aggregation over several $1 \times d$ matrixes corresponding to $\vec{I}_i$. For non-terminal states, the counterfactual values or baseline-corrected counterfactual values can be updated based on the summation or aggregation as shown in the lower (or second) equation of Eq. (18). By contrast, for the variance reduction techniques with the expected utility value baseline, the counterfactual values or baseline-corrected counterfactual values are updated based on multiplication (e.g., as shown in Eq. (4)) for all terminal and non-terminal states. As such, the computational load saved by the counterfactual value baseline relative to expected utility value baseline can depend on a depth of and/or a number of non-terminal states in the game tree or public tree that represents the environment or the IIG. The MCCFR with counterfactual value baseline is even more computationally efficient than the ones based on expected utility value baseline, if the game tree or public tree is deep and/or has a large number of non-terminal states. As an example, in HUNL, $d = 1326$ [4]. The expected-value-based method needs to conduct at least 1326×1326 add operations to update its baseline while the counterfactual-value-based method only needs 1×1326 add operations.

[0117] In some embodiments, exploration techniques can be applied to MCCFR to achieve better performance with fewer samples. In some embodiments, a hybrid or mixture sampling policy can be used to balance exploitation and exploration, which are trade-offs in MCCFR that learn state-action policy from the sampling experience. In some embodiments, the hybrid sampling policy can be represented by:

$$\sigma_i^{se}(a|I_i) = (1-\alpha) * \sigma_i^{s}(a|I_i) + \alpha * \sigma_i^{e}(a|I_i), \quad (20)$$

where $\sigma_i^s(a|I_i)$ refers to a sampling policy, $\sigma_i^e(a|I_i)$ refers to an exploration policy. $\alpha \in [0,1]$ refers to the mixture factor, which is used to control the weight of exploration. Typically, α is a decay factor. For example, set $\alpha = \frac{1}{\ln(t+10)}$, who holds $\lim_{t\to\infty} \alpha = 0$. The sampling policy σ_i^s, can be any suitable sampling policy including RCS σ_i^{rcs}, MCS σ_i^{mcs}, WAS σ_i^{was}, outcome sampling, external sampling, robust sampling, or uniform sampling. In some embodiments, both σ_i^s and σ_i^e holds $\sum_{a\in A(I_i)} \sigma_i^s(a|I_i) = 1$, $\sum_{a\in A(I_i)} \sigma_i^e(a|I_i) = 1$. Therefore, σ_i^s holds $\sum_{a\in A(I_i)} \sigma_i^{se}(a|I_i) = 1$.

[0118] Define $\Delta C^t(a|I_i)$ as the sampling times for action a at infoset I_i in iteration t. If the infoset I_i or action a is not sampled in this iteration, the $\Delta C^t(a|I_i)$ is 0. The cumulative sampling times can be computed by

$$C^t(a|I_i) = \sum_{j=1}^{t} \Delta C^j\,(a|I_i), \text{ (21).}$$

[0119] In value-form MCCFR, such as outcome sampling, if the action a is sampled at infoset I_i in iteration t, set $\Delta C^t(a|I_i) = 1$. In vector-form MCCFR, when $\vec{I}_i$ is sampled, $\Delta C^t(a|I_i)$ for each infoset $I_i \in \vec{I}_i$ should be updated accordingly. In some embodiments, a single counter is used for the entire vector of information set $\vec{I}_i$ for calculating the times of the action a is sampled. In some embodiments, a mini-batch MCCFR (which is described in PCT App. No. PCT/CN2019/072200, filed on 1/17/2019, entitled "SAMPLING SCHEMES FOR STRATEGY SEARCHING IN STRATEGIC INTERACTION BETWEEN PARTIES" and in U.S. App. No. 16/448,390, filed on 6/21/2019, entitled "SAMPLING SCHEMES FOR STRATEGY SEARCHING IN STRATEGIC INTERACTION BETWEEN PARTIES.") is used, $\Delta C^t(a|I_i)$ could be larger than 1 because a mini-batch of blocks are sampled in one iteration. The exploration policy can be computed by

$$\sigma_i^{e,t}(a|I_i) = \frac{(1+\frac{\beta}{\sqrt{c^t(a|I_i)}})}{\sum_{a\in A(I_i)}(1+\frac{\beta}{\sqrt{c^t(a|I_i)}})}, \text{ (22)}$$

where $\sigma_i^{e,t}$ refers to the exploration policy in iteration t, β is a nonnegative real number. If $\beta = 0$, then $\sigma_i^{e,t}(a|I_i)$ is a uniform random exploration. If $\beta > 0$ and action a at I_i is sampled over and over again, $\sigma_i^{e,t}(a|I_i)$ tends to become small so that there is a potentially smaller probability to sample this action than the one without exploration. Exploration is empirically helpful in MCCFR. For example, if a cumulative regret of one action is negative, its current strategy is zero. In this situation, this action will not be sampled in the next iterations. However, this action could have a larger overall regret than other actions after long running iterations. Therefore, it will need a lot of iterations to make MCCFR change its negative regret to a positive value. When using exploration, MCCFR has a certain probability to sample this action and makes an exploration after some iterations.

