Parameter Learning Optimization for Intelligent Controllers Based on CARLA-PSO Combined Model: Postprint

Authors: Gu Xuejing, Zhang Mingru, Wang Zhiliang, Yucheng Guo

Date: 2018-05-18T00:00:00+00:00

Abstract

Improvements were made to the Continuous Action Reinforcement Learning Automaton (CARLA), and the improved CARLA was applied in combination with the Particle Swarm Optimization (PSO) algorithm to optimize PID parameters. Based on CARLA, a combined optimization learning model CARLA-PSO was established, which comprises two components: a CARLA learning loop and a PSO learning loop. Learning loop selection is performed via an optimization strategy selector, and optimal control is obtained through interaction with the environment. This paper conducted simulation experiments on mold level control in continuous casting. Experimental results demonstrate that CARLA-PSO exhibits high optimization efficiency and strong global search capability in PID parameter optimization, can achieve ideal control performance, and possesses promising application prospects.

Full Text

Preamble

Parameter Learning Optimization of Intelligent Controller Based on CARLA-PSO Composite Model

Gu Xuejing[1], Zhang Mingru[1], Wang Zhiliang[2], Guo Yucheng[3]
(1. School of Electrical Engineering, North China University of Science & Technology, Tangshan Hebei 063017, China;
2. School of Computer & Communication Engineering, University of Science & Technology Beijing, Beijing 100083, China;
3. College of Light Industry, North China University of Science & Technology, Tangshan Hebei 063000, China)

Abstract: This paper presents an improved Continuous Action Reinforcement Learning Automaton (CARLA) combined with Particle Swarm Optimization (PSO) for PID parameter optimization. Based on CARLA, a composite optimization learning model called CARLA-PSO is established, which comprises two components: a CARLA learning loop and a PSO learning loop. A strategy selector chooses between the learning loops to obtain optimal control through interaction with the environment. Simulation experiments on continuous casting mold level control demonstrate that CARLA-PSO exhibits high optimization efficiency and strong global search capability when optimizing PID parameters, achieving ideal control performance with promising application prospects.

Keywords: Continuous action reinforcement learning automata; particle swarm optimization; intelligent PID controller; mold level

0 Introduction

The mold is a critical component of continuous casting machines, and its level control precision significantly impacts productivity, special steel production, and the reduction of breakout accidents [1]. Due to the nonlinear, strongly coupled nature of mold level control systems and numerous disturbance factors [2], traditional PID control can only achieve satisfactory closed-loop control under specific process conditions. Intelligent control theory, which mimics partial intelligence of humans or animals, has become a primary approach for addressing complex and difficult-to-control systems.

To overcome the limitations of conventional PID control methods for mold level control, Tong Chaonan et al. [3] applied a constrained generalized predictive control method based on genetic algorithms, demonstrating superior performance compared to traditional PID control. To achieve self-tuning of PID controller parameters for mold level control, Cao Guangming et al. [4] proposed a fuzzy adaptive PID control strategy, while Fang Qichao et al. [5] introduced an improved PSO algorithm based on variable inertia weight and multi-population parallel optimization strategies. PSO is a population-based stochastic optimization technique widely used for multi-objective, multi-parameter learning optimization problems [6-8].

Reinforcement Learning (RL) is an online machine learning method that lies between supervised and unsupervised learning [9], and has been applied in intelligent control domains such as missile guidance control and aircraft intelligent evasion [10], robot automatic obstacle avoidance in unknown environments [11], and control of learning-type two-link manipulators [12]. Learning Automata (LA) constitute a primary tool for reinforcement learning [13], with CARLA being one variant characterized by its ability to output continuous actions and strong global search capability. MN Howell et al. [14] applied CARLA to PID parameter tuning for engine idle speed systems, effectively addressing vehicle emission exceedances and stalling issues despite insufficient system model infor-

mation. M Kashki et al. [15] applied CARLA to automatic voltage regulator PID parameter self-tuning for power synchronous generators, with simulation results proving higher control precision compared to PSO and genetic algorithms.

