# State Evaluation Strategy for Exemplar-Based Policy Optimization of Dynamic Decision Problems

Kokolo IKEDA and Hajime KITA

***Abstract*— Direct policy search (DPS) that optimizes the parameters of a decision making model, combined with evolutionary algorithms which enable robust optimization, is a promising approach to dynamic decision problems. Exemplar-based policy (EBP) optimization is a novel framework for DPS in which the policy is composed of a set of exemplars and a case-based action selector, with the set of exemplars being refined and evolved using a GA. In this paper, state evaluation type EBP representations are proposed for the problem class whose state transition can be predicted. For example, the vector-real representation defines pairs of feature vector and its desirability as exemplars, and evaluate the predicted next states using the exemplars. The state evaluation type EBP-based optimization procedures are shown to be superior to conventional state-action type EBP optimization through application to the Tetris game.**

## I. Introduction

Markov Decision Processes (MDPs) are a typical class of formulation for dynamic decision-making problems such as control, economy, or game problems. In such problems, the learning agent develops a policy, representing a mapping from a set of states to a set of actions, through trial-and-errors and evaluations.

A common approach that has been studied extensively for the training of MDPs is reinforcement learning, such as the temporal difference (TD) method [12]. Another common approach is direct policy search (DPS), in which a policy is represented by a model with parameters, such as an artificial neural network [13], and the parameters are optimized so as to maximize the evaluation function by applying the parameterized policy to the problem. DPS approach is expected as a more robust method applicable to broad problem classes, compared with the TD method, for its flexibility of the model selection and its power of direct search method such as genetic algorithms (GAs). To tackle more difficult problems by the DPS approach, several properties are considered to be important; generalization and localization ability, flexibility, and the possibility of introducing prior knowledge.

Exemplar-based policy (EBP) optimization is a novel framework for DPS [1][2]. In EBP, the policy is composed of a set of exemplars and a case-based action selector. In the present context, an exemplar represents specific information, such as the pair $(s_j, a_j)$ meaning "the best action on a state $s_j$ is $a_j$". In this example, when the current state $s$ is given, the nearest $s_{j*}$ is selected and the associated action $a_{j*}$ is taken. An implementation using this style of EBP and a GA has been shown to provide favorable search performance for two higher-dimensional problems, and the generalization and localization ability of the derivation process has been shown [1]. Further, it has been shown that the combination of EBP and GAs for selecting sets of exemplars significantly improves performance.

The authors are with the Academic Center for Computing and Media Studies, Kyoto University, Yoshida-Nihonmatsu, Sakyo, Kyoto, Japan (email: kokolo@media.kyoto-u.ac.jp, kita@media.kyoto-u.ac.jp).

Considering more practical problems, the present paper discusses the two extensions for EBP-based optimization. One extension is the new style of EBP for evaluating states of MDP. When the transition of an MDP is predictable, it is standard and often effective approach to predict the all subsequent states corresponding to the possible actions and evaluate them, to select the best action [6]. For this approach, a vector-vector style EBP and a vector-real style EBP are presented. For example, in the vector-real EBP representation, an exemplar is given by the pair $(v_j, r_j)$ meaning "the desirability of $v_j$ is $r_j$", and the evaluation of states is done using the set of exemplars.

The other extension is the way for the introduction of prior knowledge. In many problems, the expert's decision records are available as the prior knowledge. For accelerating the optimization process, the way to introduce such records as the initial sets of GA solutions is presented. The effectiveness of the proposed methods is demonstrated through a search for automatic players of the Tetris game.

## II. Transition-Predictable MDP

In this paper, the learning policy is discussed with respect to a subclass of MDP in which the state transition is predictable before an action is taken, and the trial is episodic (*i.e.*, there is a beginning and an end of the trial). In other words, the transition function is given as part of the problem statement. This subclass is referred to as a transition-predictable MDP. Solitaire games are a typical example of this subclass.

Theoretically the optimal policy for a transition-predictable MDP can be obtained by search with the dynamic programming (DP) technique. However, practical problems require search with policy having generalization (though DP methods also can generalize), due to the cost of such enumerative search in large state spaces.

A transition-predictable MDP is composed of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a public transition function $\mathcal{T}(s,a) : \mathcal{S}\times\mathcal{A}\rightarrow\mathcal{S}$ and a reward function $\mathcal{R}(s,a) : \mathcal{S}\times\mathcal{A}\rightarrow\mathbb{R}$. When the state space is too large to deal with effectively, the feature values of the space are often used instead, such as the $n$-dimensional feature function $\mathcal{F}(s) : S \rightarrow \mathcal{V} = \mathbb{R}^n$. The performance of a trial given a prescribed policy can be

