2010 Sixth International Conference on Signal-Image Technology and Internet Based Systems

A Method to Control Parameters of Evolutionary Algorithms by using Reinforcement Learning

Yoshitaka Sakurai, Kouhei Takada, Takashi Kawabe, Setsuo Tsuruta
School of Information Environment, Tokyo Denki University, Chiba, Japan,
{ysakurai, kawabe, tsuruta}@sie.dendai.ac.jp

***Abstract*— A search method using an evolutionary algorithm such as a genetic algorithm (GA) is very effective if the parameter is appropriately set. However, the optimum parameter setting was so difficult that each optimal method depending on each problem pattern must be developed one by one. Therefore, this has required special expertise and large amounts of verification experiment. In order to solve this problem, a new method called "adaptive parameter control" is proposed, which adaptively controls parameters of an evolutionary algorithm. However, since this method just increases the selection probability of a search operator that generated a well evaluated individual, this is apt to be a shortsighted optimization method. On the contrary, a method is proposed to realize longsighted optimal parameter control of GA using reinforcement learning. However, this method does neither consider the calculation cost of search operators nor multipoint search characteristics of GA. This paper proposes a method to efficiently control parameters of an evolutionary algorithm by using the reinforcement learning where the reward decision rules are elaborately incorporated under the consideration of GA's multipoint search characteristics and calculation cost of the search operator. It is expected that this method can efficiently learn parameters to optimally select search operators of GA for approximately solving Travelling Salesman Problems (TSPs).**

***Keywords*— Delivery Route Scheduling System, Traveling Salesman Problems (TSP), Reinforcement Learning, Genetic Algorithm (GA)**

I. INTRODUCTION

Evolutionary algorithm including Genetic Algorithm (GA) [1] is a (approximate solution) search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. . Regarding these search heuristic, the optimal policy generally differs depending on the problem patterns such as those scale, response time and required accuracy. Concretely, in case of TSP (Traveling Salesman Problem) [2], visiting location distribution and arrival time constraint are critical factors as problem patterns. Therefore, the optimization methods of different problems and patterns had to be developed individually according to the actual sites and domains [3][4].

In the approximation methods such as evolutionary algorithm, currently there is no general purpose theory for parameter settings and heuristics appropriate for all problems, though partly, because of their use of random numbers. Generally speaking, approximate search methods show high efficiency if the parameter is set appropriately. However, appropriate parameter setting is not easy [5].

We have been researching into real-time but highly precise methods to search approximate solutions of middle and large scale TSPs (from tens to less than two thousands of cities or locations) in practical application [3][4]. In the process, it was found that some types of heuristics are strong for some problems but weak against others. Therefore, problems seem to be efficiently and generally solved if a set of parameter values or heuristics fit to each problem is selected automatically or at least semi-automatically. However, such selection is made among so many parameter value sets that this requires a large amount of experiment, and the efficiency is critical.

On the other hand, in the research of reinforcement learning (RL), this frame work having state concept was quite useful for effective parameter learning of experts' technical abilities from long term points of view [6]. In addition, if the fitness value of individuals is taken as reward, RL can be regarded as similar to GA in its evolution /learning structure, which causes high conceptual and structural compatibility between RL and GA. From these two points mentioned above, it is expected that reinforcement learning can dynamically control GA parameters to obtain optimality.

As for previous studies of parameter setting on evolutionary algorithms, there are two types according to Eiben[7]. One is to set the value of a parameter before the run of the algorithm and then running the algorithm using these values, which remain fixed during the run, so-called, parameter tuning. The other is methods for changing the value of a parameter during the run, "parameter control". Among parameter tuning types, a method is proposed to test various parameter settings in accordance with the fixed schedule [8] and set one of them as the best parameter based on the experimental results. Indeed, this method can set all parameters at once. However, since this type does not use information in the search process, it is not efficient, which is different from the following adaptive parameter control. Yet, the parameter can be switched neither dynamically nor adaptively during the search: the parameter value always remains fixed.

