2009 Ninth International Conference on Intelligent Systems Design and Applications

Solving Multi-Objective Reinforcement Learning Problems by EDA-RL — Acquisition of Various Strategies —

Hisashi Handa
Graduate School of Natural Science and Technology
Okayama University
Okayama, Japan
Email: handa@sdc.it.okayama-u.ac.jp

***Abstract*—EDA-RL, Estimation of Distribution Algorithms for Reinforcement Learning Problems, have been proposed by us recently. The EDA-RL can improve policies by EDA scheme: First, select better episodes. Secondly, estimate probabilistic models, i.e., policies, and finally, interact with the environment for generating new episodes. In this paper, the EDA-RL is extended for Multi-Objective Reinforcement Learning Problems, where reward is given by several criteria. By incorporating the notions in Evolutionary Multi-Objective Optimization, the proposed method is enable to acquire various strategies by a single run.**

***Keywords*-Estimation of Distribution Algorithms; Reinforcement Learning Problems; Evolutionary Multi-Objective Optimisation;**

I. Introduction

The purpose of Multi-objective reinforcement learning problems is to acquire policies in order to maximize a total amount of reward, where is given by several criteria. Such criteria are usually in trade-off relationships, for instance, speed and safety. Therefore, it is quite difficult to maximize all the kinds of reward simultaneously. Conventional reinforcement learning algorithms have a policy, which is often represented by the value function or the state-action value function. Hence, weighted approach for such reward is adopted: Several criteria are unified with their weights. Unfortunately, it is quite difficult to design the unification. The unification affects the characteristic of reinforcement learning problems, whilst there is no way to decide the weights such that acquired behaviors are familiar with designers' intuitions. Desirable behaviors are different with ones by optimal policies on weighted reward. For instance, suppose that designers of agents set weights to 0.6 and 0.4 for "speed" and "safety" criteria, respectively. Unfortunately, it is rarely occurred that agents acquire rewards at the same ratio as in weights. In addition, the changes of weights imply that learning process is re-carried out since it will yield the different problem instance.

Estimation of Distribution Algorithms (EDAs) are a promising evolutionary computation method. By making use of probabilistic models, EDAs can outperform conventional evolutionary computations. The EDA-RL, proposed by us, is an extension of EDA to solve reinforcement learning problems [1]. One of the primary features of the EDA-RL is direct estimation of reinforcement learning agents' policies by using Conditional Random Fields. On the other hand, conventional reinforcement learning algorithms estimate state values or state-action values, instead of a policy. The EDA-RL can directly estimate a policy as a conditional probability distribution in a statistical way. Another feature is that a kind of undirected graphical probabilistic model is used in the EDA-RL. Recently, Markov Networks have often been used by the EDA community [2][3][4]. However, they are using Markov Networks to optimize functions, not to solve reinforcement learning problems.

Evolutionary Multi-objective Optimization (EMO) has been broadly studied not only in evolutionary computation community but also in engineering design because it is able to search for Pareto set by a single run. In this paper, the EDA-RL is extended to multi-objective reinforcement learning problems by incorporating the notion of EMO: Firstly, Pareto fronts of episodes are constituted during evolution by using dominance relations in EMO. At each generation, then, the Pareto front is divided into several clusters. For each cluster, a policy is estimated. Therefore, the proposed method can learn several policies by a single run.

II. Multi-objective Reinforcement Learning Problems

The reinforcement learning problem is the problem of learning from interactions with an environment. Such interactions are composed of perceptions of the environment and actions that affect both agents and environments. Agents try to maximize the total reward received from the environment as consequences of their actions. In other words, the task for agents is to acquire a policy $\pi(s, a)$ which maximizes the prospective reward. The variables s and a in $\pi(s, a)$ denote states recognized by agents, and actions taken by agents, respectively.