This paper designs a CARLA-PSO composite optimization model based on CARLA. The optimization process of this composite model is divided into two stages, each corresponding to the CARLA learning loop and PSO learning loop respectively. The early stage employs CARLA for "coarse" optimization, rapidly converging the environmental fitness evaluation value to a stable state, while the later stage uses PSO for "fine" optimization to further improve the precision of the environmental fitness evaluation value. This model demonstrates excellent performance in online learning optimization of parameters for continuous casting mold level controllers.

---

1 CARLA-PSO Composite Model Design

1.1 CARLA Learning Loop Design

CARLA improves its strategy through trial-and-error, random action selection, and interaction with unknown or stochastic environments, making it suitable for online optimization of continuously varying parameters. Unlike conventional reinforcement learning, CARLA employs a continuous action space instead of a discrete one, obtaining convergence information through probability density functions, thus making it more appropriate for engineering applications with continuous characteristics.

Each learning variable corresponds to one CARLA, with each parameter's learning process operating independently. The only factor linking them is the dynamic environment, and they share a cost function. Within each automaton, every action corresponds to a probability density function $f_i(x)$ as the basis for action selection. Using reinforcement signals $z(k)$, the system generates better-performing action sets and increases their probability of reselection through the learning subsystem. The CARLA learning loop design consists of the following steps:

a) **Initialization of density functions.** The probability density function is initialized using a uniform distribution form, where the reciprocal of the difference between each parameter' s maximum and minimum values serves as the probability density function:

$$f(x) = \begin{cases} \frac{1}{x_{\max} - x_{\min}} & x \in [x_{\min}, x_{\max}] \\ 0 & \text{otherwise} \end{cases}$$

where $x = \{x_1, x_2, \ldots, x_n\}$ represents the decision actions and $f = \{f_1, f_2, \ldots, f_n\}$ corresponds to the density function for each action.

b) **Action selection.** A random number $r \in (0,1)$ is generated. The action set is input into the environment, and the cost value $J(k)$ is obtained through a predefined cost function:

$$\int_{x_{\min}}^{x_{\max}} f_i(x)\,dx = z(k)$$

c) **Behavior performance evaluation calculation.** After obtaining the cost value $J(k)$, the behavior performance is evaluated using:

$$\beta(k) = \begin{cases} \frac{J_{\text{med}} - J_{\min}}{J_{\max} - J_{\min}} & J_{\text{med}} \in [J_{\min}, J_{\max}] \\ 0 & \text{otherwise} \end{cases}$$

where the comparison between the current cost value $J(k)$ and the minimum value $J_{\min}$ in the historical cost value set yields the current behavior' s goodness weight. The behavior performance $\beta \in [0,1]$, where values closer to 1 indicate better behavior performance and values closer to 0 indicate poorer performance.

d) **Update of probability density function.** $H(x,r)$ is a Gaussian neighborhood function whose value can change the likelihood of actions:

$$f(x,k+1) = \begin{cases} \alpha\beta(k)H(x,r) + (1-\alpha\beta(k))f(x,k) & x \in [x_{\min}, x_{\max}] \\ 0 & \text{otherwise} \end{cases}$$

where $H(x,r) = \exp\left(-\frac{(x-r)^2}{2\sigma^2}\right)$, with λ and σ representing the height and width of the Gaussian distribution function respectively, determining the learning speed:

$$\lambda = h_g(x_{\max} - x_{\min}), \quad \sigma = w_g(x_{\max} - x_{\min})$$

where h_g is called the normalization factor, which keeps the density function $f(x,k)$ within bounds. $x_i(k)$ is the action value obtained from this learning iteration.

1.2 An Improved CARLA

CARLA's action set is a finite interval that employs a non-parametric probability model. The initial action probability follows a uniform distribution, and during learning, a symmetric Gaussian neighborhood function "propagates" rewards from well-performing actions to adjacent actions. However, due to the use of a non-parametric probability model, this method involves complex calculation, storage, and updating of probability distributions. Additionally, each learning iteration requires updating integral functions and solving integral equations,

resulting in high computational cost and limiting the algorithm' s practical implementation.