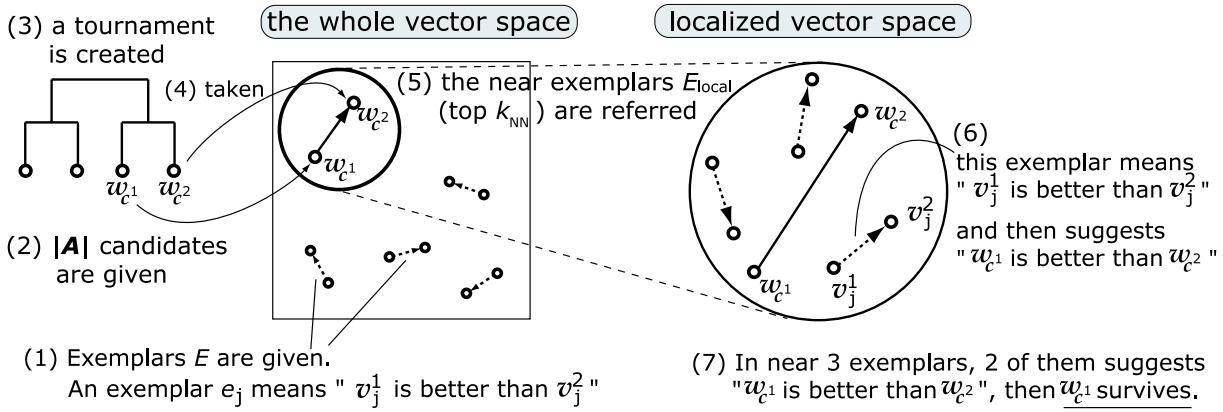


Fig. 1. Selection of most preferable vector from candidates. When the set of exemplars and candidate vectors are given, a binary tournament is created. For judging which vector is better, the localized (near to focused vectors) set of exemplars are used.

evaluated by the total reward, *i.e.* the summation of rewards from the beginning to the end.

## III. Exemplar-based Policy Optimization

The combination of GA and case-based methods is one of the most promising approaches for difficult problems, various methods have been studied [9]. Exemplar-based policy optimization using a GA (EBP-GA) is a novel optimization framework in which the policy is composed of a set of exemplars and a case-based action selector, with the set of exemplars being refined and evolved using a GA. This approach has been shown to provide favorable performance by balancing generalization ability with localization ability [1][2].

In a state-action style EBP (SAP), as used in [1], an exemplar is defined as the pair $(s_j, a_j) \in \mathcal{S} \times \mathcal{A}$ meaning "the best action on a state $s_j$ is $a_j$". The nearest-neighbor classification is used as the action selector [8], that is, when the current state is $s$, the nearest state $s_{j*}$ is selected from the set of exemplars, and the associated action $a_{j*}$ is taken (in fact, feature vectors $\mathcal{F}$ are used instead of states in this paper). The state following the action is not considered in this approach.

### A. State Evaluation Strategy for EBP

In a transition-predictable MDP, the state transition function $\mathcal{T}$ is public. Therefore, when there are $|\mathcal{A}|$ actions $\{a^j\}$ available on a state $s \in \mathcal{S}$, the next states $\{\mathcal{T}(s, a^j)\}$ or the corresponding feature vectors $\{\mathcal{F}(\mathcal{T}(s, a^j))\}$ are predictable. If a state evaluation function is given, the best next state and the corresponding action can be selected. Such a scheme is termed a "state evaluation strategy", and is often used in artificial intelligence algorithms for game playing [4] [6].

A vector-real style EBP (VRP) can be introduced to realize a state evaluation strategy. In VRP, an exemplar is represented as the pair of a feature vector $v_j \in \mathcal{V} = \mathbb{R}^n$ and the corresponding evaluation value $r \in \mathbb{R}$. When a next state (or feature vector) is given, it is evaluated from the set of exemplars and a function generalization method. In this paper, the weighted averaging is employed as the function generalization, and the procedure for decision making is as follows.

1) Exemplars $E = \{e_j\}_j = \{(v_j, r_j)\}_j$ are given, where $v_j \in \mathcal{V} = \mathbb{R}^n$ and $r_j \in \mathbb{R}$.
2) The next states corresponding to possible actions are predicted, and their feature vectors $\{w_{c^a}\} = \{\mathcal{F}(\mathcal{T}(s, a)) \mid a \in \mathcal{A}\}$, $\mid \mathcal{A} \mid$ candidates, are given to be evaluated.
3) For each feature vector $w_{c^a}$, the weighted average value $g(w_{c^a})$ is calculated as the following equations.

$$g(w_{c^a}) = \frac{\sum_{e_j \in E} r_j d(v_j, w_{c^a})}{\sum_{e_j \in E} d(v_j, w_{c^a})},\ d(v_j, w_{c^a}) = \frac{1}{1 + ||v_j - w_{c^a}||}$$

4) The action $a^* = \mathrm{argmax}_{a \in \mathcal{A}} g(w_{c^a})$ is taken.

A vector-vector style EBP (VVP) can also be derived from the state evaluation strategy. In VVP, an exemplar is defined as the pair $(v_j^1, v_j^2) \in \mathcal{V} \times \mathcal{V}$ meaning "$v_j^1$ is better than $v_j^2$". When the feature vectors $\{w_{c^a}\} = \{\mathcal{F}(\mathcal{T}(s, a)) \mid a \in \mathcal{A}\}$ are given, a tournament is created and $\mid \mathcal{A} \mid -1$ competitions based on the set of exemplars are conducted, yielding a winning vector $w_{c^{a^*}}$ with a corresponding action $a^*$. One implementation for conducting such competitions has been proposed in [2] as follows.