Experiments have been carried out to evaluate the example techniques to accelerate the convergence of MCCFR on three different poker games: heads-up no-limit preflop hold'em poker (NLPH), heads-up no-limit flop hold'em poker (NLFH) and the river subgame of headsup no-limit Texas hold'em poker (HUNL-R). The techniques include the vector-form sampling policies, the variance reduction techniques with the counterfactual value baseline, the hybrid sampling policy with exploration, and hybrid MCCFR variants

[0123] A set of ablation studies are conducted related to different sampling policies, exploration techniques, and variance reduction techniques with a counterfactual baseline on NLPH. Then different MCCFR methods are compared on HUNL-R and extremely large NLFH.

[0124] All the experiments were evaluated by exploitability, which was used as a standard win rate measure. The method with a lower exploitability is better. Nash equilibrium has zero exploitability. The unit of exploitability in this specification is millibig blinds per game (mbb/g). It denotes how many thousandths of a big blind one player wins on average per hand of poker. For the abstracted large games, the exploitability is computed on the abstracted game. In the experiment, $\alpha = \frac{1}{\ln(t+10)}, \beta = \ln(t+10), \text{and } \gamma = 0.5$. Other values can be used. The experiments follow the typical procedure of MCCFR to traverse the public tree or game tree separately for each player. FIGS. 2-6 show examples of simulation results of multiple MCCFR variants in the experiments. The x-axis of each figure represents the number of iterations, and the y-axis of each figure represents the exploitability. Without loss of generality, robust sampling is used as an example sampling scheme for different MCCFR variants on NLPH poker. One effective version of robust sampling is the traverser samples 1 action according to the uniform random policy and the opponent samples 1 action according to her current strategy.

[0125] FIG. 2 is a log-log plot 200 illustrating convergence performances of several MCCFR variants applied to NLPH with different sampling policies in accordance with embodiments of this specification. MCCFR refers to the semi-vector-form MCCFR. MCCFR-RCS, MCCFR-MCS, MCCFR-WCS, and MCCFR-WAS refer to the vector-form MCCFR variants with different MCS, WCS, and WAS sampling policies, respectively. The results showed that MCCFR-RCS achieved similar convergence with semi-vector-form MCCFR because RCS randomly selected infoset $I_i$from $\vec{I}_i$and sampled one action according to σI^i. Such random selection does not consider the importance of different infosets. Except for MCCFR-RCS, other vector-form MCCFR variants achieve 2 or 3 orders of magnitude improvement against the semi-vector-form MCCFR. The WCS and WAS, which weighted each infoset by the range, have better long-term performance than MCS. Note that, typically semi-vector-form MCCFR converges faster than its value-form version so a convergence

[0129] FIG. 4B is a log-log plot 405 illustrating example computational efficiencies of several MCCFR variants applied to NLPH with and without different variance reduction techniques in accordance with embodiments of this specification. In the experiment, a semi-vector-form MCCFR (denoted as MCCFR) costs 5.9 seconds every 1000 iterations; the vector-form MCCFR-WCS (denoted as MCCFR-WCS) costs 6.2 seconds; the method with counterfactual baseline (e.g., either MCCFR-WCS(cfv b) and MCCFR-WCS(cfv b, boot)) costs 6.5 seconds and the method with expected utility value baseline (denoted as MCCFR-WCS(ev b)) costs 48.7 seconds.

[0130] Although the vector-form MCCFR samples more infosets than semi-vector-form MCCFR on each iteration, they cost similar computation time because of the benefit of the matrix manipulation. Empirically, the method with bootstrapping baseline (MCCFR-WCS(cfv b, boot)) converged slightly faster than the one using the CFV baseline described w.r.t. Eq. (19) (denoted as MCCFR-WCS(cfv b)). In the remaining experiment, the method with bootstrapping counterfactual baseline is selected as a default MCCFR variant.