978-0-7695-4319-2/10 $26.00 © 2010 IEEE
DOI 10.1109/SITIS.2010.22

IEEE computer society

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 28,2023 at 16:49:58 UTC from IEEE Xplore. Restrictions apply.

As parameter control methods, an adaptive one is proposed, which enables to adaptively control the parameter of an evolutionary algorithm. For examples, literatures [9] and [10] correspond to this type. Putting weight on each of GA's search operators such as crossover and mutation, they set parameters based on the evaluation of search points (namely child individuals generated by these operators). However, since these methods need to give rewards according to immediate estimation of selection results, this can be used only for short-sighted parameter setting of operators such as crossover or mutation.

As a method to optimize GA's parameter on a long term basis, the one using RL is proposed. As an initial study, there is a method called RL-GA [11], which learns a policy to select the operators using Q-learning [16]. Besides, considering a practical application use such as real-time response, SCGA [12] is proposed, which learns a policy using Sarsa that can do without training first. However, these methodologies neither consider the calculation cost nor GA's characteristics as a multi-point search method. Thus, efficiency should be improved in these method.

In this paper, a method is proposed, which enables to efficiently control parameters of an evolutionary algorithm, using RL that have a rewarding considering the calculation cost of search operators and GA's multi-point search feature. Thereinafter, GA implementing this idea is called "MT-RLcGA: Multi-point and Time-cost considered Reinforcement Learning Control Genetic Algorithm".

The next section describes problems of methods to apply new learning methods such as adaptive parameter control and Reinforcement Learning. Then, their solving approaches are described. The section 3 describes the algorithm MT-RLcGA, its behavior, and the expected effects. Finally, section 4 concludes this paper.

II. PROBLEMS

A. Genetic Algorithm

Genetic Algorithm (GA) [1] is an approximate solution search algorithm modeled upon the evolution of living things. Firstly, by its initial individual generation operator, GA generates individuals (initial populations) that represent solution candidates as the component genes. Then, based on initial populations, using search operators such as crossover and mutation, GA generates new individuals (new solution candidates), and selects next populations from them based on their fitness values. Through the iteration of such generation and selection, GA searches the optimal solution.

The latter parts of this paper discuss the case where GA solves a TSP (Traveling Salesman Problem, TSP) [13] famous for a combinatorial optimization problem. Especially, a symmetrical (non-directed) Euclidean TSP is assumed in this paper. TSP is a problem to find the shortest path to travel around multiple locations. This can be formulated as follows:

The delivery route is represented by a weighted complete graph $G=(V,E,w)$. V is a node (vertex) set. A node $v_i \in V$ ($i=1,\ldots,N$) represents a location (address) for delivery. $N=|V|$ is the number of nodes. $E \subseteq V^2$ is a set of edges. An edge e_{ij} represents a route from v_i to v_j. w is the edge weight set. An edge weight d_{ij} represents a distance (or the cost to go) from node v_i to node v_j. Here, we presume $d_{ij} = d_{ji}$, due to the assumption of an Euclidean TSP.

The problem to find the shortest or minimal-length Hamilton path in such a graph $G=(V,E,w)$ is called a Traveling Salesman Problem (TSP). The Hamilton path is called an practicable or feasible solution. The shortest one is called an optimal solution. Each individual to be a solution candidate has (or is) a chromosome representing a tour (a delivery route) in TSP. Each gene of a chromosome represents a node number or ID number (identification number) of a delivery destination). A chromosome is a sequence of nodes or genes. This sequence represents a tour order as shown in Fig.1.