$$R = \sum_{t}^{\infty} r(t)$$

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.92

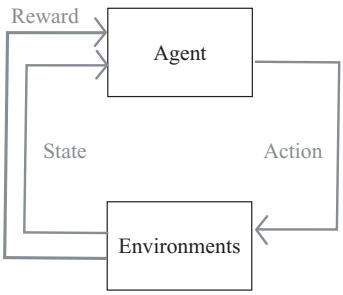

Figure 1. Reinforcement Learning Problems

The policy can be defined by using a probabilistic formulation such as $P(a|s)$. Most Reinforcement Learning algorithms learn value functions, e.g., a state-action value function $Q(s,a)$ and a state value function $V(s)$, instead of estimating the policy $\pi(s,a)$ directly. That is, in the case of conventional Reinforcement Learning algorithms, the policy $\pi(s,a)$ is approximated by value functions. In this paper, we adopt Conditional Random Fields to estimate the policy $\pi(s,a)$ from selected episodes in the previous generation.

In the case of multi-objective reinforcement learning, reward R_i is given by each criterion i $(1,\ldots,n)$. Agents try to maximized all kinds of reward:

$$\max(R_1,\ldots,R_n)$$

As in usual multi-objective optimization, such criteria are in trade-off relationships. Therefore, in order to judge the effectiveness of policies or episodes, the following dominance relations are utilized: An episode e_1 **dominates** an episode e_2 iff $R_i^{e_1} \geq R_i^{e_2}(1,\ldots,n)$ and $R_i^{e_1} \neq R_i^{e_2}$ at least one criterion. In addition, a policy $\pi_1(s,a)$ dominates a policy $\pi_2(s,a)$ iff $E_{\pi_1(s,a)}[R_i] \geq E_{\pi_2(s,a)}[R_i](1,\ldots,n)$ and $E_{\pi_1(s,a)}[R_i] \neq E_{\pi_2(s,a)}[R_i]$ at least one criterion.

III. Conditional Random Fields

A. Overview

Conditional Random Fields (CRFs) were first proposed by Lafferty *et al.* in order to apply statistical learning to segmentation problems in text processing [5]. The CRFs can handle a conditional distribution $P(\mathbf{y}|\mathbf{x})$ with an associated graphical structure. A notable feature of the CRFs is that they model such conditional distributions whereas Hidden Markov Models, which are traditionally used in broad areas such as voice recognition and text processing, estimate joint probability distributions $P(\mathbf{x},\mathbf{y})$. This implies that we do not have to consider the probabilistic distribution of inputs $P(\mathbf{x})$ since $P(\mathbf{x},\mathbf{y}) = P(\mathbf{y}|\mathbf{x}) \cdot P(\mathbf{x})$. In general, the probability distribution of the inputs $P(\mathbf{x})$ is unknown. Moreover, in the case of Reinforcement Learning Problems, it depends on agents' actions.

CRFs can be regarded as a sort of Markov Network since CRFs use an undirected graph model to represent a probabilistic distribution. That is, variables in the problem are factorized in advance. Each clique (factor) is associated with a local function.

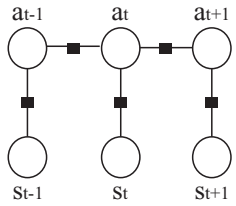

Figure 2. Graphical model of a Linear-Chain CRF

In the case of a log-linear Markov Network, the joint probability $P(\mathbf{y},\mathbf{x})$ can be represented as follows:

$$P(\mathbf{y},\mathbf{x}) = \frac{1}{Z}\prod_A \Psi_A(\mathbf{y}_A,\mathbf{x}_A), \tag{1}$$

where Ψ_A denotes a local function for a set of variables $\mathbf{y}_A,\mathbf{x}_A \in \{\mathbf{y},\mathbf{x}\}$. Z is a normalizing factor which ensures that the distribution sums to 1:

$$Z = \sum_{\mathbf{y},\mathbf{x}}\prod_A \Psi_A(\mathbf{y}_A,\mathbf{x}_A). \tag{2}$$