To overcome these limitations, this paper proposes several improvements to CARLA, including updating the behavior vector using equation (5):

$$x_i(k+1) = x_i(k) + \omega_i(k) \cdot \text{norm}(\mu, \sigma)$$

where $\text{norm}(\mu, \sigma)$ is a normal random number generator with mean μ and variance σ, and $\omega_i(k)$ is the learning rate factor. Figure 2 [Figure 2: see original paper] compares the algorithm execution speed before and after improvement. The figure clearly shows that the original CARLA experiences a sharp increase in execution time after a certain number of learning iterations, becoming nearly inoperable. In contrast, the improved CARLA exhibits a linear relationship between learning iterations and algorithm execution time, significantly reducing computation time and substantially improving real-time performance.

1.3 PSO Learning Loop Design

PSO was originally proposed by Kennedy and Eberhart [16], inspired by the simulation of simple social systems. PSO combines cognitive behavior and social cooperation observed in bird flocks. This method demonstrates excellent robustness in solving nonlinear, non-differentiable, multi-modal, and high-dimensional problems [17]. PSO mimics the flocking behavior of birds and fish that search for food cooperatively, with group members continuously adapting their search patterns by learning from their own experiences and those of other members. Similar to genetic algorithms, PSO is an iterative optimization algorithm that initializes the system with a set of random solutions and searches for optimal values through iteration.

The PSO learning loop design consists of the following steps:

a) **Initialization of parameters**, such as iteration count, population size, and particle ranges. The initial velocity $V_i^d(0) = \{V_{i1}, V_{i2}, \ldots, V_{in}\}$ is set, and the velocity and position update formulas are:

b) **Velocity and position updates:**

$$V_i^d(k+1) = V_i^d(k) + V_i^d(k+1) + V_i^d(k+1) + \delta$$

$$X_i^d(k+1) = X_i^d(k) + V_i^d(k+1)$$

where $i = 1, 2, \ldots, N$ is the particle index, $d = 1, 2, \ldots, n$ is the dimension index, V_i^d is the velocity of the d-th behavior in the i-th particle, X_i^d is the position, δ is the inertia factor ($\delta > 0$), and C_1 and C_2 are acceleration constants typically in $[0, 4]$. The inertia factor δ controls the balance between global and local search

capabilities—larger values enhance global search while smaller values improve local search. random$(0,1)$ represents a uniform random number in $[0,1]$, $p^i_{\text{best}}(k)$ is the current best position of the i-th particle, and $g_{\text{best}}(k)$ is the current best position of the entire swarm.

c) **Termination conditions.** Two termination conditions are available: reaching the maximum generation $G_{\max}$ or stopping when the environmental fitness value remains unchanged for $G_{\text{consecutive}}$ consecutive generations.

---

2 Mold Level Control Based on CARLA-PSO

2.1 Description of Mold Level Control

As shown in Figure 3 [Figure 3: see original paper], the continuous casting mold level control process can be summarized as follows: Molten steel flows from the ladle (1) through the sliding gate to the tundish (2). The control system adjusts the opening of the tundish nozzle by moving the stopper (3) up and down according to the casting speed. The mold level is controlled by regulating the molten steel flow from the tundish to the mold (4). The level sensor (5) transmits mold level changes to the controller (6), which drives the stopper lifting mechanism (7) to change the stopper position and control the molten steel flow from tundish to mold, thereby stabilizing the mold level.

Mold level control is characterized by nonlinearity, time delay, and numerous disturbance factors, making it difficult for conventional PID controllers to achieve satisfactory control performance. Consequently, incorporating intelligent control concepts and leveraging computer technology advantages to transform traditional PID control has become a research focus in continuous casting mold level control. This paper attempts to improve conventional PID controllers using the proposed composite model to achieve online learning optimization of mold level controller parameters.