1) Exemplars $E = \{(v_j^1, v_j^2)\}_j$ are given, where $v_j^1, v_j^2 \in \mathcal{V} = \mathbb{R}^n$.
2) The next states corresponding to possible actions are predicted, and their feature vectors $\{w_{c^a}\} = \{\mathcal{F}(\mathcal{T}(s, a)) \mid a \in \mathcal{A}\}$, $\mid \mathcal{A} \mid$ candidates, are given to be evaluated.
3) An unbiased tournament for candidates is randomly created (the transitive law may not necessarily hold in this competition procedure).
4) A pair of competitors $w_{c^1} \in \mathcal{V}$ and $w_{c^2} \in \mathcal{V}$ are taken by following the tournament.
5) For each exemplar $(v_j^1, v_j^2) \in E$, the distance to the competitors $dist_j = |\frac{w_{c^1} + w_{c^2}}{2} - \frac{v_j^1 + v_j^2}{2}|$ is calculated,

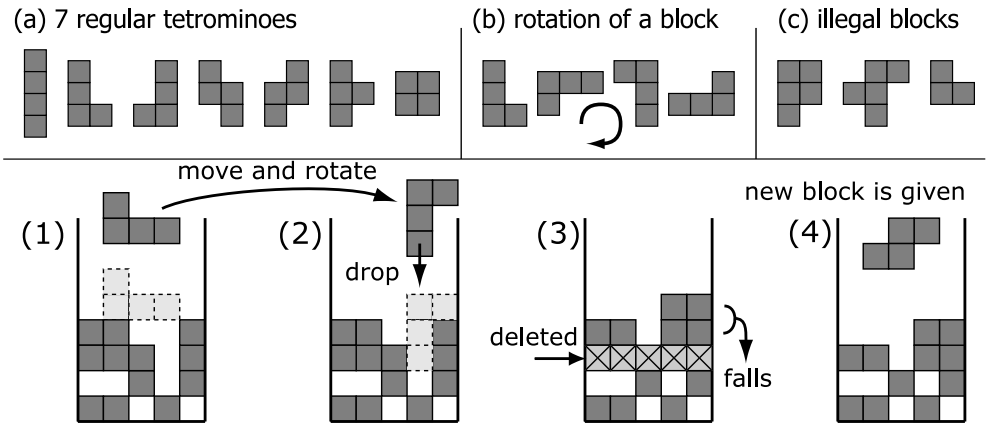


Fig. 2. The tetrominoes (upper) and the state transition (lower) of Tetris: (1) the field and tetromino, (2) move, rotate and drop, (3) a line is filled then deleted, (4) the blocks over the deleted line fall, and the next tetromino is given.

and $E_{local} \in E$, the top $k_{\mathrm{NN}}$ exemplars nearest within $dist_j$, are selected ($k_{\mathrm{NN}}$ is the localization parameter).

6) For each exemplar $(v_j^1, v_j^2) \in E_{local}$, the direction $\overrightarrow{v_j^2 - v_j^1}$ and the inner product $IP_j = \overrightarrow{v_j^2 - v_j^1} \cdot \overrightarrow{w_{c^2} - w_{c^1}}$ are calculated. When $IP_j > 0$, the exemplar suggests that "$w_{c^1}$ is better than $w_{c^2}$".
7) The number of exemplars in $E_{local}$ for which $IP_j > 0$, *i.e.* $|\ \{(v_j^1, v_j^2) \in E_{\mathrm{local}}, IP_j > 0\}\ |$ is counted. If the number is larger than $|E_{local}|/2$, $w_{c^1}$ survives the competition (otherwise the opposite judgment is obtained).
8) After $|\mathcal{A}|-1$ competitions have been completed, the winner is obtained.

### B. Knowledge introduction using expert records

There are many approaches for introducing prior knowledge into optimization methods, including definition of feature values for a state, fixation of a policy model, and adjustment of parameters. In this paper, the records made by experts are introduced as the initial set of exemplars. In actual control problems or decision-making problems, it is likely to be easier for an expert to demonstrate actions rather than extract appropriate rules under the actions. Especially in many games, such as chess or go, many expert records are publicly available. Such records are usually given in the form of sequences of "the action $a_j$ was selected at the state $s_j$". As such records are state-action exemplars, they are directly applicable to the GA for SAP optimization (SAP-GA) procedure as initial exemplars.