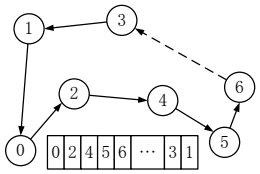

Fig. 1. Chromosome representing delivery route

B. Parameter Control

Various efficient search methods are proposed, using heuristics for search operators to generate solution candidates (Hamiltonian circuit near optional solution) of such TSP [14]. Yet, there are also some proposed methods which use plural search operators to make search effective. We can not tell which search operators work effectively, since they differ from each other depending on the types of problems in question, the original individuals, and the states of population. Therefore, it is indispensable to adjust their parameters.

We have been researching into real-time and highly precise methods to search approximate solutions of middle and large scale TSPs (from tens to less than two thousands of cities or locations) in practical application [3][4]. In the process, it was found that some types of heuristics are strong for some problems but weak against others. Therefore, problems seem to be efficiently and generally solved if a set of parameter values including their representing types of heuristics fit to each problem is selected automatically or at least semi-automatically. However, such selection is made among so many types of each value of parameters and heuristics that this requires a large amount of experiments, and the efficiency is critical.

On the other hand, in the research of RL, this frame work having the state concept was quite useful to learn expertise as a set of parameters with long-term effects remaining [6]. In addition, if the fitness value of each individual is taken as

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 28,2023 at 16:49:58 UTC from IEEE Xplore. Restrictions apply.

reward, RL is similar to GA in its evolution and learning structure, which causes high conceptual and structural compatibility. Due to these two reasons, it is considered that RL could control parameters of GA dynamically and effectively from a long sight view, aiming at optimization of individuals.

C. Reinforcement Learning(RL)

RL is a kind of machine learning methods that deal in a problem how an agent observes the current state and decides its actions taken upon under a certain environment. Agents receive their reward through the environment by selecting their actions. RL agent learns a policy for agents to be given the highest reward through a series of actions.

The agent and the environment interacts each other at each stage of scattered time steps (t=0,1,2$\cdots$). The agent receives some state expression $s_t \in S$ (S is a group of possible states) of the environment at each time step t, and selects an action $a_t \in A(s_t)$ based on it. ($A(s_t)$ is a group of actions selectable at a state s_t). After finishing each time step, the agent receives the reward $r_{t+1} \in R$ as a result of its action, and proceeds to a new state. The probability mapping of selecting a possible action under the state s at each time step t, is called a policy $\pi_t(s, a)$. RL learns a policy to finally maximize the amount of received reward. The amount of reward finally received is given by means of introducing the concept of discount. This is formulated as the expression (0.1).

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_t + k + 1 \qquad (0.1)$$

γ is a parameter called discount rate, which is $0 \le \gamma \le 1$ and specifies how worthy the future reward is in the current situation. The value when action a is selected following the policy π under state s, is indicated as $Q^{\pi}(s, a)$. This defines the expected reward when following the policy π.

$$Q^{\pi}(s,a) = E_{\pi}\left\{R_t \mid s_t = s, a_t = a\right\} \qquad (0.2)$$

Q^{π} is called the action value function as for policy π. In Q-learning [16] famous for RL, this action value function is approximated or learned by iterative improvement as follow.

$$Q(s_t,a_t) \leftarrow (1-\alpha)Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1},a)\right] \qquad (0.3)$$

α is a learning rate which is $0 \le \alpha \le 1$. In Q-learning, an off-policy type algorithm that independently and directly approximates optimal action value function Q^* is used as a policy. As an on-policy type algorithm, Sarsa is proposed, which approximates the optimal action value function mentioned above.

$$Q(s_t,a_t) \leftarrow (1-\alpha)Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1},a_{t+1})\right] \qquad (0.4)$$

D. Problem with conventional method

As for methods to adaptively control parameters of evolutionary algorithms or adaptive parameter control, some methods are proposed, which increase selection probability of search operators who generated more optimal individuals [9][10]. However, those methods control parameters relying just upon the instantaneous search result that is a new individual generated by the search operator. Thus, there is a possibility that its optimization is short sighted. Yet, due to the necessity of indicators in order to directly evaluate, such method can be applied to only operators such as cross-over or mutation that are directly related to generate search points or child individuals.

