Multi-Agent Reinforcement Learning Methods with Dynamic Parameters for Logistic Tasks

Eugene Fedorov[1], Olga Nechyporenko[1], Yaroslav Korpan[1] and Tetiana Neskorodieva[2]

[1]*Cherkasy State Technological University, Shevchenko blvd., 460, Cherkasy, 18006, Ukraine*
[2]*Uman National University of Horticulture, Uman, Instytutska str., 1, Uman, 20305, Ukraine*

Abstract

Part of Industry 4.0 is building computer systems by combining artificial intelligence with robotics. Such computer systems play an important role in the planning of cargo transportation in supply chain management. One of the approaches to building such computer systems is the use of multi-agent systems. The aim of the work is to create a methodology for constructing proactive agents based on reinforcement learning to solve the problem of optimal planning of cargo transportation. To solve the problem of insufficient efficiency of computer agents, the existing methods of statistical and machine learning were investigated. To date, the most efficient approaches to creating proactive agents are reinforcement learning approaches. The formalization of the functioning of proactive agents is performed. As a part of creating a model for the functioning of proactive agents based on reinforcement learning, a procedure for generating a quasi-optimal action plan is proposed that models the planning function of a proactive agent, which speeds up the decision-making process. Multi-agent reinforcement learning methods are proposed, which are close to random search at the initial iterations, and close to directed search at the final iterations. This is ensured by the use of dynamic parameters and allows the increase in the learning rate by approximately 10 times while maintaining the mean squared error of the method.

Keywords

supply chain management, multi-agent system, proactive agent, reinforcement learning, dynamic programming, Monte Carlo, Temporal-Difference Learning

1. Introduction

The fourth industrial revolution or Industry 4.0 has brought about rapid changes in technology, manufacturing and social processes in the 21st century due to increasing interconnection and intelligent automation. Part of this phase of industrial change is the integration of artificial intelligence with robotics, which blurs the boundaries between the physical, digital and biological worlds and is based on parallel and distributed computing [1].

Such computer systems play an important role in the planning of cargo transportation in supply chain management (CSM) [2] and audit [3-4]. One of the approaches to building such computer systems is the use of multi-agent systems.

Despite a large number of studies on the problem of improving the efficiency of supply chains and reducing logistics costs, some questions remain open. The complexity of supply chains is constantly increasing due to globalization and beyond. If earlier goods were purchased in centralized hypermarkets, now online trading is developing with its unique SCM stages.

The aim of the work is to create a methodology for constructing proactive agents based on reinforcement learning to solve the problem of optimal planning of cargo transportation. To achieve the goal, the following tasks were set and solved:

- formalization of the functioning of proactive agents;
- propose models for the functioning of proactive agents with a utility function based on reinforcement learning;
- propose a multi-agent reinforcement learning method with time difference and dynamic parameters;

CMIS-2024: Seventh International Workshop on Computer Modeling and Intelligent Systems, May 3, 2024, Zaporizhzhia, Ukraine
fedorovee75@ukr.net (E. Fedorov); olne@ukr.net (O. Nechyporenko); y.korpan@chdtu.edu.ua (Y. Korpan); tvnesk1@gmail.com (T. Neskorodieva)
0000-0003-3841-7373 (E. Fedorov); 0000-0002-3954-3796 (O. Nechyporenko); 0000-0002-1455-5977 (Y. Korpan); 0000-0003-2474-7697 (T. Neskorodieva)

- propose a multi-agent reinforcement learning method based on Monte Carlo and dynamic parameters;
- propose a multi-agent reinforcement learning method based on adaptive dynamic programming and dynamic parameters.

2. Formulation of the research problem

The problem of increasing the efficiency of optimal cargo transportation planning comes down to the problem of finding such a set of plans $\{\widehat{\pi}_\mu\}$, that delivers a minimum of the mean square error (the difference between the cost of the resulting plan and the cost of the optimal plan), $F=\frac{1}{P}\sum_{\mu=1}^{P}(f(\widehat{\pi}_\mu)-f(\pi_\mu))^2 \rightarrow \min_{\{\widehat{\pi}_\mu\}}$, where P – power of multiple plans, $\widehat{\pi}_\mu$.– μth received plan, π_μ .– μth optimal plan, $f(\cdot)$ – cost function of the plan (for example, the length of the route in the case of the traveling salesman problem).

3. Literature review

Currently, the main types of computer agents of multi-agent systems are reactive and proactive agents [5-6].

Typically, a simple reactive agent has a set of behaviours (production rules), a database (stores its current state), and a knowledge base (stores its behaviours). A simple reactive agent makes a decision based on production rules. Each production rule consists of an antecedent (one or more perceptions) and a consequent (action).

Advantages of simple reactive agents [5-6]:

1. Simplicity of software implementation.
2. Ease of organization of multi-agent interaction.
3. High decision-making speed.
4. High probability of making the right decision.

Disadvantages of simple reactive agents [5-6]:

1. Simple reactive agents require a lot of information about their current state to determine an acceptable action.
2. Simple reactive agents do not take into account information about the current state of other agents.
3. Weak adaptability of simple reactive agents.
4. The behaviour of simple reactive agents is not based on a formal mathematical apparatus.
5. In the case of a large base of production rules, it is difficult to create a simple reactive agent.
6. Lack of logical inference leads to low autonomy.