In this paper, the way to introduce such records for VVP is presented. A record "the action $a^{i*} \in \mathcal{A}$ was selected by the expert at the state $s$" means that "other actions $a^i \in \mathcal{A} \setminus \{a^{i*}\}$ are judged to be inferior (or even) to $a^{i*}$". Let $w^i = \mathcal{F}(\mathcal{T}(s, a^i)) \in \mathcal{V}$ represent the predicted feature vector of the next state after action $a^i$. This record also suggests "vectors $w^i (a^i \in \mathcal{A} \setminus \{a^{i*}\})$ are inferior (or even) to $w^{i*}$". Then these $|\mathcal{A}|-1$ suggestions "$w^{i*}$ is better than $w^i$" can be introduced into the GA for VVP optimization (VVP-GA) as initial exemplars. On the other hand, introduction of records into vector-real EBP is not straightforward, requiring extra learning, since no specific value of a feature vector is written in the records.

## IV. Tetris Game

Tetris is a popular puzzle game developed by Aleksey Pazhitnov as a real-time game. Substantial research , especially by the reinforcement learning community, has been conducted on the generation of automatic Tetris players [10]. For example, in [3], a real robot was developed to play the game using a camera to recognize the display and fingers to push the keyboard. A MDP version of Tetris has also been considered. In [11], an elaborate learner for Tetris was developed, in which state evaluation function adaptation using many heuristic features and neural networks was adopted.

The main components of Tetris are the tetrominoes (see Fig. 2(a)) and the field, which consists of $C_{\mathrm{width}} \times C_{\mathrm{height}}$ squares. The field is normally initialized such that all squares are empty. In each turn, a random tetromino is given and the player decides where to drop it, and whether to rotate it or not. When the tetromino is dropped, if there exist horizontal lines fully filled with blocks, the lines are deleted and blocks above the lines fall by corresponding squares. The purpose of Tetris is to delete $C_{\mathrm{norm}}$ lines, and if the player has no space to drop the given tetromino, the game is over. Figure 2 shows a brief sample of the game. In this paper, the parameters are set at $(C_{\mathrm{width}}, C_{\mathrm{height}}, C_{\mathrm{norm}}) = (6, 12, 10)$.

### A. Tetris as a transition-predictable MDP

In this paper, very simplified version of Tetris is used as a MDP problem. The state space $\mathcal{S}$ for Tetris is defined by the kind of tetromino and the field matrix $M_{xy}$, where $M_{xy} = 1$ when a square of position $(x, y)$ is filled, and $M_{xy} = 0$ when empty. As the full set of information for a state $s \in \mathcal{S}$ is too large (*ca.* $2^{6\times12}$) to deal with effectively, the feature vector $\mathcal{F}(s)$ is employed for artificial players. The feature values are calculated as follows. Such values are comparatively primitive.

1) The first feature $f_0$ is the ID of the current tetromino. This feature is ignored in VRP/VVP because the tetromino of the next step is not predictable.
2) The latter feature $f_x$ is the height of vertical lines, *i.e.*, $f_x = \max(y \mid M_{xy} = 1)$. When there are no blocks in the vertical line, $f_x = 0$.
3) Totally $C_{\text{width}}+1$ features are given. In the optimization phase, such feature values are independently normalized to $[0, 1]$ and Euclidean distance is used as a metric.

The action set $\mathcal{A}$ is defined by the left/right movements and rotations. There are $C_{\text{width}}$ movements and $4$ rotations at max, so we consider there are $4C_{\text{width}}$ actions in this MDP though the definition is a little redundant. The state transition $\mathcal{T}$ is defined by the rule, and the reward $\mathcal{R}$ is defined as follows: If lines are deleted, a reward of $1.0$ is given for each line deleted, otherwise $0.0$. When the game ends successfully, $10$ lines are deleted, totally reward $10.0$ is given.

### B. Heuristic automatic player for Tetris

In this section, a well-considered heuristic Tetris player is introduced, to define the standards for evaluating the optimization methods, to show the sensitivity of this game, and to provide game records to be utilized in EBP-GA optimization.

In Tetris, the next state after an action is predictable for players, except the kind of next tetromino given randomly. The heuristic player is therefore based on a state evaluation strategy. When a feature vector $(f_0, f_1, ..., f_{C_{\text{width}}})$ is given, the squared height $g_h = \sum_{x=1}^{C_{\text{width}}} f_x^2$ and the roughness $g_r = \sum_{x=2}^{C_{\text{width}}} (f_x + f_{x-1}) \, |f_x - f_{x-1}|$ are regarded. Each next state is evaluated by $g = g_h + g_r$, and the state with the lowest score is selected. This heuristic player tends to minimize the height and the roughness of the surface, because roughness often results in harmful empty spaces. The performance of the player is evaluated for comparison, the average reward is $9.80$ and the probability of a successful game end is $93.5\%$.

### C. Sensitivity of Tetris

The sensitivity of the playing policy in this game can be demonstrated by intermixing random actions with the heuristic player at a probability of $\varepsilon$. Figure 3 shows the number of successful games in 1000 trials for various values of $\varepsilon$. By intermixing random actions only $1/10$ times, even the heuristic player wins only 2/3 of games. The result suggests that Tetris is a sensitive game in which even a few mistakes can have a substantial detrimental effect.