2.2 Liquid Level Control Model and Parameters

This paper performed system identification on the level control object of a No. 2 continuous caster at a Tangshan steel plant, with main parameters shown in Table 1 . To obtain the system model using parameter identification, the hydraulic servo valve opening was used as the system input variable and the mold level as the output variable. Using the least squares method for system model identification [18], the obtained transfer function is:

$$G(s) = \frac{0.081}{0.11s^2 + 0.10s + 0.002}$$

2.3 Definition of Environmental Fitness Function

When designing an intelligent mold level controller, the first step is to define a controller performance evaluation function (also called the environmental fitness function). Typical time-domain output performance indicators include overshoot, rise time, settling time, and steady-state error of the system' s step response. Common controller performance evaluation criteria include: Integral Absolute Error (IAE, $J = \int |e|\, dt$), Integral Squared Error (ISE, $J = \int e^2\, dt$), Integral Time-weighted Absolute Error (ITAE, $J = \int t|e|\, dt$), and Integral Time-weighted Squared Error (ITSE, $J = \int te^2\, dt$). Each criterion has its own advantages and disadvantages. While minimizing IAE and ISE can effectively reduce overshoot, they have minimal impact on shortening settling time. ITAE and ITSE can overcome these limitations but involve more complex and time-consuming derivation of analytical formulas. Considering the practical requirements of mold level control, this paper adopts the ITSE evaluation criterion, with the CARLA-PSO environmental fitness function defined as:

$$J = \sum_{i=1}^{n} k \cdot (Y_i - y_i)^2$$

where Y_i is the actual response curve, y_i is the ideal response curve, n is the number of samples, and k is the number of CARLA-PSO iteration learning cycles.

2.4 Simulation Results and Analysis

Based on practical parameter tuning experience for mold level controllers, the action set intervals for K_p, K_i, and K_d were set to $[0, 2]$, $[0, 1]$, and $[0, 2]$ respectively. The CARLA learning rate factor was initialized to 0.1, particle swarm size $N = 300$, initial particle velocity $V_i^d(0) = \{0, 0, 0\}$, particle search velocity interval $[-0.5, 0.5]$, and inertia factor $\delta = 0.4$ or 0.9. Since the later stage is the "fine-tuning" phase, δ should be smaller, so $\delta = 0.4$ was used, with $C_1 = C_2 = 0.8$. The mold level control system input signal was a unit step signal with a sampling time of 50s.

Figure 4 [Figure 4: see original paper] shows the response curve of the original system without controller action (solid line), which exhibits obvious overshoot, steady-state error, and excessively long response time. The dashed line represents the ideal response curve. The figure presents three representative learning processes. After the 4th iteration, the fitness value decreased by nearly half. When the iteration count reached 100, the system response curve showed no obvious overshoot, the response time was significantly shortened, and the fitness value dropped to 10% of its initial value. At this point, the optimization strategy selector disconnected the CARLA learning loop and connected the PSO learning loop, transitioning the system to the later optimization stage. After 500 CARLA-PSO iterations, the system response approached the ideal state.

To verify the performance of the CARLA-PSO algorithm, Figure 5 [Figure 5: see original paper] compares the fitness convergence effects of CARLA and PSO algorithms acting alone. The CARLA-PSO algorithm combines CARLA' s strong global search capability with PSO' s fast convergence speed. Compared with CARLA and PSO algorithms individually, it demonstrates faster convergence speed and higher optimization precision.

---

3 Conclusion

This paper discussed the CARLA model and modified the traditional CARLA behavior update function, designing an improved CARLA that enhances execution speed and practical applicability. Building upon this, the CARLA-PSO composite optimization model was established by combining it with the PSO algorithm. This model can autonomously learn control rules with fast convergence and high precision. Finally, the model was applied to continuous casting mold level control. Simulation experiments demonstrate that after multiple interactions with the environment, the model can autonomously acquire satisfactory mold level control rules. Compared with CARLA alone, CARLA-PSO exhibits faster convergence speed and higher optimization efficiency. Future research will focus on applying this model to other control loops in continuous casting and implementing it in actual mold level control systems or other controlled objects.