On the other hand, CRFs can handle the following conditional probabilities $P(\mathbf{y}|\mathbf{x})$:

$$\begin{aligned} P(\mathbf{y}|\mathbf{x}) &= \frac{P(\mathbf{y},\mathbf{x})}{\sum_{\mathbf{y}'}P(\mathbf{y}',\mathbf{x})} = \frac{\frac{1}{Z}\prod_A \Psi_A(\mathbf{y}_A,\mathbf{x}_A)}{\sum_{\mathbf{y}'}\frac{1}{Z}\prod_A \Psi_A(\mathbf{y}'_A,\mathbf{x}_A)} \\ &= \frac{\prod_A \Psi_A(\mathbf{y}_A,\mathbf{x}_A)}{\sum_{\mathbf{y}'}\prod_A \Psi_A(\mathbf{y}'_A,\mathbf{x}_A)} \\ &= \frac{1}{Z(\mathbf{x})}\prod_A \Psi_A(\mathbf{y}_A,\mathbf{x}_A), \end{aligned} \tag{3}$$

where

$$Z(\mathbf{x}) = \sum_{\mathbf{y}'}\prod_A \Psi_A(\mathbf{y}'_A,\mathbf{x}_A).$$

B. Linear-chain CRF

This subsection introduces the linear-chain CRF which is widely used in the natural text-processing area. The linear-chain CRF is one of the simplest CRFs. Here, input and output variables $\mathbf{x}$, $\mathbf{y}$ in the previous subsection are substituted by a sequence of states $\mathbf{s} = \{s_1, s_2, \ldots, s_T\}$ and a sequence of corresponding actions $\mathbf{a} = \{a_1, a_2, \ldots, a_T\}$, respectively. The linear-chain graphical model in this case is depicted in Figure 2. Each circle (node) in this figure represents a state or an action at the corresponding time step. Each line with a black square in the middle of the line indicate that nodes are associated with a local function Ψ.

As we can see from this figure, the linear-chain CRFs factorize an episode, i.e., a sequence of state-action pairs $(\mathbf{s},\mathbf{a})$ into state-action pairs (s_t,a_t) and transitions of actions

(a_{t-1}, a_t). The local function for each time step is defined as follows:

$$\Psi_k(\mathbf{a}, \mathbf{s}) = \begin{cases} \exp(\lambda_k \cdot u_k(a_t, s_t)) & k \leq K' \\ \exp(\lambda_k \cdot v_k(a_{t_{k-1}}, a_t)) & K' < k <\leq K, \end{cases}$$

where $u_k(a_t, s_t)$ and $v_k(a_{t_{k-1}}, a_t)$ are feature functions and K' is the total number of possible combinations of states and actions (a, s). K is the total number of the feature functions. The values $u_k(a_t, s_t)$ and $v_k(a_{t_{k-1}}, a_t)$ are set to 1 if the corresponding state-action pair and action-action pair are observed, respectively. λ_k denotes a parameter for a factor k. Equation (3) can be rewritten using the above notation as follows:

$$P(\mathbf{a}|\mathbf{s}) = \frac{1}{Z(\mathbf{s})} \exp\left\{\sum_{t=1}^{T}\left\{\sum_{k=1}^{K'} \lambda_k u_k(a_t, s_t) + \sum_{k=K'+1}^{K} \lambda_k v_k(a_{t-1}, a_t)\right\}\right\}. \quad (4)$$

C. Parameter Estimation

Suppose that N episodes $(\mathbf{s}^{(i)}, \mathbf{a}^{(i)}),\ (i = 1, \ldots N)$ are acquired to estimate the policy and each episode $(\mathbf{s}^{(i)}, \mathbf{a}^{(i)})$ is composed of a series of state-action pairs:

$$(\mathbf{s}^{(i)}, \mathbf{a}^{(i)}) = \{(s_1, a_1), (s_2, a_2), \ldots (s_{T_i}, a_{T_i})\},$$

where T_i denotes the length of episode i. The log likelihood method is used to estimate the parameters $\theta = (\lambda_1, \ldots, \lambda_K)$:

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{N} \log P(\mathbf{a}^{(i)}|\mathbf{s}^{(i)}) \\ &= \sum_{i=1}^{N}\sum_{t=1}^{T_i}\left\{\sum_{k=1}^{K'} \lambda_k u_k(a_t, s_t) + \sum_{k=K'+1}^{K} \lambda_k v_k(a_{t-1}, a_t)\right\} \\ &\quad - \sum_{i=1}^{N} \log Z(\mathbf{s}^{(i)}) \end{aligned}$$

In order to calculate the optimal parameter θ, the partial derivative of the above equation is used. For $k \leq K'$, we can calculate the partial derivative as follows:

$$\begin{aligned} \frac{\partial l}{\partial \lambda_k} &= \sum_{i=1}^{N}\sum_{t=1}^{T_i} u_k(a_t, s_t) - \sum_{i=1}^{N} \frac{(Z(\mathbf{s}^{(i)}))'}{Z(\mathbf{s}^{(i)})} \\ &= \sum_{i=1}^{N}\sum_{t=1}^{T_i} u_k(a_t, s_t) - \sum_{i=1}^{N}\sum_{a'_t} P(a'_t|\mathbf{s}^{(i)}) \cdot u_k(a'_t, s_t). \end{aligned}$$

The first and second terms of the right hand side of this equation denote the number of observations of the factor $u_k(a_t, s_t)$ and the expected value of the factor $u_k(a_t, s_t)$ under the current value of the parameter θ, respectively. Therefore, this derivative indicates that the parameter λ_k is modified along that expected value closest to the actual observation in the selected episodes. Further descriptions needed to calculate $l(\theta)$ and $\frac{\partial l}{\partial \lambda_k}$, i.e., $Z(\mathbf{s}^{(i)})$ and $P(a'_t|\mathbf{s}^{(i)})$ are described in appendix.

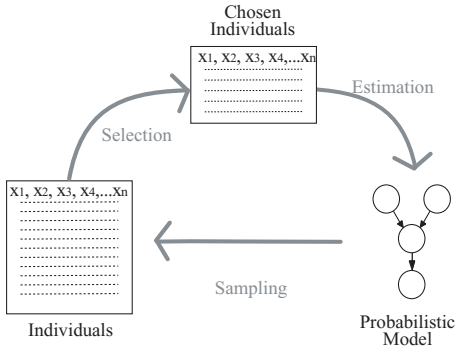

Figure 3. Search Process by Estimation of Distribution Algorithms

IV. Estimation of Distribution Algorithms

Estimation of Distribution Algorithms are a class of evolutionary algorithms which adopt probabilistic models to reproduce individuals in the next generation, instead of conventional crossover and mutation operations. The probabilistic model is represented by conditional probability distributions for each variable. This probabilistic model is estimated from the genetic information of selected individuals in the current generation. Figure 3 shows the general process of EDAs. As depicted in this figure, the main calculation procedure of the EDAs is as follows:

1) Firstly, N individuals are selected from the population in the previous generation.
2) Secondly, the probabilistic model is estimated from the genetic information of the selected individuals.
3) A new population whose size is M is then sampled by using the estimated probabilistic model.
4) Finally, the new population is evaluated.
5) Steps 1)-4) are iterated until the stopping criterion is reached.

V. EDA-RL

A. Calculation Procedure of EDA-RL

Figure 4 depicts the calculation procedure of the proposed method, i.e., EDA-RL. The procedure is summarized as follows:

1) The initial policy $\pi(s, a)(= P(a|s))$ is taken to be a uniform distribution. That is, according to the initial policy $\pi(s, a)$, agents move randomly.
2) Agents interact with the environment by using policy $\pi(s, a)$ until Ts episodes are generated. An episode is a sequence of pairs (state s_t, action a_t).
3) The best episode, i.e., the episode with greatest reward, among the Ts episodes in the previous step is stored in the episode database. Return to 2) until the number