## V. Experiments

In this section, positive game records are prepared, the particular implementations of EBP-GA are introduced, and the performance of them with and without initial positive records are determined through numerical simulation. The aims of the experiments are to show the effectiveness of introducing the initial positive records, and to compare the performance of conventional EBP (SAP) and the proposed methods with the state evaluation strategy (VVP, VRP).

Fig. 3. Performance variation for the heuristic player by mixing random actions at probability $\varepsilon$

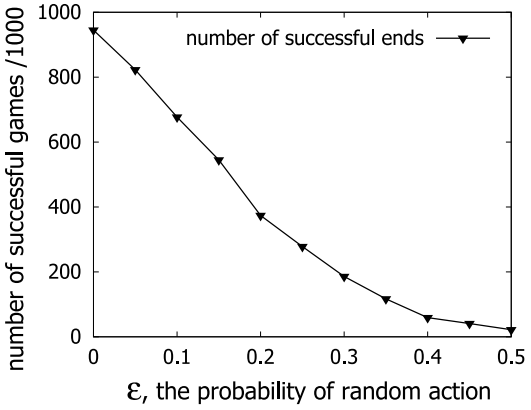


TABLE II
Performance of SAP and VVP with respect to the number of exemplars

| **SAP** | | All policies | | |
|---|---|---|---|---|
| $N_{\text{exemplars}}$ | $N_{\text{policy}}$ | Av. rwd. | Std. dev. | Successful |
| 300 | 3000 | 1.890 | 0.517 | 0.13% |
| 1000 | 900 | 3.060 | 0.708 | 1.50% |
| 3000 | 300 | 4.538 | 0.874 | 6.30% |

| **VVP** | | | All policies | | |
|---|---|---|---|---|---|
| $N_{\text{exemplars}}$ | $k_{\text{NN}}$ | $N_{\text{policy}}$ | Av. rwd. | Std. dev. | Successful |
| 300 | 75 | 3000 | 6.926 | 1.153 | 39.82% |
| 1000 | 251 | 900 | 7.626 | 0.981 | 49.70% |
| 3000 | 751 | 300 | 7.973 | 0.959 | 53.80% |

### A. Artificial game records for Tetris

When a set of positive records of games is given, it can be introduced as the set of exemplars of SAP and VVP. In this section, we prepare many positive exemplars and evaluate them.

Here, records produced by the heuristic player are utilized. Over 24,000 games were carried out by the player, and 900,000 state-action exemplars were prepared. Note that only the first 1,100 games were required for preparation of the same number of vector-vector exemplars.

At first the initial set of exemplars were evaluated without evolution. The 900,000 exemplars were divided into $N_{\text{policy}}$ policies such that each policy consists of $N_{\text{exemplars}}$ exemplars. Table II shows the average rewards and the average probabilities of successful game end for the policies. The results reveal that the performance of a policy increases with the number of exemplars employed, and that the performance of VVP is superior to that of SAP.

In the following experiments, $N_{\text{exemplars}}$ is limited to $300$, and the collection of useful exemplars by the GA is confirmed.

### B. GA for EBP optimization

The procedure for the evolution of the set of exemplars using a GA is common to SAP-GA, VRP-GA, and VVP-GA. The procedure is as follows (see Fig. 4).

TABLE I

NOTATION AND VALUES OF THE PARAMETERS OF EBP-GA

| Symbol | Explanation | Value |
|---|---|---|
| $N_{\text{exemplars}}$ | Number of exemplars in a policy | 100 |
| $k_{\text{NN}}$ | Localization parameter (the smaller, the more localized) | 25 |
| $N_{\text{pop}}$ | Number of solutions (policies) in a population | 100 |
| $N_{\text{children}}$ | Number of children produced in a reproduction | 10 |
| $N_{\text{samples}}$ | Number of games for one evaluation | 10 |
| $N_{\text{generations}}$ | Number of generations for a GA | 100 |
| $E_i$ | The $i$th solution (policy), the set of exemplars of the $i$th policy | - |
| $e_{i,j}$ | The $j$th exemplar of $E_i$ | - |