Typically, a reactive agent with an internal state has a database (stores its current state), a knowledge base (stores knowledge about the world changes both independently and dependent on the agent's actions) and an inference engine. A reactive agent with an internal state makes a decision through logical inference.

Advantages of reactive agents with an internal state [5-6]:

1. The behaviour of reactive agents with an internal state is based on a formal mathematical apparatus (first-order predicate logic).
2. The presence of a logical inference leads to high autonomy.

Disadvantages of reactive agents with an internal state [5-6]:

1. Insufficient decision-making speed.
2. The complexity of organizing multi-agent interaction.
3. The complexity of the implementation of the perception function that maps signals from receptors into formulas in the language of first-order predicate logic.
4. The complexity of the formal description in the language of the first-order predicate logic of the dynamic environment.

Typically, a proactive agent has a database (stores information about its internal state, as well as the selected goal), a knowledge base (stores knowledge about the world changes both independently and dependent on the actions of the agent) and an inference engine. The proactive agent makes a decision about choosing a goal from a set of possible goals and how to achieve it by forming an action plan based on logical inference. A proactive agent may also be based on a utility function.

The advantages and disadvantages of proactive agents and reactive agents with an internal state are practically the same [5-6].

Thus, the current problem is the low efficiency of the considered software agents.

At present, instead of expert systems with logical inference used in decision-making agents, reinforcement learning is actively used [7-8]. The main areas of single-agent reinforcement learning are:

- dynamic programming [9-10];
- adaptive dynamic programming [11-12];
- Monte Carlo [13-14];
- temporal-difference learning [15-16];
- policy-based methods [17-18];
- actor-critic methods [19-20].

Today, multi-agent methods are actively developed [21–22].

The advantages of reinforcement learning over inference are:

- no labeled data sets are required, this is especially relevant for large amounts of data [23-24];
- there is no imitation of a teacher, but a new solution can be proposed that people have not even thought about [25-26];
- the quality criterion / utility function is used [27-28].

Disadvantages of reinforcement learning based on dynamic programming [9-10]:

- a priori knowledge about the probabilities of transitions between states is required;
- action is not selected (for a fixed policy).

Disadvantages of reinforcement learning based on adaptive dynamic programming [11-12]:

- action is not selected (for a fixed policy);
- cannot directly optimize the policy;
- a large number of interactions between the agent and the environment;
- converges to the global optimum only in the case of a finite number of actions and states;
- susceptible to retraining.

Disadvantages of Monte Carlo based reinforcement learning [13-14]:

- action is not selected (for a fixed policy);
- cannot directly optimize the policy;
- a large number of long trajectories is required;
- updating the value of the cost function only after receiving the entire trajectory;
- does not always converge to the global optimum;
- susceptible to retraining.

Disadvantages of reinforcement learning based on temporal-difference learning [15-16]:

- the policy is fixed, so no action is selected (if TD-learning);
- cannot directly optimize the policy;
- a large number of interactions between the agent and the environment;
- converges to the global optimum only in the case of a finite number of actions and states;
- susceptible to undertraining (if one-step TD learning).

Disadvantages of policy-based reinforcement learning [17-18]:

- requires a large number of long trajectories;
- does not always converge to the global optimum;
- subject to retraining.

Disadvantages of actor-critic reinforcement learning [19-20]:

- a large number of long trajectories (if MC learning) or a large number of interactions between the agent and the environment (if TD learning);
- does not always converge to the global optimum (if MC learning) or converges to the global optimum only in the case of a finite number of actions and states (if TD learning);
- subject to retraining (if MC training);
- susceptible to undertraining (if one-step TD training).

4. Formalization of models of proactive agents functioning

For such agents, the internal state is called belief, the possible goal is called desire, the best goal is called intention.

Formalization of the functioning of a proactive agent.

Perception function (1)

$$see: E \to Per \qquad (1)$$

maps the current state of the environment E into a new perception *Per*.

The state change function $next$ is called the belief change function *brf* (2)

$$brf: Bel \times Per \to Bel \qquad (2)$$

and maps belief (internal state) *Bel* (belief) and perception *Per* into belief (internal state) *Bel*.

Changing the intention (the best goal) is the sequential execution of the function for selecting the set of wishes (possible goals) *options* and the filtering function *filter*, which ensures the choice of the intention (the best goal) from the set of desires (possible goals).

Function to generate possible variants *options* (3)

$$options: Bel \times Int \to \mathsf{Des} \qquad (3)$$

maps belief (internal state) *Bel* and intention (best goal) *Int* into a set of desires (possible goals) **Des.**

Filter function *filter* (4)

$$filter: Bel \times \mathsf{Des} \times Int \to Int \qquad (4)$$

maps belief (internal state) *Bel*, a subset of desires (possible goals) **Des** and intention (best goal) *Int* to intention (best goal) *Int*.

Plan π is a sequence of actions

$$\pi = \{\alpha_1, \ldots, \alpha_n\},$$

where each α_i is an element of the set Ac .

$Plan = \{\pi_0, \pi_1, \ldots\}$ – set of all plans.