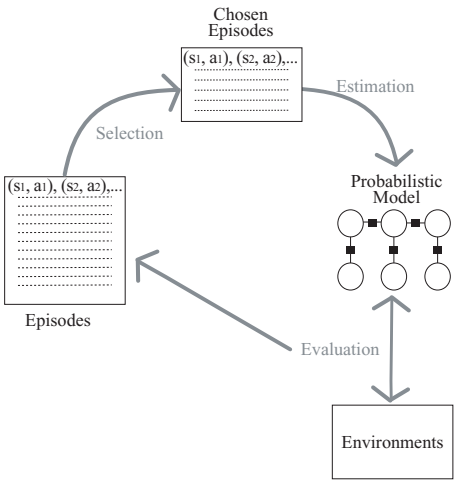

Figure 4. Search Process of an EDA-RL

of chosen episodes reaches a predefined constant value C_d.

4) A new policy $\pi(s, a)$ for the set of episodes in the database is estimated by CRF. After the estimation, all episode data in the database is erased.
5) Return to 2) until terminal conditions are met.

One of the main differences between conventional EDAs and EDA-RL is the use of the estimated probabilistic model. Conventional EDAs employ a probabilistic model to generate individuals, i.e., solutions for given problems. On the other hand, the probabilistic model estimated by CRF represents the policy $\pi(s, a)$. In other words, the probabilistic model denotes the solution itself.

We note that in 1) in the above procedure, we do not have to assume a uniform distribution for the initial policy. That is, if there is plenty of observation data available, e.g. play-data by humans, or episodes from the conventional approach, such data can be used to generate an initial policy. This means that the proposed method can easily incorporate human knowledge and can improve on it by using an evolutionary approach.

B. Extension of EDA-RL to Multi-Objective Reinforcement Learning Problems

In comparison with conventional EDA-RL in the previous subsection, what we extend for Multi-Objective Reinforcement Learning Problems is that 1) several policies $\pi_j(s, a)(j = 1, \ldots, m)$ are employed, 2) fitness assignment and selection method of episodes are used ones in Zittler's SPEA2 [6], and 3) clustering algorithm is carried out to selected episodes in order to separate such episodes into several learning sets for policies.

1) The initial policies $\pi_j(s, a)$ are taken to be a uniform distribution.
2) Agents interact with the environment by using one of policies $\pi_j(s, a)$ until noc episodes are generated.
3) Fitness F_l for the l^{th} episode is assigned by using the fitness assignment algorithm in SPEA2, and better episodes are selected from the noc episodes and episodes selected in the previous generation.
4) Episodes selected in the previous step are clustered by using K-means method.
5) New policies $\pi_j(s, a)$ for the clustered set s_j of episodes are estimated by CRF.
6) Return to 2) until terminal conditions are met.

C. Interaction with the Environment

As mentioned in the previous subsection, a probabilistic model formed from selected episodes represents the policies of agents, which decide the actions for the current situation. The original CRFs were used in text-processing and bio-informatics, where several outputs $(y_1, y_2, \ldots, y_T)$ have to be determined simultaneously for corresponding inputs $(x_1, x_2, \ldots, x_T)$. Unfortunately, reinforcement learning problems are not problems of this type. That is, at every time step t, agents need to decide on their outputs. Therefore, we employ factors related to the current state and the previous action, which are then used to decide on the output[1]. The factors used to choose an action a_t for state s_t are $u_k(a_t, s_t)$ and $v_k(a_{t-1}, a_t)$. Hence, from equation (4)

$$P(a_t|s_t, a_{t-1}) = \frac{1}{Z(s_t)} \exp\left\{\sum_{k=1}^{K'} \lambda_k u_k(a_t, s_t) + \sum_{k=K'+1}^{K} \lambda_k v_k(a_{t-1}, a_t)\right\}, \quad (5)$$

where $Z(s_t)$ can be calculated as follows:

$$Z(s_t) = \sum_{a'_t} \exp\left\{\sum_{k=1}^{K'} \lambda_k u_k(a'_t, s_t) + \sum_{k=K'+1}^{K} \lambda_k v_k(a_{t-1}, a'_t)\right\}.$$

By using this probability and the following equation, an action a_t at time step t is chosen.

$$a = \text{argmax } P(a_t|s_t, a_{t-1}).$$

Moreover, an ϵ-greedy method is used in this paper so that, with probability ϵ, a new action is randomly chosen, instead of the above action. The parameter ϵ is set to be 0.05.

