference order dynamic programming. *Manag. Sci.* **21**, 43–46 (1974).
4. Bellman, R. Dynamic programming. *science* **153**, 34–37 (1966).
5. Lee, J. & Lee, J. Approximate dynamic programming based approach to process control and scheduling. *Comput. Chem. Eng.* **30**, 1603–1618 (2006).
6. Gast, N., Gaujal, B. & Le Boudec, J. Y. Mean field for Markov decision processes: from discrete to continuous optimization. *IEEE Trans. Autom. Control* **57**, 2266–2280 (2012).
7. Powell, W. B. A unified framework for optimization under uncertainty. *INFORMS TutORials in Operations Research*, 45–83. (2016).
8. Guo, X., Song, X. & Zhang, Y. First passage optimality for continuous-time Markov decision processes with varying discount factors and history-dependent policies. *IEEE Trans. Autom. Control* **59**, 163–174 (2013).
9. Abe, N., Biermann, A. & Long, P. Reinforcement learning with immediate rewards and linear hypotheses. *Algorithmica* **37**, 263–293 (2003).
10. Lutter, M. Continuous-time fitted value iteration for robust policies. In *Inductive Biases in Machine Learning for Robotics and Control*, **156**, 71–111. (2023).
11. Boybat, I. et al. Neuromorphic computing with multi-memristive synapses. *Nat. Commun.* **9**, 2514 (2018).
12. Wang, Z. et al. Reinforcement learning with analogue memristor arrays. *Nat. Electron.* **2**, 115–124 (2019).
13. Wu, Z. et al. A habituation sensory nervous system with memristors. *Adv. Mater.* **32**, 2004398 (2020).
14. Joshi, V. et al. Accurate deep neural network inference using computational phase-change memory. *Nat. Commun.* **11**, 2473 (2020).
15. Le Gallo, M. et al. A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *Nat. Electron.* **6**, 680–693 (2023).
16. Lim, D. H. et al. Spontaneous sparse learning for PCM-based memristor neural networks. *Nat. Commun.* **12**, 319 (2021).
17. Berdan, R. et al. Low-power linear computation using nonlinear ferroelectric tunnel junction memristors. *Nat. Electron.* **3**, 259–266 (2020).
18. Kim, I. J., Kim, M. K. & Lee, J. S. Highly-scaled and fully-integrated 3-dimensional ferroelectric transistor array for hardware implementation of neural networks. *Nat. Commun.* **14**, 504 (2023).
19. Kim, M. K., Kim, I. J. & Lee, J. S. CMOS-compatible compute-in-memory accelerators based on integrated ferroelectric synaptic arrays for convolution neural networks. *Sci. Adv.* **8**, eabm8537 (2022).
20. Papp, Á, Porod, W. & Csaba, G. Nanoscale neural network using non-linear spin-wave interference. *Nat. Commun.* **12**, 6422 (2021).
21. Kaiser, J. et al. Hardware-aware in situ learning based on stochastic magnetic tunnel junctions. *Phys. Rev. Appl.* **17**, 014016 (2022).
22. Zahedinejad, M. et al. Memristive control of mutual spin Hall nano-oscillator synchronization for neuromorphic computing. *Nat. Mater.* **21**, 81–87 (2022).
23. Hao, Y. et al. Uniform, fast, and reliable CMOS compatible resistive switching memory. *J. Semicond.* **43**, 054102 (2022).
24. Shi, T. et al. A review of resistive switching devices: performance improvement, characterization, and applications. *Small Struct.* **2**, 2000109 (2021).
25. Zhang, Y. et al. Evolution of the conductive filament system in HfO2-based memristors observed by direct atomic-scale imaging. *Nat. Commun.* **12**, 7232 (2021).
26. Milano, G. et al. In materia reservoir computing with a fully memristive architecture based on self-organizing nanowire networks. *Nat. Mater.* **21**, 195–202 (2022).
27. Rao, M. et al. Thousands of conductance levels in memristors integrated on CMOS. *Nature* **615**, 823–829 (2023).
28. Onen, M. et al. Nanosecond protonic programmable resistors for analog deep learning. *Science* **377**, 539–543 (2022).