On the other hand, parameter control methods using RL are proposed. RL is an algorithm that learns a policy to maximize the amount of received reward in a long term prospect. This makes it possible to gain optimal parameter control policies in a long term prospect.

As an initial research adapting RL to GA, there is RL-GA that learned selection policy of search operators using Q-learning. This utilizes Q-learning that is an off-policy type method requiring a training period before the run of the algorithm. But in the practical application, such training can be unfavorable in some cases [12].

To solve such a problem, SCGA using Sarsa(0) which is an on-policy type RL method, is proposed (see literature [12]). On-policy method does not require separation of training and action: this is proposed for such practical use as different delivery points with each round trip.

In such algorithm, the reward is considered as the difference between parent's fitness and child's one. But the calculation cost differs from one GA operation to another: it is natural that the results of costly calculation are better than others. Such a way of rewarding would be effective in case a good solution (final) is required taking no account of time. But it is not favorable when optimal solution is required within the shortest length of time. Therefore, the use of improvement rate per unit hour for reward is suggested to be necessary.

Yet, in literatures [11] and [12] reward value is calculated with each individual generation, updating the action value function each time. However, as GA is a multi-points search method, more efficient policy could be learned through reward calculation groups' (multiple search points) search results, instead of reward calculation with each individual's reward.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 28,2023 at 16:49:58 UTC from IEEE Xplore. Restrictions apply.

III. PROPOSED METHOD

In this section, by rewarding considering GA's multi search feature as well as calculation cost, we propose a parameter control method based on RL to realize efficient parameter control. We call this GA algorithm implemented with this multi-point GA "Multi-point and Time cost regarded Reinforcement Learning Control Genetic Algorithm, MT-RLcGA".

This algorithm based on RL-GA [11], adopts a step to renew reward determining rule and action value function that we propose in the real section. We used Sarsa for renewal, which is the same as SCGA [12] for action value function.

A. MT-RLcGA

Our proposed method MT-RLcGA controls selection of GA search operator through RL agent.

Our GA is implemented with plural search operators (crossover, mutation operators) which generate child individuals. This search operator is selected depending on the state.

Through the evaluation of the child individual, the agent receives reward and learns a search operator's selection policy. Among generated children's individual groups and parents' individual groups, an upper level individual group is selected, based on the adaptivity, which becomes a parents' individual group of the next generation. Through repeating this process, solutions are searched.

1) Learning Algorithm

We propose two types of algorithm where learning timing of agents is different. One of them remains its agent's action value function per generation of individual, that is, Q evolution per individual.

The other one renews its action value function after generation of all individuals, that is, group Q renewal. The algorithm of RLcGA using Q renewal per group is shown in fig. 2.

As to RL-GA, SCGA Q renewal per individual is used. In these researches, evaluation differences between parents and children are used as reward. In this case, agents learn so that the sum of improved rate of solutions is possible since GA realizes multi-point search with plural individuals.

For this purpose, the optimization of parameter's multi-point search is critical. Namely, it is not necessarily the effect of the selection rate of operators which generate good individuals that improves the performance as a whole. Thus, by evaluating the whole generation with the influence of other operators' action taken into account, as well as the generation per individual, it may be possible for us to gain more effective GA multi-point policy. Yet, Sarsa is adopted for the rule renewal of action value function so that it can learn a policy by on-policy.

Initialize state-value function $Q(s, a)$
Generate initial population
  s←current states
  Repeat until terminal generation:
    Repeat until finishing M children generation:
      Select action a based on states s ... equation(0.8)
      Start measuring computational time
      Select parents using a method of a
      Generate child individual using an operator of a
      Calculate difference of the fitness value between the parent and the child
      End measuring computational time
      Record reward of each individual r_m ... equation(0.9)
    Calculate reward by population search r_{pop} from recorded rewards $\{r_m\}_{m=1\ldots M}$
    Weed out individuals that have low fitness value
    Observe reward r and state s' as a result of action a ... equation(0.12)
  Update state-value function $Q(s, a)$ ... equation(0.4)
  s←s'

Fig 1. Q value update algorithm by population

2) State

The state of the search, s_t is a vector which characterizes the genetic population. Same as RL-GA, the quantities are generation, average population fitness and entropy of the population fitness.