Instead of an action selection function $action$ a new planning function plan is used (5)

$$plan: Bel \times Int \times \mathsf{Ac} \to Plan \qquad (5)$$

which maps a belief (internal state) *Bel*, an intention (best goal) *Int*, and a subset of actions **Ac** into *Plan*.

5. Modelling the functioning of proactive agents with a utility function through reinforcement learning

Let a utility function u assign a utility to a state and be represented as (6)

$$u(\mathbf{s}(n)) = \max_{\mathbf{a} \in A(\mathbf{s}(n))} Q(\mathbf{s}(n), \mathbf{a}), \qquad (6)$$

where $Q(\mathbf{s}(n), \mathbf{a})$ – state-action cost function (profit in case of state $\mathbf{s}(n)$ and action **a**),
$A(\mathbf{s}(n))$ – set of actions available in state $\mathbf{s}(n)$.

Let there be a memory of reproducing experiments (7)

$$M = \{(\mathbf{s}, \mathbf{a}, R(\mathbf{s}, \mathbf{a}, \mathbf{s}'), \mathbf{s}')\}, \qquad (7)$$

where $R(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ – reward (reward for the transition from state **s** to state $\mathbf{s}'$ as a result of action **a**).

Then, for a proactive agent with a utility function, the procedure for generating an action plan π for the transition from the internal state (belief) $\mathbf{s}_0$ to the target state (intention) $\mathbf{s}^*$ models the planning function plan and is presented in the following form.

1. Initialization

$\mathbf{s}(0) = \mathbf{s}_0$, iteration number $n = 1$.

2. Choice of action (8) and observation of the internal state (9)

$$\mathbf{y}(n) = \arg\max_{\mathbf{a} \in A(\mathbf{s}(n-1))} Q(\mathbf{s}(n-1), \mathbf{a}), \qquad (8)$$

$$\exists (\mathbf{s}, \mathbf{a}, R(\mathbf{s}, \mathbf{a}, \mathbf{s}'), \mathbf{s}') \in M : \mathbf{s}(n-1) = \mathbf{s} \wedge \mathbf{y}(n) = \mathbf{a} \to \mathbf{s}(n) = \mathbf{s}'. \qquad (9)$$

3. Termination condition

If $n < N$, then $n = n + 1$, go to step 2, otherwise $\pi = (\mathbf{y}(1), \ldots, \mathbf{y}(n))$.

The paper proposes reinforcement learning methods based on temporal-difference learning, based on Monte Carlo and based adaptive dynamic programming.

6. Multi-agent reinforcement learning with temporal-difference and dynamic parameters

The method consists of the following steps.

1. Initialization.

1.1. The maximum number of iterations N, the number of agents K, the maximum length of the states' sequence T, the discrete set of states S, the discrete set of actions $A(s)$, $s \in S$, the reward $R(s,a)$, $a \in A(s)$, $s \in S$, the parameters $\rho_1^{\min}, \rho_1^{\max}, \rho_2^{\min}, \rho_2^{\max}$ (control the learning rate), $0 < \rho_1^{\min} < \rho_1^{\max} < 1$, $0 < \rho_2^{\min} < \rho_2^{\max} < 1$, the parameters $\varepsilon^{\min}, \varepsilon^{\max}$ (control the ε-greedy policy), $0 < \varepsilon^{\min} < \varepsilon^{\max} < 1$, the parameters $\gamma^{\min}, \gamma^{\max}$ (control discounting), $0 < \gamma^{\min} < \gamma^{\max} < 1$ are set.

1.2. Reward tables are initialized for the kth agent

$$Q_k = [Q_k(s,a)],$$

$$Q_k(s,a) = 0,\ a \in A(s),\ s \in S,\ k \in \overline{1,K}.$$

1.3. The reward table is initialized for a swarm of agents

$$Q_{swarm} = [Q_{swarm}(s,a)],$$

$$Q_{swarm}(s,a) = 0,\ a \in A(s),\ s \in S.$$

2. Iteration number n=1.

3. The parameters are calculated (10)-(13)

$$\rho_1(n) = \rho_1^{\max} - (\rho_1^{\max} - \rho_1^{\min})\frac{n-1}{N-1}, \tag{10}$$

$$\rho_2(n) = \rho_2^{\max} - (\rho_2^{\max} - \rho_2^{\min})\frac{n-1}{N-1}, \tag{11}$$

$$\varepsilon(n) = \varepsilon^{\max} - (\varepsilon^{\max} - \varepsilon^{\min})\frac{n-1}{N-1}, \tag{12}$$

$$\gamma(n) = \gamma^{\min} + (\gamma^{\max} - \gamma^{\min})\frac{n-1}{N-1}. \tag{13}$$

4. The number of the moment in time is set $t=1$.

5. The initial state s_{kt}, $k \in \overline{1,K}$ is observed for each kth agent.

6. For each kth agent, action a_{kt} is chosen using the ε-greedy policy π. If $U(0,1) < \varepsilon(n)$, then choose action a_{kt} randomly from the set of allowed actions $A(s_{kt})$, otherwise choose action a_{kt} in the form (14)

$$a_{kt} = \underset{b \in A(s_{kt})}{\arg\max}\, Q(s_{kt}, b),\ b \in A(s_{kt})), \tag{14}$$

i.e.