29. Li, C. et al. Analogue signal and image processing with large memristor crossbars. *Nat. Electron.* **1**, 52–59 (2018).
30. Lin, P. et al. Three-dimensional memristor circuits as complex neural networks. *Nat. Electron.* **3**, 225–232 (2020).
31. Yao, P. et al. Face classification using electronic synapses. *Nat. Commun.* **8**, 15199 (2017).
32. Dalgaty, T. et al. In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling. *Nat. Electron.* **4**, 151–161 (2021).
33. Barto, A. & Mahadevan, S. Recent Advances in Hierarchical Reinforcement Learning. *Discret. Event Dyn. Syst.* **13**, 341–379 (2003).
34. Rust, J. Using randomization to break the curse of dimensionality. *Econometrica: J. Econometr. Soc.*, 487–516 (1997).
35. Ormoneit, D. & Sen, Ś Kernel-based reinforcement learning. *Mach. Learn.* **49**, 161–178 (2002).
36. Mahmoodi, M. R., Prezioso, M. & Strukov, D. B. Versatile stochastic dot product circuits based on nonvolatile memories for high performance neurocomputing and neurooptimization. *Nat. Commun.* **10**, 5113 (2019).
37. Cai, F. et al. Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks. *Nat. Electron.* **3**, 409–418 (2020).
38. Sun, Z., Pedretti, G., Ambrosi, E., Bricalli, A. & Ielmini, D. In-memory Eigenvector Computation in Time O (1). *Adv. Intell. Syst.* **2**, 2000042 (2020).
39. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).
40. Zhu, R., Zhao, Y. Q., Chen, G., Ma, S. & Zhao, H. Greedy outcome weighted tree learning of optimal personalized treatment rules. *Biometrics* **73**, 391–400 (2017).
41. Lin, H. et al. Implementation of highly reliable and energy efficient in-memory hamming distance computations in 1 Kb 1-Transistor-1-Memristor arrays. *Adv. Mater. Technol.* **6**, 2100745 (2021).
42. Lu, J. et al. Quantitatively evaluating the effect of read noise in memristive Hopfield network on solving traveling salesman problem. *IEEE Electron. Device Lett.* **41**, 1688–1691 (2020).
43. Bradtke, S. & Barto, A. Linear least-squares algorithms for temporal difference learning. *Mach. Learn.* **22**, 33–57 (1996).
44. Cervellera, C., Gaggero, M. & Macciò, D. Low-discrepancy sampling for approximate dynamic programming with local approximators. *Comput. Oper. Res.* **43**, 108–115 (2014).
45. Rincón-Zapatero, J. P. & Rodríguez-Palmero, C. Existence and uniqueness of solutions to the Bellman equation in the unbounded case. *Econometrica* **71**, 1519–1555 (2003).
46. Rudin, W. Principles of Mathematical Analysis, 3rd edn. (McGraw-Hill, 1976).

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant Nos. 62304001, 62274002 and 62201005, in part by the Anhui Provincial Natural Science Foundation under Grant No 2308085QF213, in part by the Natural Science Research Project of Anhui Educational Committee under Grant No 2023AH050072.

Author contributions

Z. Wu, Y. Dai and Q. Liu directed the research. Z. Feng, Z. Wu and J. Zou conceptualize the memristive Bellman solve and related applications. Z. Feng, Z. Wu and L. Cheng completes the device characterization. Z. Feng, and J. Zou completes system construction and test. Z. Feng,

Z. Wu and H. Wang completes the theoretical derivation. X. Zhao, X. Zhang, C. Wang, J. Lu, Y. Wang, W. Guo, Z. Qian, Y. Zhu, and Z. Xu helped with data analysis. All authors reviewed and edited the paper.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41467-025-60085-w.

Correspondence and requests for materials should be addressed to Zuheng Wu, Yuehua Dai or Qi Liu.

Peer review information *Nature Communications* thanks the anonymous, reviewer(s) for their contribution to the peer review of this work. A peer review file is available.