Generation measured on a logarithmic scale, log$t$: The logarithmic time was divided into 4 equally spaced (on a log scale) epochs for use in the state-action and eligibility trace tables.

Average population fitness normalized by the initial population fitness (equation(0.5)): The average fitness was binned into four intervals ([0,0.3),[0.3,0.4),[0.4,0.6),[0.6,1]) for use in the state-action and eligibility trace tables. M is the number of individuals. $f(x)$is fitness of individual x. f^t is fitness in generation t.

$$\bar{f}^t = \frac{\sum_{m=1}^{M} f(x_m^t)}{\sum_{m=1}^{M} f(x_m^0)} \qquad (0.5)$$

The entropy of the population fitness: The Shannon entropy H of the population fitness is equation(0.7).

$$p_m = f(x_m) / \sum_m^M f(x_m) \qquad (0.6)$$

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 28,2023 at 16:49:58 UTC from IEEE Xplore. Restrictions apply.

$$H = \sum_{m=1}^{M} p_m \log_2 \frac{1}{p_m} \quad (0.7)$$

The entropy is minimum when all individuals in the population have the same fitness, and it is maximum when fitness are uniformly distributed. In a similar manner to the generation and average fitness, the entropy was divided into three equally sized discrete states.

3) Action

The functions of the RL agent are to choose a parent selection method and a genetic operator from a range of a crossover or mutation operator. The RL agent selects the type of individuals to which the operator is applied. To this end, the population was divided into two classes, fit (F) and unfit (U). An individual in the fittest 10% of the working population is deemed fit, and all others are unfit. For each crossover operator, there are four parental combinations {FF, FU, UF, UU}. In similar manner, for each mutation operator there are two parental combinations {F, U}.

The RL agent select a genetic operator from seven type operators that are three crossover operators (MPX, GER, PMX) and tour mutation operators (MOVE, SCRM, INVR, SWAP) [11].

Actions are selected with probability given by the policy $\pi(s,a)$ derived from the state-action table via the softmax function (equation (0.8)).

$$\pi(s_t, a_t) = \frac{e^{\beta Q(s_t, a_t)}}{\sum_{a_t'=1}^{N} e^{\beta Q(s_t, a_t')}} \quad (0.8)$$

4) Reward

In RL-GA, SCGA, the reward for a crossover or mutation is calculated as the improvement in fitness of the best offspring over the best parent. However, because each genetic operator has different computational cost, high cost operators have advantage in RL-GA, SCGA. The reward setting in RL-GA and SCGA is for searching a strategy of not considering computational time. Such reward setting is unsuitable for searching a better strategy in the shorter time. In the proposed algorithm, to attempt the speed-up of the GA search, the improvement of fitness per unit time is used for the reward. For the *Q* value update by each individual, the reward for a crossover is equation(0.9).

$$r = \frac{\max\{f(x) \mid x \in parents\} - \max\{f(x) \mid x \in offspring\}}{\max\{f(x) \mid x \in parents\} \times Time_{\pi(s,a)}} \quad (0.9)$$

$Time_{\pi(s,a)}$ is computational time for generating new individuals by action "a" at state "s". For a mutation action there is only a single parent with a single offspring, so the reward is just the fractional improvement in fitness of the mutated individual.