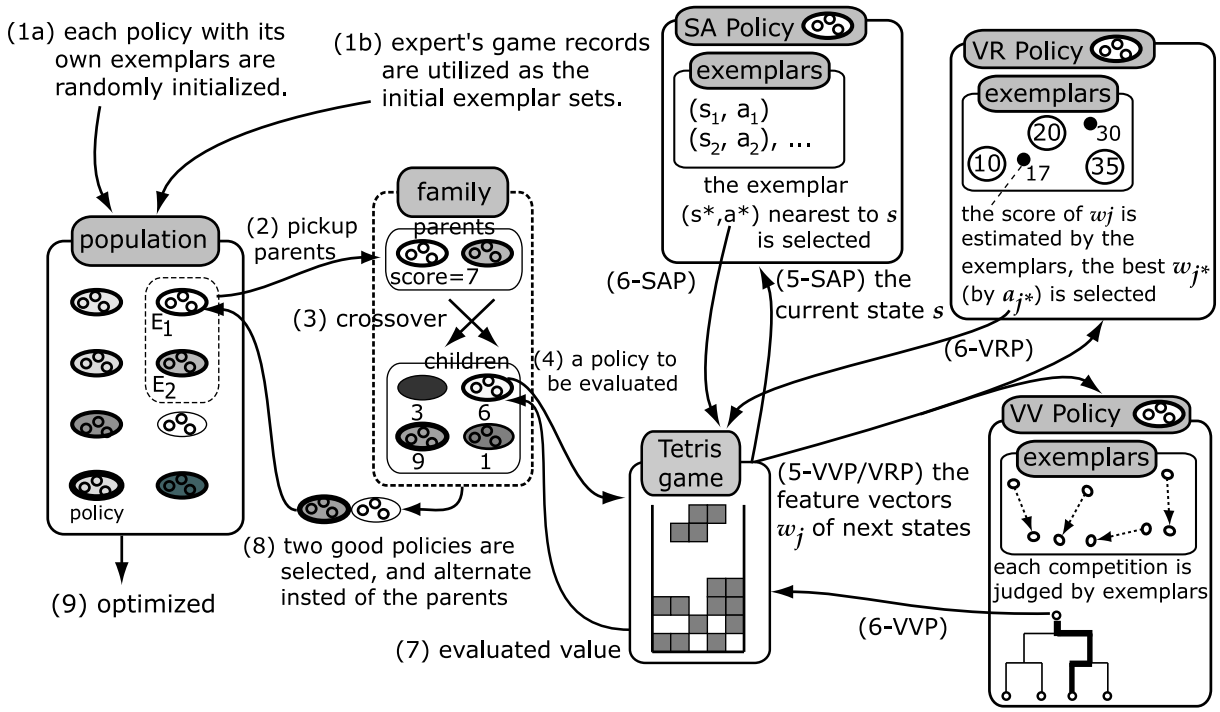


Fig. 4. EBP-GA process. (1) Population initialization using (1a) random exemplars or (1b) heuristic game records. (2) Parents selection. (3) Children production. (4) Policy evaluation. (5-SAP) The current state is given, (6-SAP) the exemplar nearest to it is selected, and the associated action is taken. Or, (5-VVP/VRP) feature vectors for all actions are given, (6-VVP/VRP) the best feature vector is selected by the exemplars, and the corresponding action is taken. (7) Total rewards is obtained. (8) Two good policies are sent back to the population instead of the parents.

1) The parameters such as $N_{\text{pop}}$ are fixed (see Table I).
2) As the population, $N_{\text{pop}}$ sets of exemplars $\{E_i\}_i$ are initialized. Each solution has $N_{\text{exemplars}}$ exemplars.
   - If positive game records are available, the records are divided into initial solutions. (30 trials $\times$ 100 solutions $\times$ 300 exemplars per a solution $=$ 900,000 exemplars are used)
   - If no prior knowledge is given:
     - A state-action exemplar $e_{i,j} = (s_{i,j}, a_{i,j}) \in E_i$ is generated such that $s_{i,j} \in [0,1]^n$ and $a_{i,j} \in \mathcal{A}$. ($n$ is the number of features, in this case 7.)
     - A vector-real exemplar $e_{i,j} = (v_{i,j}, r_{i,j}) \in E_i$ is generated such that $v_{i,j} \in [0,1]^n$ and $r_{i,j} \in [0,1]$.
     - A vector-vector exemplar $e_{i,j} = (v^1_{i,j}, v^2_{i,j}) \in E_i$ is generated such that $v^1_{i,j} + v^2_{i,j} \in [0,2]^n$ and $v^2_{i,j} - v^1_{i,j} \in [-1,1]^n$.
3) The parent solutions $E_{p1}$ and $E_{p2}$ are randomly selected from the population, $N_{\text{pop}}$ solutions.
4) Children are reproduced by applying the crossover operator $N_{\text{children}}$ times. The crossover we employ is performed by mixing the exemplars of the parents.
   a) The set of exemplars of a child $E_c$ is initialized as an empty set.
   b) An exemplar $e \in E_{p1} \cup E_{p2}$ is randomly selected. If $e \notin E_c$, $e$ is added to $E_c$.
   c) Step (b) is repeated until $|E_c| = N_{\text{exemplars}}$.
5) The evaluation values of them are calculated by applying a policy to the Tetris game. To reduce random