$$a_{kt} = \pi(s_{kt}),\ k \in \overline{1,K}.$$

7. For each kth agent, a reward $R(s_{kt}, a_{kt})$, $k \in \overline{1,K}$ is calculated.

8. For each kth agent a new state $s'_{kt} = a_{kt}$, $k \in \overline{1,K}$ is observed.

9. For each kth agent, the value of the combinations of the state-action cost functions of the swarm and the kth agent is calculated, i.e.

$$Q_{swarm}(s_{kt}, a_{kt}) \text{ and } \tilde{Q}_k(s_{kt}, a_{kt}),$$

in the form (15)

$$\tilde{Q}_k(s_{kt}, a_{kt}) = (1 - \rho_2(n))Q_{swarm}(s_{kt}, a_{kt}) + \rho_2(n) Q_k(s_{kt}, a_{kt}),\ k \in \overline{1,K}. \tag{15}$$

10. For each kth agent, the value of the cost function of the state-action $Q_k(s_{kt},a_{kt})$ is calculated as (16)

$$Q_k(s_{kt},a_{kt}) = \begin{cases} (1-\rho_1(n))\tilde{Q}_k(s_{kt},a_{kt}) + \\ +\rho_1(n)\left(R(s_{kt},a_{kt}) + \gamma(n)\max\limits_{b\in A(s'_{kt})}\tilde{Q}_k(s'_{kt},b)\right), & t<T \\ (1-\rho_1(n))\tilde{Q}_k(s_{kt},a_{kt}) + \rho_1(n)R(s_{kt},a_{kt}), & t=T \end{cases}, \quad k\in\overline{1,K}. \tag{16}$$

11. Calculate the value of the cost function of the state-action of the swarm of agents $Q_{swarm}(s_{kt},a_{kt})$ for each kth agent in the form (17)

$$Q_{swarm}(s_{kt},a_{kt}) = \begin{cases} \max\limits_{z\in\overline{1,K}} Q_z(s_{kt},a_{kt}), & \left|\max\limits_{z\in\overline{1,K}} Q_z(s_{kt},a_{kt})\right| > \left|\min\limits_{z\in\overline{1,K}} Q_z(s_{kt},a_{kt})\right| \\ \min\limits_{z\in\overline{1,K}} Q_z(s_{kt},a_{kt}), & \left|\max\limits_{z\in\overline{1,K}} Q_z(s_{kt},a_{kt})\right| \le \left|\min\limits_{z\in\overline{1,K}} Q_z(s_{kt},a_{kt})\right| \end{cases}, \quad k\in\overline{1,K}. \tag{17}$$

12. For each kth agent, the current state $s_{kt} = s'_{kt}$, $k\in\overline{1,K}$ is set.

13. If the current time is not the last, i.e. $t<T$, then increase the iteration number, i.e. $t=t+1$, go to step 6.

14. If the current iteration is not the last one, i.e. $n<N$, then increase the iteration number, i.e. $n=n+1$, go to step 3.

Note. Upon completion of the method, plan $\pi_k = (y_{k1},\ldots,y_{kt},\ldots,y_{kT})$ is formed for each kth agent (18)

$$y_{kt} = \operatorname*{argmax}_{a\in A(s_{kt})} Q_k(s_{kt},a), \quad s\in S, \ k\in\overline{1,K}. \tag{18}$$

The plan of the agent that satisfies the quality criterion better than others is selected.

7. Multi-agent Monte Carlo reinforcement learning with dynamic parameters

This method is presented in the following form.

1. Initialization.

1.1. The maximum number of iterations N, the number of agents K, the discrete set of states S, the discrete set of actions $A(s)$, $s\in S$, the reward $R(s,a)$, $a\in A(s)$, $s\in S$, the parameters $\rho_1^{\min},\rho_1^{\max},\rho_2^{\min},\rho_2^{\max}$ (control the learning rate), $0<\rho_1^{\min}<\rho_1^{\max}<1$, $0<\rho_2^{\min}<\rho_2^{\max}<1$, the parameters $\varepsilon^{\min},\varepsilon^{\max}$ (control the ε-greedy policy), $0<\varepsilon^{\min}<\varepsilon^{\max}<1$, the parameters $\gamma^{\min},\gamma^{\max}$ (control discounting), $0<\gamma^{\min}<\gamma^{\max}<1$ are set.

1.2. Reward tables are initialized for the kth agent

$$Q_k = [Q_k(s,a)],$$

$$Q_k(s,a)=0, \ a\in A(s), \ s\in S, \ k\in\overline{1,K}.$$

1.3. The reward table is initialized for a swarm of agents

$$Q_{swarm} = [Q_{swarm}(s,a)],$$

$$Q_{swarm}(s,a)=0, \ a\in A(s), \ s\in S.$$

1.4. Tables of the number of transitions for the kth agent are initialized

$$D_k = [D_k(s,a)],$$

$$D_k(s,a)=0, \ a\in A(s), \ s\in S, \ k\in\overline{1,K}.$$

2. Iteration number $n=1$.

3. The parameters are calculated (19)-(22)

$$\rho_1(n) = \rho_1^{\max} - (\rho_1^{\max}-\rho_1^{\min})\frac{n-1}{N-1}, \tag{19}$$

$$\rho_2(n) = \rho_2^{\max} - (\rho_2^{\max} - \rho_2^{\min})\frac{n-1}{N-1}, \tag{20}$$

$$\varepsilon(n) = \varepsilon^{\max} - (\varepsilon^{\max} - \varepsilon^{\min})\frac{n-1}{N-1}, \tag{21}$$

$$\gamma(n) = \gamma^{\min} + (\gamma^{\max} - \gamma^{\min})\frac{n-1}{N-1}. \tag{22}$$