[0131] FIGS. 5A-5C are log-log plots 500, 530, and 560 illustrating convergence performances of several MCCFR variants by external sampling on three different poker games: NLPH, HUNL-R and NLFH, in accordance with embodiments of this specification. FIGS. 5A-5C show that MCCFR with the described WCS sampling policy and bootstrapping baseline can significantly improve the convergence of MCCFR in many settings (including an extremely large game NLFH). The improved MCCFR could benefit many poker AIs and help them achieve better strategy in less running time.

FIG. 6A is log-log plot 600 illustrating convergence performances of several MCCFR variants with and without skipping on NLPH in accordance with embodiments of this specification. The experiments are performed by external sampling vector-form MCCFR. In CFR, the cumulative regret is initialized by zero and the current strategy starts from a uniform random strategy. In some embodiments, only the average strategy profile within all iterations is proved to converge to Nash equilibrium. In some embodiments, skipping previous iterations of CFR can obtain faster convergence of MCCFR. FIG. 6A shows that the MCCFR variants with different skipping iterations significantly improve the performance on NLPH. FIG. 6A shows the long-term performance of the MCCFR algorithm on NLPH over a long iteration horizon. The method with

skipping previous 10000 iterations (denoted as WCS(skip 10k)) converged to 0.94-Nash equilibrium. This exploitability was considered sufficiently converged in Texas hold'em.

FIG. 6B is log-log plot 650 illustrating convergence performances of MCCFR variants with and without skipping on NLPH in accordance with embodiments of this specification. The experiments are performed by external sampling vector-form MCCFR. As a discounting mechanism, a linear MCCFR weights the regrets and average strategies with a value dependent on the iteration t. In the experiment, this discounting mechanism is combined with the vector-form MCCFR with a specified weight of t^w. FIG. 6B shows that the linear MCCFR with a weight t^w, where w = 1 and w =2 (denoted as Linear WCS (w=1) and Linear WCS (w=1)), improves the convergence more than the vector-form MCCFR without discounting (denoted as WCS).

[0132] FIG. 7 is a flowchart of an example of a process 700 for performing Monte Carlo counterfactual regret minimization (MCCFR) for determining action selection policies for software applications in accordance with embodiments of this specification. The process 700 can be an example of the MCCFR algorithm with a sampling scheme described above.

[0133] The example process 700 shown in FIG. 7 can be modified or reconfigured to include additional, fewer, or different operations, which can be performed in the order shown or in a different order. In some instances, one or more of the operations can be repeated or iterated, for example, until a terminating condition is reached. In some implementations, one or more of the individual operations shown in FIG. 7 can be executed as multiple separate operations, or one or more subsets of the operations shown in FIG. 7 can be combined and executed as a single operation.

[0134] In some embodiments, the process 700 can be performed in an iterative manner, for example, by performing two or more iterations. In some embodiments, the process 700 can be used in automatic control, robotics, or any other applications that involve action selections. In some embodiments, the process 700 can be performed by an execution device for generating an action selection policy (e.g., a strategy) for completing a task (e.g., finding Nash equilibrium) in an environment that includes the execution device and one or more other devices. In some embodiments, generating the action selection policy can include some or all operations of the process 700, for example, by initiating an action selection policy at 702 and updating the action selection policy at 750 over iterations. The execution

[0170] In a current iteration (e.g., t-th iteration), at 810, a sampling policy in a state of the execution device is identified. The sampling policy specifies a respective sampling probability of sampling each of the plurality of possible actions in the state. In some embodiments, the sampling policy comprises a first probability distribution over the plurality of possible actions in the state. The sampling policy can be any one of the sample policies described with respect to FIG. 7. For example, the sampling policy can be one or more of a uniform sampling policy, a random sampling policy, a specified random policy, Random Current Strategy (RCS), Mean Current Strategy (MCS), Weighted Current Strategy (WCS), Weighted Average Strategy (WAS), or any other sampling policy. In some embodiments, the sampling policy in the state of the execution device is identified, for example, according to the example techniques described w.r.t. 730 of the process 700.

[0171] At 820, an exploration policy in the state of the execution device is identified. The exploration policy specifies a respective exploration probability corresponding to each of the plurality of possible actions in the state, wherein the exploration probability is negatively correlated with a number of times that the each of the plurality of possible actions in the state has been sampled. In some embodiments, the exploration policy comprises a second probability distribution over the plurality of possible actions in the state.

[0172] In some embodiments, the exploration policy in the state of the execution device is identified, for example, by computing the exploration probability of each of the plurality of possible actions according to Eq. (22), wherein i represents an identifier of the execution device (e.g., associated with player i); I_i represents an information set of the state; $A(I_i)$ represents the plurality of possible actions in the state; a represents one of the plurality of possible actions in the state; t represents a current iteration; $C^t(a|I_i)$ represents a number of times that the action a has been sampled in the state up to the iteration t; and $\sigma_i^{e,t}(a|I_i)$ represents an exploration policy of exploring the action a at the state in iteration t; and β is a nonnegative real number.