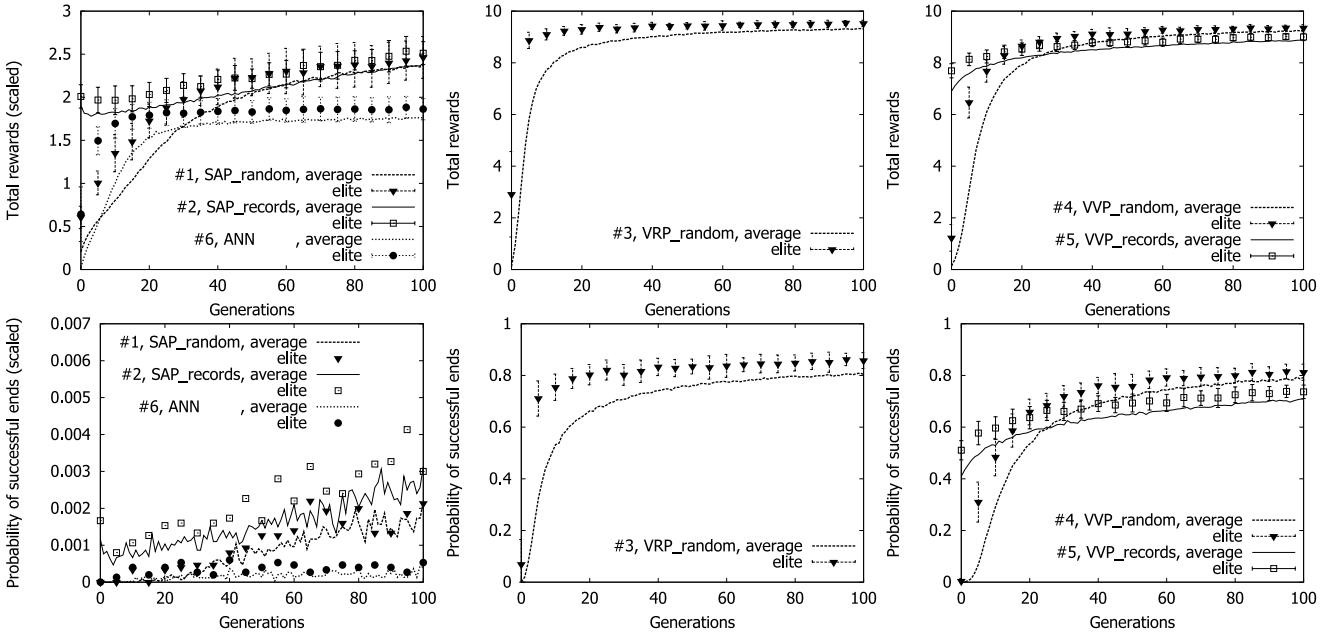


Fig. 5. EBP-GA results, showing the rewards (upper) and the probabilities of success (lower). VRP-GA and VVP-GA perform better than state-action methods (ANN, SAP). Further, introducing positive records to the initial set of exemplars improves the performance in early stage of GA.

fluctuation in the evaluation value, $N_{\text{samples}}$ games are performed independently, and the average of rewards is used.

6) The best evaluated policy $E_*$ in the family is selected, and one more policy $E_{**}$ is selected based on a ranking selection [7].
7) The parents in the population are replaced by $E_*$ and $E_{**}$.
8) Step 3 is repeated $N_{\text{generations}} \times N_{\text{pop}}$ times.

### C. GA for Artificial Neural Network

As one of other DPSs, we employ the combination of an artificial neural network (ANN) and a GA. In this approach, three-layered ANN is used as the decision making mapping from the observed state to an action (see Figure 6), and its connection weights $w$ are optimized by a real-coded GA using MGG alternation model [7] and UNDX-m crossover [5]. Parameters such as $N_{\text{pop}}$ and $N_{\text{children}}$ are same to EBP's, and the number of hidden-layer units is fixed to 20.

### D. Optimization Results

For comparison, 6 series of GAs were performed. Table III shows the summary of the GAs.

As a performance measure, the reward and the probability of successful game end were used. All series were repeated 30 times. Figure 5 shows the average performance of the population, and the re-evaluated performance of the elite policy.

The results demonstrate that the VRP-GA and the VVP-GA performed well, with the final elite policy achieving over 9.0 rewards (#3, #4, #5). The well-trained policy from the raw heuristic records (#5) is superior to the policy consisting of 3000 raw records reported in section V-A, and the progress from the initial populations (rewards 6.926) is clear. Therefore, it is suggested that useful exemplars were correctly collected by the GA.

The performances of the SAP-GA and the ANN (rewards about $1.5 - 3.0$) were worse, despite exhibiting some evolution in GA processing. The reason why they were worse than the VRP-GA and the VVP-GA is considered in the next

Fig. 6. Three-layered perceptron as a decision maker

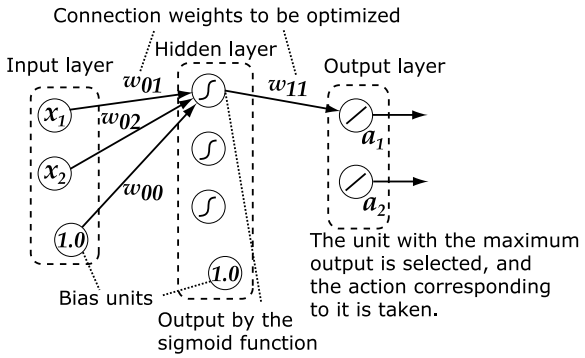


TABLE III
THE LIST OF GAS COMPARED

| No. | Name | Decision making | Using records |
|---|---|---|---|
| #1 | $\text{SAP}_{\text{random}}$ | State-Action EBP | no |
| #2 | $\text{SAP}_{\text{records}}$ | State-Action EBP | yes |
| #3 | $\text{VRP}_{\text{random}}$ | Vector-Real EBP | no |
| #4 | $\text{VVP}_{\text{random}}$ | Vector-Vector EBP | no |
| #5 | $\text{VVP}_{\text{records}}$ | Vector-Vector EBP | yes |
| #6 | ANN | State-Action ANN | no |