4. Trajectory $\tau_k = (s_{k0}, a_{k0}, r_{k0}, \ldots, s_{kT}, a_{kT}, r_{kT})$ is generated for each kth agent, and $a_{kt} = \pi(s_{kt})$, $r_{kt} = R(s_{kt}, a_{kt})$, as a result of action a_{kt} a new state $s_{k,t+1}$ and reward r_{kt}, are observed, state s_{k0} can change at each iteration, the policy of choosing action *π* is ε-greedy, $k \in \overline{1,K}$.

5. Number of the moment in time $t = T$.

6. Calculate for each kth agent the profit in the form of a discounted amount of reward from time t to time *T* (23)

$$R_{kt}(\tau_k) = \sum_{t'=t}^{T} \gamma^{t'-t}(n) r_{kt'}, \quad k \in \overline{1,K}. \tag{23}$$

7. For each kth agent, the value of the combinations of the state-action cost functions of the swarm and the kth agent is calculated, i.e.

$$Q_{swarm}(s_{kt}, a_{kt}) \text{ and } \tilde{Q}_k(s_{kt}, a_{kt}),$$

in the form (24)

$$\tilde{Q}_k(s_{kt}, a_{kt}) = (1 - \rho_2(n))Q_{swarm}(s_{kt}, a_{kt}) + \rho_2(n)Q_k(s_{kt}, a_{kt}), \quad k \in \overline{1,K}. \tag{24}$$

8. For each kth agent, the transition counter $D_k(s_{kt}, a_{kt})$ is increased, i.e.

$$D_k(s_{kt}, a_{kt}) = D_k(s_{kt}, a_{kt}) + 1, \quad k \in \overline{1,K}.$$

9. For each kth agent, the value of the cost function of the state-action $Q_k(s_{kt}, a_{kt})$ is calculated as (25)

$$Q_k(s_{kt}, a_{kt}) = \left(1 - \frac{\rho_1(n)}{D_k(s_{kt}, a_{kt})}\right)\tilde{Q}_k(s_{kt}, a_{kt}) + \frac{\rho_1(n)}{D_k(s_{kt}, a_{kt})} R_{kt}(\tau_k), \quad k \in \overline{1,K}. \tag{25}$$

10. Calculate the value of the cost function of the state-action of the swarm of agents $Q_{swarm}(s_{kt}, a_{kt})$ for each kth agent in the form (26)

$$Q_{swarm}(s_{kt}, a_{kt}) = \begin{cases} \max\limits_{z \in \overline{1,K}} Q_z(s_{kt}, a_{kt}), & \left|\max\limits_{z \in \overline{1,K}} Q_z(s_{kt}, a_{kt})\right| > \left|\min\limits_{z \in \overline{1,K}} Q_z(s_{kt}, a_{kt})\right| \\ \min\limits_{z \in \overline{1,K}} Q_z(s_{kt}, a_{kt}), & \left|\max\limits_{z \in \overline{1,K}} Q_z(s_{kt}, a_{kt})\right| \le \left|\min\limits_{z \in \overline{1,K}} Q_z(s_{kt}, a_{kt})\right| \end{cases}, \quad k \in \overline{1,K}. \tag{26}$$

11. If $t > 0$, then $t = t - 1$, go to step 6.

12. If the current iteration is not the last one, i.e. $n < N$, then increase the iteration number, i.e. $n = n + 1$, go to step 3, otherwise stop.

Note. Upon completion of the method, plan $\pi_k = (y_{k1}, \ldots, y_{kt}, \ldots, y_{kT})$ is formed for each kth agent (27)

$$y_{kt} = \underset{a \in A(s_{kt})}{\arg\max}\, Q_k(s_{kt}, a), \quad s \in S, \quad k \in \overline{1,K}. \tag{27}$$

The plan of the agent that satisfies the quality criterion better than others is selected.

8. Multi-agent reinforcement learning method based on adaptive dynamic programming and dynamic parameters

The method consists of the following steps.

1. Initialization.

1.1. The maximum number of iterations N, the number of agents K, the maximum length of the states' sequence *T*, the discrete set of states S, the discrete set of actions $A(s)$, $s \in S$, the parameters $\rho_1^{\min}, \rho_1^{\max}, \rho_2^{\min}, \rho_2^{\max}$ (control the learning rate), $0 < \rho_1^{\min} < \rho_1^{\max} < 1$, $0 < \rho_2^{\min} < \rho_2^{\max} < 1$, the parameters $\varepsilon^{\min}, \varepsilon^{\max}$ (control the ε-greedy policy), $0 < \varepsilon^{\min} < \varepsilon^{\max} < 1$, the parameters $\gamma^{\min}, \gamma^{\max}$ (control discounting), $0 < \gamma^{\min} < \gamma^{\max} < 1$ are set.