VI. Experiments

A. Problem Settings

Probabilistic Transition Problems and the Perceptual Aliasing Maze Problem are introduced in order to investigate the effectiveness of the proposed method.

[1]On the other hand, in building the probabilistic model, we take account of the whole sequence of (state, action) pairs.

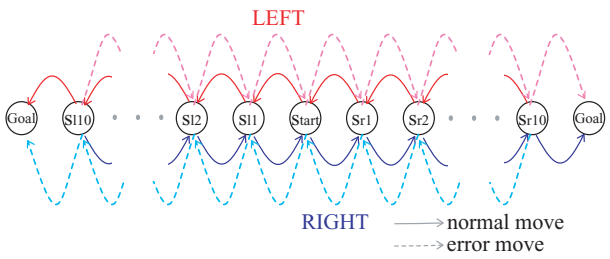

Figure 5. A depiction of Probabilistic Transition Problems in the case where there are 10 intermediate states

As depicted in Figure 5, a start point is located at the center position. Agents can take two actions at any states: a left or a right move. By taking these actions, agents move to an adjacent state. However, with probability P_w, agents move in the opposite direction to their chosen action; this is called an error move. There are two goals: both goals give two symmetrical reward to agents

$$\begin{aligned} R_{left,1} &= 100/count + nos \\ R_{left,2} &= S_{MRS} \\ R_{right,1} &= S_{MLS} \\ R_{right,2} &= 100/count + nos, \end{aligned}$$

where S_{MRS} and S_{MLS} denote the absolute value of the index number of the most right and left state in the episode, respectively. nos denote the number of intermediate states. This paper sets nos to be 20. For instance, consider "optimal policy" in terms of $R_{left,1}$. The "optimal behavior" is that all the actions are left in the episode. In this case, $R_{left,1}$ is $100/nos + nos$, and $R_{left,2}$ is 0 if there is no error move. Now, suppose that an agent moves twice to right, then go back to left goals as in the "optimal policy". In this case, $R_{left,1}$ is $100/(nos + 4) + nos$, and $R_{left,2}$ is 2 if there is no error move. Agents recognize two kinds of reward $R_{*,1}$ and $R_{*,2}$. When agents reach one of the goals, the episode is finished.

B. Results

Figure 6 shows the experimental results of the probabilistic transition problems by the conventional EDA-RL and the proposed method. The graphs on the left and right sides denote error move probability $P_W = 0.1$ and $P_W = 0.3$, respectively. The number of intermediate states is set to be 20. These graphs represent all the episodes during evolution, except for fail episodes, which cannot reach to either of goals until 200 steps. The distribution of episodes in the proposed method is more broad than the one in the conventional EDA-RL. The proposed method acquired two strategies such either of two kinds of reward can mainly be received.

ACKNOWLEDGEMENT

This work was partially supported by the Grant-in-Aid for Exploratory Research and the Grant-in-Aid for Scientific Research (B) of MEXT, Japan (18656114 and 21360191), and by the research grant of Hayao Nakayama Foundation (H19-A51).

VII. CONCLUSIONS

In this paper, EDA-RL, Estimation of Distribution Algorithms for Reinforcement Learning Problems, proposed earlier, is extended to apply Multi-objective Reinforcement Learning Problems. In the proposed method, better episodes are selected by using dominance relations as in SPEA2. Then selected episodes are clustered by using clustering algorithms, and are used to estimate policies. By incorporating the scheme in EMO and separately estimating policies, the proposed method is enable to acquire several strategies by a single run. The experimental results of the probabilistic transition problems confirm this.