[0173] At 830, a hybrid sampling policy is computed based on the sampling policy and the exploration policy. In some embodiments, computing a hybrid sampling policy based on a sampling policy and an exploration policy comprises: computing a probability of each of the plurality of possible actions in the state based on a weight sum of the sampling probability of each of the plurality of possible actions in the state and the exploration probability of each of the

plurality of possible actions in the state. In some embodiments, computing a hybrid sampling policy based on a sampling policy and an exploration policy comprises computing a probability of each of the plurality of possible actions in the state according to Eq. (20), wherein I_i represents an information set of the state; a represents one of the plurality of possible actions in the state; $\sigma_i^{se}(a|I_i)$ represents the hybrid sampling policy of sampling the action a in the state; $\sigma_i^{s}(a|I_i)$ represents the sampling policy of sampling the action a in the state; $\sigma_i^{e}(a|I_i)$ represents the exploration policy of exploring the action a in the state; and $\alpha \in [0,1]$ represents a factor that controls a weight of exploration.

[0174] At 840, an action among the plurality of possible actions in the state is sampled according to a sampling probability of the action specified in the hybrid sampling policy. For example, for player 1 at the node 135a of the game tree 150 in FIG.1B, the hybrid sampling policy can include a hybrid sampling probability of sampling the action A_{1a} among the two possible actions A_{1a} and A_{1b} in the state of the node 135a (say a probability of 0.2), and a hybrid sampling probability of sampling the action A_{1b} among the two possible actions A_{1a} and A_{1b} in the state of the node 135a (say a probability of 0.8). The action A_{1b} can be sampled with a higher probability of 0.8 at the node 135a than the action A_{1a}. The sampled action A_{1b} can be used for updating an action selection policy for the next iteration.

[0175] At 842, in response to sampling the action out of the plurality of possible actions in the state according to the hybrid sampling policy, a number of times that the action has been sampled in the state is increased. In some embodiments, the number of times that the action has been sampled in the state comprises a number of times that the action has been sampled in the state up to the current iteration (e.g., $C^t(a|I_i)$).

[0176] At 844, the exploration probability corresponding to the action out of the plurality of possible actions in the state is decreased for computing the hybrid sampling policy in a next iteration (e.g., the *(t+1)-th* iteration) so that the action has a lower probability to be sampled in the next iteration. In some embodiments, the exploration policy in the state of the execution device is decreased w.r.t. the number of times that the action has been sampled in the state, for example, according to Eq. (22), or another function.

[0177] At 850, an action selection policy of the execution device in the state is updated based on the action (e.g., sampled action A_{1b} in the example described in 840) by performing Monte

Carlo counterfactual regret minimization (MCCFR) based on the action. The action selection policy specifies a respective probability of selecting an action among the plurality of possible actions in the state for completing the task in the environment. The action selection policy can be, for example, a current strategy of the execution device in the state. For example, in MCCFR, once an action is sampled according to the sampling policy, then the action selection policy can be updated based on one or more of a regret, CFV, and other values calculated based on the sampled action.

[0178] In some embodiments, updating the action selection policy of the execution device in the state based on the action comprises performing the MCCFR based on the action, for example, according to some or all of Eqs. (4)-(12). For example, performing Monte Carlo counterfactual regret minimization (MCCFR) based on the action comprises: calculating a probability of a sampled terminal sequence of actions (e.g. $q(z) = \sum_{j:z\in Q_j} q_{Q_j}$) based on a hybrid sampling probability of the action, the sampled terminal sequence of actions including the action and a terminal state for completing a task; calculating a sampled counterfactual value of the action based on the probability of the sampled terminal sequence of actions (e.g., according to Eq. (9)); calculating a regret value of the action based on the sampled counterfactual value of the action (e.g., according to some or all of Eqs. (10)-(12)); and updating the action selection policy of the execution device in the state based on the regret value of the action (e.g., according to regret matching based on Eq. (8) or regret matching+). In some embodiments, an *average strategy* $\bar{\sigma}_i^t$ after *the current* iteration can be computed, for example, according to Eq. (8).

[0179] After 850, the process 800 can go back to 804 to determine whether a convergence condition is met is determined. In some embodiments, in response to determining that the convergence condition is met, operations of the execution device are controlled based on the action selection policy. In some embodiments, in response to determining that the convergence condition is met, an average action selection policy across all iterations (e.g., from the first iteration to the current iteration) in each state can be computed. for example, according to Eq. (8). In some embodiments, the average action selection policy can serve as an output of the process 800, for example, as the computed Nash equilibrium.