1.2. Reward tables are initialized for the *k*th agent

$$Q_k(s,a),\ Q_k(s,a) = 0,\ a \in A(s),\ s \in S,\ k \in \overline{1,K}.$$

1.3. The reward table is initialized for a swarm of agents

$$Q_{swarm}(s,a),\ Q_{swarm}(s,a) = 0,\ a \in A(s),\ s \in S.$$

1.4. The tables of the number of transitions for the *k*th agent are initialized

$$D_k(s,a),\ D_k(s,a) = 0,\ a \in A(s),\ s \in S,\ k \in \overline{1,K}.$$

1.5. The state observation quantity tables for the *k*th agent are initialized

$$D_k(s),\ D_k(s) = 0,\ s \in S,\ k \in \overline{1,K}.$$

2. Iteration number *n*=1.

3. The parameters are calculated (28)-(31)

$$\rho_1(n) = \rho_1^{\max} - (\rho_1^{\max} - \rho_1^{\min})\frac{n-1}{N-1}, \tag{28}$$

$$\rho_2(n) = \rho_2^{\max} - (\rho_2^{\max} - \rho_2^{\min})\frac{n-1}{N-1}, \tag{29}$$

$$\varepsilon(n) = \varepsilon^{\max} - (\varepsilon^{\max} - \varepsilon^{\min})\frac{n-1}{N-1}, \tag{30}$$

$$\gamma(n) = \gamma^{\min} + (\gamma^{\max} - \gamma^{\min})\frac{n-1}{N-1}. \tag{31}$$

4. The number of the moment in time is set $t=1$.

5. The initial state s_{kt}, $k \in \overline{1,K}$ is observed for each *k*th agent.

6. For each *k*th agent, action a_{kt} is chosen using the ε-greedy policy *π*. If $U(0,1) < \varepsilon(n)$, then choose action a_{kt} randomly from the set of allowed actions $A(s_{kt})$, otherwise choose action a_{kt} in the form (32)

$$a_{kt} = \underset{b \in A(s_{kt})}{\arg\max}\, Q(s_{kt}, b),\ b \in A(s_{kt})), \tag{32}$$

i.e.

$$a_{kt} = \pi(s_{kt}),\ k \in \overline{1,K}.$$

7. For each *k*th agent, a reward $R(s_{kt}, a_{kt})$, $k \in \overline{1,K}$ is calculated.

8. For each *k*th agent a new state $s'_{kt} = a_{kt}$, $k \in \overline{1,K}$ is observed.

9. The transition function is calculated (33)

$$P_k(s'_{kt} \mid s_{kt}, a_{kt}) = \frac{D_k(s_{kt}, a_{kt})}{D_k(s_{kt})}, \tag{33}$$

and in advance

$$D_k(s_{kt}) = D_k(s_{kt}) + 1,\ D_k(s_{kt}, a_{kt}) = D_k(s_{kt}, a_{kt}) + 1.$$

10. For each *k*th agent, the value of the combinations of the state-action cost functions of the swarm and the *k*th agent is calculated, i.e.

$$Q_{swarm}(s_{kt}, a_{kt}) \text{ and } \tilde{Q}_k(s_{kt}, a_{kt}),$$

in the form (34)

$$\tilde{Q}_k(s_{kt}, a_{kt}) = (1 - \rho_2(n))Q_{swarm}(s_{kt}, a_{kt}) + \rho_2(n)Q_k(s_{kt}, a_{kt}),\ k \in \overline{1,K}. \tag{34}$$

11. For each kth agent, the value of the cost function of the state-action $Q_k(s,a)$ is calculated as (35)

$$Q_k(s_{kt},a_{kt}) = \begin{cases} (1-\rho_1(n))\tilde{Q}_k(s_{kt},a_{kt}) + \\ +\rho_1(n) P_k(s'_{kt}|s_{kt},a_{kt})\left(R(s_{kt},a_{kt}) + \gamma(n) \max\limits_{b\in A(s'_{kt})} \tilde{Q}_k(s'_{kt},b)\right), & t<T \\ (1-\rho_1(n))\tilde{Q}_k(s_{kt},a_{kt}) + \rho_1(n) P_k(s'_{kt}|s_{kt},a_{kt}) R(s_{kt},a_{kt}), & t=T \end{cases}, \quad (35)$$

$$k \in \overline{1,K}$$

12. Calculate the value of the cost function of the state-action of the swarm of agents $Q_{swarm}(s_{kt},a_{kt})$ for each kth agent in the form (36)

$$Q_{swarm}(s_{kt},a_{kt}) = \begin{cases} \max\limits_{z\in\overline{1,K}} Q_z(s_{kt},a_{kt}), & \left|\max\limits_{z\in\overline{1,K}} Q_z(s_{kt},a_{kt})\right| > \left|\min\limits_{z\in\overline{1,K}} Q_z(s_{kt},a_{kt})\right| \\ \min\limits_{z\in\overline{1,K}} Q_z(s_{kt},a_{kt}), & \left|\max\limits_{z\in\overline{1,K}} Q_z(s_{kt},a_{kt})\right| \le \left|\min\limits_{z\in\overline{1,K}} Q_z(s_{kt},a_{kt})\right| \end{cases}, \quad k\in\overline{1,K}. \quad (36)$$

13. For each kth agent, the current state $s_{kt} = s'_{kt}$, $k\in\overline{1,K}$ is set.

14. If the current time is not the last, i.e. $t<T$, then increase the iteration number, i.e. $t=t+1$, go to step 6.

15. If the current iteration is not the last one, i.e. $n<N$, then increase the iteration number, i.e. $n=n+1$, go to step 3.