REFERENCES

[1] H. Handa, "EDA-RL: estimation of distribution algorithms for reinforcement learning problems," in *Proc. of the 2009 ACM/SIGEVO Genetic and Evol. Comput. Conf.*, 2009, pp. 405–412.

[2] R. Santana, "Estimation of distribution algorithms with Kikuchi approximations," *Evol. Comput.*, vol. 13, no. 1, pp. 67–97, 2005.

[3] S. K. Shakya, J. A. W. McCall, and D. F. Brown, "Incorporating a metropolis method in a distribution estimation using markov random field algorithm," in *Proc. of 2005 IEEE Congress on Evol. Comput.*, vol. 3, 2005, pp. 2576–2583.

[4] ——, "Using a markov network model in a univariate EDA: An emperical cost-benefit analysis," in *Proc. of 2005 Genetic and Evol. Comput. Conf.*, 2005, pp. 727–734.

[5] J. Lafferty, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of 18th International Conference on Machine Learning*. Morgan Kaufmann, 2001, pp. 282–289.

[6] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," in *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, K. Giannakoglou *et al.*, Eds. International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.

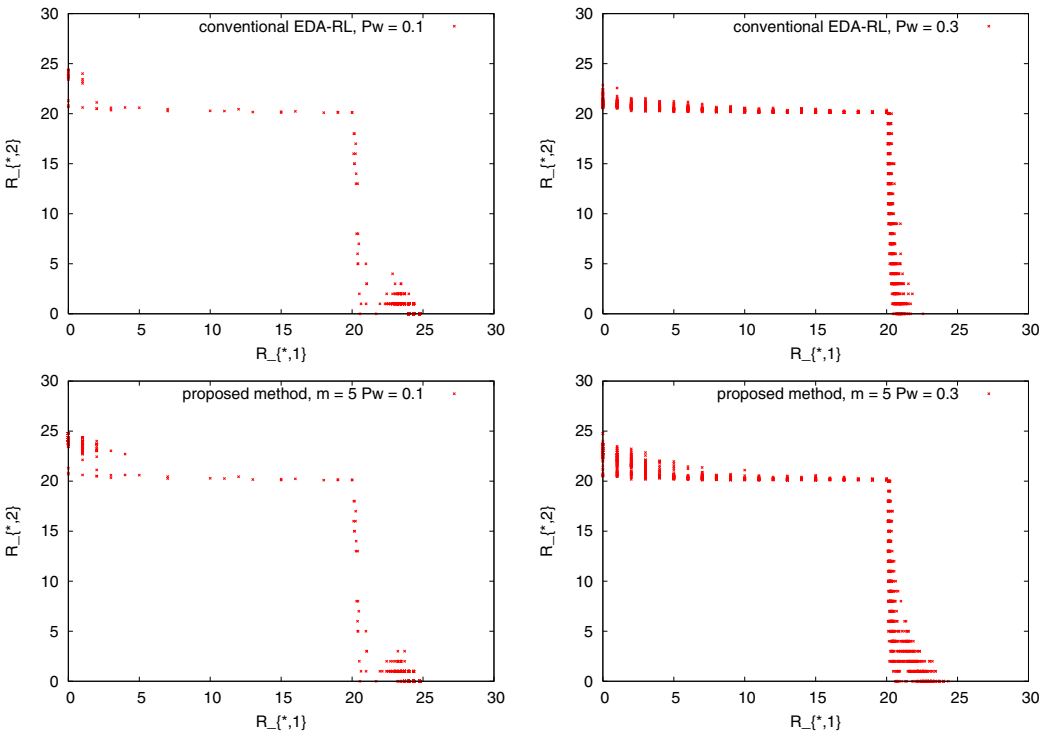

Figure 6. Experimental results: conventional EDA-RL (UPPER) and the proposed method(LOWER); error move Probability $P_w = 0.1$ (LEFT) and $P_w = 0.3$ (RIGHT)