For Q value update by population fitness, the reward is calculated from not only each individual's improvements but also population's improvements in one generation. r_m is the reward calculated from an individuals' improvement by one genetic operation. r_{pop} is the reward calculated from all individual's improvement in one generation. r_{pop} is calculated by equation(0.10) or equation(0.11). As defined in equation(0.12), the reward r is calculated by weighted summation of r_m and r_{pop}. ω is contributing rate of population evaluation.

$$r_{pop} = \left(\sum_{m=1}^{M} r_m\right) \Big/ M \quad (0.10)$$

$$r_{pop} = \max\{r_m \mid m = 1 \ldots \quad (0.11)$$

$$r = (1-\omega) r_m + \omega r_{pop} \quad (0.12)$$

B. Behaviors and effects of proposed method

We verify GA's operations and its effects from the proposed reward function. At first, we simulate the reward discount operation based on the computational time. We think about 2 cases on Q value update by each individual improvement.

CASE1: in state s_t , action a_t was selected, so that the fitness is improved 0.2 in 3CPU times. Continuously, in state s_{t+1}, action a_{t+1} was selected, so that the fitness is improved 0.1 in 1CPU times.

CASE2: in state s_t , action a'_t was selected, so that the fitness is improved 0.1 in 1CPU times. Continuously, in state s_{t+1}, action a_{t+1} was selected, so that the fitness is improved 0.1 in 1CPU times.

CASE2 obtained the fitness improvement same as CASE1 in shorter time. To simplify the calculation, we set the discount rate one. Calculating the reward from only improvements, CASE1 is 0.3, CASE2 is 0.2, so that CASE2 is evaluated unfairly and inferior. By contrast, calculating the reward from the improvements per one CPU time, CASE1 is 0.17, CASE2 is 0.2, so that CASE2 is evaluated fairly and superior.

This way, the genetic operator that works well by repeating small cost searches is evaluated appropriately, so that there is a possibility that the optimization strategy in short time is found.

Next, we think about the reward from population search results. The state-action (Q) value function is updated collectively for each generation.

We assume a GA method that has 2 type genetic operators. The genetic operator A has high optimization efficiency, but its diversity maintenance of the solution is low. Contrary, the

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 28,2023 at 16:49:58 UTC from IEEE Xplore. Restrictions apply.

genetic operator B has high diversity maintenance of the solution, but low optimization efficiency. Using the reward by each individual, the selection rate of the operator A becomes high. However, as a result of heavy usage of operator A, the diversity of the solutions decreases, so that there is a possibility that the reward lowers in the future. Reinforcement learning updates the state-action function by counting on estimated future rewards. When the state space that expresses the diversity of the solution is divided properly, the selection rate of another operator becomes high before the diversity becomes worse. To the contrary, when the state space is divided coarsely, there is high possibility that the advantage of the multipoint search cannot be used properly.

On the Contrary, when calculating the reward by population fitness improvement, because the reward by effect that diversity was maintained by operator B is accounted, not only operator A but also the selection probability of operator B increases. Therefore, the proposed method has a possibility to obtain a more effective parameter control strategy that counts on characteristics of multipoint searching.

As a problem, the genetic operator with actually low contribution level also has the possibility that the reward will be allotted. This will be settled while multiple-time learning is done under the same conditions. However, there is a possibility to require a lot of training time because it is not possible to evaluate it directly. To reduce this influence as much as possible, the proposed method implements the adjustment of the trade-off by using the weight sum function with not only the reward in the population search but also an immediate reward at each individual.

IV. CONCLUSION

A learning algorithm that controls parameter setting was proposed to improve the efficiency of a search method using an evolutionary algorithm such as a genetic algorithm. Compared with the on-policy type reinforcement learning method, this tries to further improve the efficiency by means of using operator calculating cost and multiple point search characteristic of GA. In this method, parameters are set through not only evaluating the result of parameter setting but also considering the search overhead such as the calculation time to apply one operator and multiple point search characteristic of GA.

This method is expected to improve the learning efficiency in comparison with the above method that does not consider the calculation cost of the search operator as well as the multiple point search characteristic.

In the future, learning method for parameter control in an evolutionary algorithm will be researched, especially focusing on the improvement method for giving the reward in reinforcement learning.