Note. Upon completion of the method, plan $\pi_k = (y_{k1},\ldots,y_{kt},\ldots,y_{kT})$ is formed for each kth agent (37)

$$y_{kt} = \arg\max_{a\in A(s_{kt})} Q_k(s_{kt},a),\ s\in S,\ k\in\overline{1,K}. \quad (37)$$

The plan of the agent that satisfies the quality criterion better than others is selected.

9. Experiments and results

The numerical study of the proposed methods was carried out using the Python package.

For multi-agent reinforcement learning methods, the value of parameters $\rho_1^{\min}=0.1, \rho_1^{\max}=0.9$, $\rho_2^{\min}=0.1, \rho_2^{\max}=0.9$ (control the learning rate), parameters $\varepsilon^{\min}=0.1, \varepsilon^{\max}=0.9$ (control the ε-greedy policy), parameters $\gamma^{\min}=0.1, \gamma^{\max}=0.9$ (control discounting), the number of agents is $K=20$.

The dependence of parameter $\gamma(n)$ is defined as

$$\gamma(n) = \gamma^{\min} + (\gamma^{\max} - \gamma^{\min})\frac{n-1}{N-1}.$$

The dependence of parameter $\gamma(n)$ on the iteration number n is linear and shows that its share increases with the iteration number.

The dependence of parameter $\rho_1(n), \rho_2(n)$ and $\varepsilon(n)$ is defined as

$$\rho_1(n) = \rho_1^{\max} - (\rho_1^{\max} - \rho_1^{\min})\frac{n-1}{N-1},$$

$$\rho_2(n) = \rho_2^{\max} - (\rho_2^{\max} - \rho_2^{\min})\frac{n-1}{N-1},$$

$$\varepsilon(n) = \varepsilon^{\max} - (\varepsilon^{\max} - \varepsilon^{\min})\frac{n-1}{N-1}.$$

The dependence of parameter $\rho_1(n), \rho_2(n)$ and $\varepsilon(n)$ on the iteration number n is linear; it shows that their share decreases with increasing iteration number.

The results of comparing the proposed temporal-difference reinforcement learning method with dynamic parameters and the traditional Q-learning method based on the mean squared error criterion and the number of iterations for solving the travelling salesman problem (Berlin52 standard dataset), which is used for planning cargo transportation are presented in Table 1.

Table 1
Comparison of the proposed optimization method with the traditional Q-learning method

Mean squared error of the method		Number of iterations	
proposed	existing	proposed	existing
0.05	0.05	310	2030

The results of comparing the proposed Monte Carlo based reinforcement learning method with dynamic parameters and with the traditional every-visit method based on the mean squared error criterion and the number of iterations for solving the travelling salesman problem (Berlin52 standard dataset), which is used for planning cargo transportation are presented in Table 2.

Table 2
Comparison of the proposed optimization method with the traditional every-visit method

Mean squared error of the method		Number of iterations	
proposed	existing	proposed	existing
0.05	0.05	420	4050

The results of a comparison of the proposed reinforcement learning method based on adaptive dynamic programming with dynamic parameters and the traditional passive adaptive dynamic programming method based on the mean square error criterion and the number of iterations for solving the traveling salesman problem (Berlin52 standard dataset), which is used for freight planning, are presented in Table 3.

Table 3
Comparison of the proposed optimization method with the traditional passive adaptive dynamic programming method

Mean squared error of the method		Number of iterations	
proposed	existing	proposed	existing
0.05	0.05	110	1040

10. Discussion

Advantages of the proposed methods:

1. Modification of reinforcement learning methods due to dynamic parameters allows for an increase in the learning rate while maintaining the mean squared error of the method (Tables 1-3).
2. The use of a multi-agent approach makes distributed computing possible and increases the learning speed while maintaining the root-mean-square error of the method (Tables 1-3).
3. Reinforcement learning methods with dynamic parameters use the ε-greedy approach, which is close to random search at initial iterations, and close to directed search at final iterations. This is ensured by the use of dynamic parameters and allows for an increase in the learning rate while maintaining the mean squared error of the method (Tables 1-3).

11. Conclusions

To solve the problem of insufficient efficiency of computer agents, the existing methods of statistical and machine learning were investigated. These studies have shown that, to date, the most effective approaches to creating proactive agents are reinforcement learning approaches. The formalization of the functioning of proactive agents has been conducted.

As part of creating a model for the functioning of proactive agents based on reinforcement learning, a procedure for generating a quasi-optimal action plan is proposed that models the planning function of a proactive agent, which speeds up the decision-making process. Reinforcement learning methods are proposed, which at the initial iterations are close to random search, and at the final iterations are close to the directed search. This is ensured by the use of dynamic parameters and multi-agent approach and allows for an increase in the learning rate while maintaining the mean squared error of the method.
The proposed multi-agent methods will be used for freight planning in supply chain management and auditing, and were investigated on a standard data set.