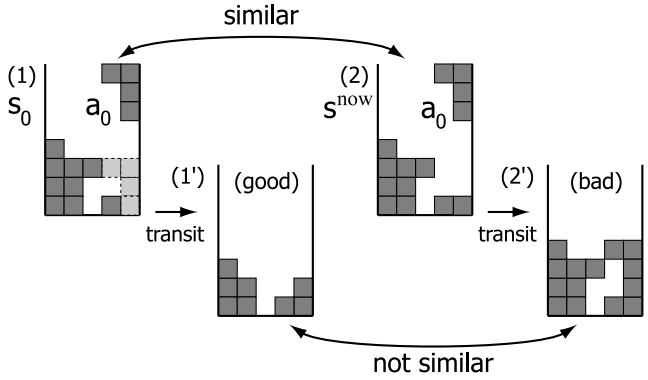


Fig. 7. Typical failure of state-action EBP

section.

Finally, the introduction of records to the initial population can be seen to assist in the early stage of GA evolution, and is considered to be useful especially in the problems whose evaluation costs are very expensive.

### *E. Discussion*

To examine the poor performance of the SAP-GA, the typical example shown in Figure 7 can be considered. Given the exemplar $(s_0, a_0)$ (1), the next state following the action is good (1'). However, if another state $s^{now}$ is given to an SAP player and $s^{now}$ is similar to $s_0$, the associated action $a_0$ is taken (2). Unfortunately, the slight difference between the two states causes a great difference between the next states (2'). Also, the reason of the poor performance of ANN is the difficulty of dividing the state space enough finely to distinguish the slight differences. Such transition structure of MDPs and such failure of the state-action approaches are expected to be observed not only in Tetris game. VRP and VVP approaches in which the decision is made after the next states are predicted, are promising in such cases.

At last an interesting ability of VRP and VVP is considered: VRP/VVP player can respond to the case of illegal blocks (see Fig. 2(c)) inserted among the regular tetrominoes. This is attributable to their focus not on the current state and actions but on the next states. In contrast, the state-action approaches give up in this situation.

## VI. Conclusion

Exemplar-based policy optimization is a novel approach for learning problems, which uses evolutionary optimization for selecting sets of exemplars. In this paper, two advanced usages of exemplar-based policy (EBP) optimization were proposed. A vector-vector style EBP and a vector-real style EBP were presented as novel representations of a policy, which ensures that if the transition of a MDP is predictable, all the subsequent states are predictable and can be evaluated using the set of exemplars, allowing the best next state and corresponding action to be taken. The introduction of expert records into EBP optimization was also shown to accelerate the early stage of optimization. The effectiveness of these methods was demonstrated through search of automatic players for the Tetris game.

## References

[1] K. Ikeda. Exemplar-based direct policy search with evolutionary optimization. *Congress on Evolutionary Computation*, pages 2357–2364, 2005.

[2] K. Ikeda, H. Suzuki, S. Markon, and H. Kita. Evolutionary optimization of a controller for multi-car elevators. *IEEE International Conference on Industrial Technology*, 2006.

[3] M. Jun. Game playing intelligent robot with logical and intuitive information processing architecture. *13th SICE symposium on decentralized Autonomous system*, pages 363–366, 2001.

[4] G. Kendall and G. Whitwell. An evolutionary approach for the tuning of a chess evaluation function using population dynamics. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 995–1002, 2001.

[5] H. Kita, I. Ono, and S. Kobayashi. Multi-parental extension of the unimodal normal distribution crossover for real-coded genetic algorithms. *Congress on Evolutionary Computation*, pages 1581–1587, 1999.

[6] A. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal*, 3(3).

[7] H. Satoh, M. Yamamura, and S. Kobayashi. Minimal generation gap model for gas considering both exploration and exploitation. *Proc. of IIZUKA*.

[8] J. Sheppard and S. Salzberg. A teaching strategy for memory-based control. *Artificial Intelligence Review 11*, pages 343–370, 1997.

[9] Sushil J. Louis and John McDonnell. Learning with case injected genetic algorithms. *IEEE Transactions on Evolutionary Computations, 8(4)*, pages 316–328, 2004.

[10] Szita, I. and Lorincz, A. Learning Tetris Using the Noisy Cross-Entropy Method. *Neural Computation 18*, pages 2936–2941, 2006.

[11] N. Sotaro and Y. Kazuo. Application of evolutionary reinforcement to learning of tetris game. *13th SICE symposium on decentralized Autonomous system*, pages 265–270, 2001.

[12] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning 3*, pages 9–44, 1988.

[13] D. Whitley and J. Kauth. Genitor: A different genetic algorithm. *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, pages 116–121, 1988.