ACKNOWLEDGMENT

This work was partially supported by Research Institute for Science and Technology of Tokyo Denki University (Q10J-04).

REFERENCES

[1] L. Davis, editor,: "Handbook of Genetic Algorithms", Van Nostrand Reinhold, N.Y., (1991).

[2] D. Applegate, R. Bixby, V. Chvatal, and W. Cook, The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics): Princeton University Press, 2007.

[3] Y. Sakurai, T. Onoyama, S. Kubota, S. Tsuruta: "A Multi-inner-world Genetic Algorithm using Multiple Heuristics to Optimize Delivery Schedule", Proc. of the 2009 IEEE International Conference on Systems, Man and Cybernetics (SMC 2009), Texas, USA, pp611-616, (2009.10).

[4] Y. Sakurai, T. Onoyama, S. Kubota, S. Tsuruta: "A Multi-inner-world Genetic Algorithm to Optimize Delivery Problem with Interactive-time", Proc. of 4th IEEE Conference on Automation Science and Engineering (CASE 2008), Washington DC, USA, pp.583-590, (2008.8).

[5] F. G. Lobo, C. F. Lima, and Z. Michalewicz, Parameter Setting in Evolutionary Algorithms: Springer Publishing Company, Incorporated, 2007.

[6] Y. Sakurai, N. Honda, J. Nishino: "Acquisition of Knowledge for Gymnastic Bar Action by Active Learning Method", Journal of Advanced Computational Intelligence & Intelligent Informatics (JACIII), Vol.7, No.1, pp.10-18 (2003)

[7] A. Eiben, Z. Michalewicz, M. Schoenauer, and J. Smith, "Parameter Control in Evolutionary Algorithms," in F.G. Lobo, C.F. Lima, Z. Michalewicz (eds.), Parameter Setting in Evolutionary Algorithms , Springer-Verlag, Berlin Heidlberg, 2007, pp. 19-46.

[8] V. Nannen and A. E. Eiben, "A method for parameter calibration and relevance estimation in evolutionary algorithms," in Proceedings of the 8th annual conference on Genetic and evolutionary computation Seattle, Washington, USA: ACM, 2006.

[9] L. DaCosta, A. Fialho, M. Schoenauer, Mich, and l. Sebag, "Adaptive operator selection with dynamic multi-armed bandits," in Proceedings of the 10th annual conference on Genetic and evolutionary computation Atlanta, GA, USA: ACM, 2008.

[10] D. Thierens, "Adaptive Strategies for Operator Allocation," in F.G. Lobo, C.F. Lima, Z. Michalewicz (eds.), Parameter Setting in Evolutionary Algorithms , Springer-Verlag, Berlin Heidlberg, 2007, pp. 77-90.

[11] J. E. Pettinger and R. M. Everson, "Controlling Genetic Algorithms With Reinforcement Learning," in Proceedings of the Genetic and Evolutionary Computation Conference: Morgan Kaufmann Publishers Inc., 2002.

[12] Fei. Chen, Yang. Gao, Zhao-qian. Chen, and Shi-fu. Chen, "SCGA: Controlling Genetic Algorithms with Sarsa(0)," in Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on, 2005, pp. 1177-1183.

[13] Yamamoto, Y., and Kubo, M., “Invitation to Traveling Salesman Problem”, Asakura Syoten, Tokyo, 1997.

[14] Nagata, Y. and Kobayashi, S., “The Proposal and Evaluation of a Crossover for Traveling Salesman Problems: Edge Assembly Crossover”, Journal of Japan Society for AI, Vol.14, No.5, pp.848-859, 1999.

[15] R. Sutton and A. Barto, Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning): Mit Pr, 1998.

[16] Watkins C J C H and Dayan P., “Technical note: Qlearning,” Machine Learning, Vol. 8(3-4), pp. 279-292, 1992.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on September 28,2023 at 16:49:58 UTC from IEEE Xplore. Restrictions apply.