References

[1] Li Zulin. Research on expert system control of mold level [J]. Computer Engineering and Applications, 2009, 45(10): 199-201.

[2] Passenbrunner T. Mold level control of a continuous casting plant by switching control strategies [J]. IFAC Proceedings Volumes, 2010, 43(14): 367-372.

[3] Tong Chaonan, Xiao Lei, Peng Kaixiang, et al. Constrained generalized predictive control of mold level based on genetic algorithm [J]. Control and Decision, 2009, 24(11): 1735-1739.

[4] Cao Guangming, Wu Di, Zhang Dianhua. Mold level control system for casting-rolling mill based on fuzzy adaptive PID [J]. Control and Decision, 2007, 22(4): 399-402.

[5] Fang Qichao, Xu Lin, Wang Jianhui, et al. Improved PSO and its application in mold level control [J]. Chinese Journal of Scientific Instrument, 2006, 27(11): 1399-1402.

[6] Delgarm N, Sajadi B, Kowsary F, et al. Multi-objective optimization of the building energy performance: a simulation-based approach by means of particle swarm optimization (PSO) [J]. Applied Energy, 2016, 170: 293-303.

[7] Xiong Weili, Xu Baoguo, Zhou Qiming. Research on PID parameter optimization method based on improved particle swarm algorithm [J]. Computer Engineering, 2005, 31(24): 41-43.

[8] Huang Weiyong, Xu Xiaoju, Pan Xiaobo, et al. Research on contraction-expansion coefficient control strategy of quantum-behaved particle swarm optimization algorithm [J]. Application Research of Computers, 2016, 33(9): 2592-2595.

[9] Liu Xiao. Associative reinforcement learning based on continuous action learning automata [J]. Journal of Shanxi University: Natural Science Edition, 2015, 38(3): 426-431.

[10] Zhou Rui, Chen Zongji. Application of reinforcement learning in missile guidance [J]. Control Theory & Applications, 2001, 18(5): 748-750.

[11] Zhong Yuping, Wang Lidan, Duan Shukai, et al. Intelligent control system based on neural network and reinforcement learning [J]. Journal of Southwest University: Natural Science Edition, 2013, 35(11): 172-179.

[12] Shah H, Gopal M. A reinforcement learning algorithm with evolving fuzzy neural networks [J]. IFAC Proceedings, 2014, 47(1): 1161-1165.

[13] Mirsaleh M R, Meybodi M R. A learning automata-based memetic algorithm [J]. Genetic Programming & Evolvable Machines, 2015, 16(4): 399-453.

[14] Howell M N, Gordon T J, Best M C. The application of continuous action reinforcement learning automata to adaptive PID tuning [C]// Learning Systems for Control. London: IEEE Xplore, 2000: 2//1-2//4.

[15] Kashki M, Abdel-Magid Y L, Abido M A. A reinforcement learning automata optimization approach for optimum tuning of PID controller in AVR system [C]// Proc of International Conference on Intelligent Computing: Advanced Intelligent Computing Theories and Applications-with Aspects of Artificial Intelligence. [S. l.]: Springer-Verlag, 2008: 684-693.

[16] Hasanien H M. Design optimization of PID controller in automatic voltage regulator system using taguchi combined genetic algorithm method [J]. IEEE Systems Journal, 2013, 7(4): 825-831.

[17] Song Mingzhi, Yang Le. WSN coverage optimization method based on improved adaptive PSO algorithm [J]. Application Research of Computers, 2013, 30(11): 3472-3475.

[18] Zhang Jiayan, Lang Jiahong, Wang Jinzhong. Sliding mode variable structure control technology for converter mouth micro-differential pressure [J]. Control Engineering, 2009, 16(4): 419-422.

Note: Figure translations are in progress. See original paper for figures.