[0180] In some embodiments, the action selection policy can serve as an output of the software-implemented application to automatically control the execution device's action at

[0230] In some embodiments, wherein computing a hybrid sampling policy based on a sampling policy and an exploration policy comprises: computing a probability of each of the plurality of possible actions in the state based on a weight sum of the sampling probability of each of the plurality of possible actions in the state and the exploration probability of each of the plurality of possible actions in the state.

[0231] In some embodiments, wherein computing a hybrid sampling policy based on a sampling policy and an exploration policy comprises: computing a probability of each of the plurality of possible actions in the state according to:

$$\sigma_i^{se}(a|I_i) = (1-\alpha) * \sigma_i^{s}(a|I_i) + \alpha * \sigma_i^{e}(a|I_i),$$

wherein: I_i represents an information set of the state; a represents one of the plurality of possible actions; $\sigma_i^{se}(a|I_i)$ represents a hybrid sampling policy of sampling the action a in the state; $\sigma_i^{s}(a|I_i)$ represents a sampling policy of sampling the action a in the state; $\sigma_i^{e}(a|I_i)$ represents an exploration policy of exploring the action a in the state; and $\alpha \in [0,1]$ represents a factor that controls a weight of exploration.

[0232] In some embodiments, wherein the exploration probability of each of the plurality of possible actions in the state is computed according to:

$$\sigma_i^{e,t}(a|I_i) = \frac{(1+\frac{\beta}{\sqrt{c^t(a|I_i)}})}{\sum_{a\in A(I_i)}(1+\frac{\beta}{\sqrt{c^t(a|I_i)}})},$$

wherein: i represents an identifier of the execution device; I_i represents an information set of the state; $A(I_i)$ represents the plurality of possible actions in the state; a represents one of the plurality of possible actions in the state; t represents a current iteration; $C^t(a|I_i)$ represents a number of times that the action a has been sampled in the state up to the current iteration t; $\sigma_i^{e,t}(a|I_i)$ represents an exploration policy of exploring the action a at the state in the current iteration t; and β is a nonnegative real number.

[0233] FIG. 13 is a diagram of an example of modules of an apparatus 1300 in accordance with embodiments of this specification. The apparatus 1300 can be an example embodiment of a data processing apparatus or an execution device for generating an action selection policy for completing a task in an environment that includes the execution device and one or more other devices. The apparatus 1300 can correspond to the embodiments described above, and the

apparatus 1300 includes the following: for in a current iteration of a plurality of iterations, a first computing module 1301 for computing a counterfactual value (CFV) of the execution device in a terminal state of completing a task based on a payoff of the execution device at the terminal state and a reach probability of the one or more other devices reaching the terminal state, wherein the terminal state results from a sequence of actions taken at a plurality of non-terminal states by the execution device and by the one or more other devices, wherein each of the plurality of non-terminal states has one or more child states; a second computing module 1302 for computing a baseline-corrected CFV of the execution device in the terminal state based on the CFV of the execution device in the terminal state, a CFV baseline of the execution device in the terminal state of a previous iteration, or both; for each of the non-terminal states and starting from a non-terminal state that has the terminal state and one or more other terminal states as child states: a third computing module 1303 for computing a CFV of the execution device in the non-terminal state based on a weighted sum of the baseline-corrected CFVs of the execution device in the child states of the non-terminal state; a fourth computing module 1304 for computing a baseline-corrected CFV of the execution device in the non-terminal state based on the CFV of the execution device in the non-terminal state, a CFV baseline of the execution device in the non-terminal state of a previous iteration, or both; a fifth computing module 1305 for computing a CFV baseline of the execution device in the non-terminal state of the current iteration based on a weighted sum of the CFV baseline of the execution device in the non-terminal state of the previous iteration and the CFV or the baseline-corrected CFV of the execution device in the non-terminal state; and a determining module 1306 for determining an action selection policy in the non-terminal state for the next iteration based on the baseline-corrected CFV of the execution device in the non-terminal state of the current iteration.

[0234] In some embodiments, the apparatus 1300 further includes the following: a controlling identifying module 1307 for controlling operations of the execution device in the non-terminal state based on the action selection policy in the non-terminal state for the next iteration in response to determining that a convergence condition is met.

[0235] In some embodiments, wherein determining an action selection policy in the non-terminal state for the next iteration based on the baseline-corrected CFV of the execution device in the non-terminal state of the current iteration comprises: calculating a regret value