12. References

[1] G. G. Shvachych, O. V. Ivaschenko, V. V. Busygin, Ye. Ye. Fedorov, Parallel computational algorithms in thermal processes in metallurgy and mining, Naukovyi Visnyk Natsionalnoho Hirnychoho Universytetu, 4 (2018) 129–137. doi: 10.29202/nvngu/2018-4/19.

[2] O. Grygor, E. Fedorov, O. Nechyporenko, M. Grygorian, Neural network forecasting method for inventory management in the supply chain, in: CEUR Workshop Proceedings, 2022, volume 3137, pp. 14-27.

[3] T. Neskorodieva, E. Fedorov, Method for automatic analysis of compliance of expenses data and the enterprise income by neural network model of forecast, in: CEUR Workshop Proceedings, 2020, volume 2631, pp. 145–158

[4] T. Neskorodieva, E. Fedorov, I. Izonin, Forecast method for audit data analysis by modified liquid state machine, in: CEUR Workshop Proceedings, 2020, volume 2623, pp. 25-35.

[5] G. Jezic, J. Chen-Burger, M. Kusek, R. Sperka, R. J. Howlett, L. C. Jain (Eds.), Agents and multi-agent systems: technologies and applications, volume 186 of Smart innovation, systems and technologies, 2020. doi: 10.1007/978-981-15-5764-4.

[6] S. Russell, P. Norvig, Artificial Intelligence: a Modem Approach, Englewood Cliffs, NJ: Prentice Hall PTR, 2020.

[7] A. L. C. Ottoni, E. G. Nepomuceno, M. S. de Oliveira, D. C. R. de Oliveira, Reinforcement learning for the traveling salesman problem with refueling, Complex & Intelligent Systems, 8 (2021) 2001-2015. doi: 10.1007/s40747-021-00444-4.

[8] A. Oroojlooy, D. Hajinezhad, A review of cooperative multi-agent deep reinforcement learning, Applied Intelligence, 53 (2023) 13677–13722. doi:10.1007/s10489-022-04105-y.

[9] D. Wang, X. Li, P. Xin, A. Liu, J. Qiao, Supplementary heuristic dynamic programming for wastewater treatment process control, Expert Systems with Applications, 247 (2024) 123280. doi: 10.1016/j.eswa.2024.123280.

[10] U. Satic, P. Jacko, C. Kirkbride, A simulation-based approximate dynamic programming approach to dynamic and stochastic resource-constrained multi-project scheduling problem, European Journal of Operational Research, 315 (2024) 454–469. doi: 10.1016/j.ejor.2023.10.046.

[11] H. Shen, Z. Li, J. Wang, J. Cao, Nonzero-sum games using actor-critic neural networks: A dynamic event-triggered adaptive dynamic programming, Information Sciences, 662 (2024) 120236. doi: 10.1016/j.ins.2024.120236.

[12] K. Xie, Y. Zheng, Y. Jiang, W. Lan, X. Yu, Optimal dynamic output feedback control of unknown linear continuous-time systems by adaptive dynamic programming, Automatica, 163 (2024) 111601. doi: 10.1016/j.automatica.2024.111601.

[13] J. Pascal, Artificial neural networks to solve dynamic programming problems: A bias-corrected Monte Carlo operator, Journal of Economic Dynamics and Control, 162 (2024) 104853. doi: 10.1016/j.jedc.2024.104853.

[14] S. V. Albrecht, F. Christianos, L. Schäfer, Multi-Agent Reinforcement Learning: Foundations and Modern Approaches, MIT Press, Cambridge, MA, USA, 2023.

[15] X. Chen, G. Yang, Sh. Yang, H. Wang, Sh. Dong, Ya. Gao, Online attentive kernel-based temporal difference learning, Knowledge-Based Systems, 278 (2023) 110902. doi:10.1016/j.knosys.2023.110902.

[16] M. S. Stanković, M. Beko, S. S. Stanković, Distributed consensus-based multi-agent temporal-difference learning, Automatica, 151 (2023) 110922. doi:10.1016/j.automatica.2023.110922.

[17] A. S. Stebenkov, N. O. Nikitin, Automated Generation of Ensemble Pipelines using Policy-Based Reinforcement Learning method, Procedia Computer Science, 229 (2023) 70-79. doi:10.1016/j.procs.2023.12.009.

[18] F. Huang, X. Deng, Y. He, W. Jiang, A novel policy based on action confidence limit to improve exploration efficiency in reinforcement learning, Information Sciences, 640 (2023) 119011. doi:10.1016/j.ins.2023.119011.

[19] J. Zhang, Sh. Han, X. Xiong, Sh. Zhu, Sh. Lu, Explorer-Actor-Critic: Better actors for deep reinforcement learning, Information Sciences, 662 (2024) 120255. doi:10.1016/j.ins.2024.120255.

[20] Zh. Zhang, X. Liang, C. Chen, D. Liu, Ch. Yu, W. Li, Defense penetration strategy for unmanned surface vehicle based on modified soft actor–critic, Ocean Engineering, 304 (2024) 117840. doi: 10.1016/j.oceaneng.2024.117840.

[21] T. Li, K. Zhu, N. C. Luong, D. Niyato, Q. Wu, Y. Zhang, B. Chen, Applications of multi-agent reinforcement learning in future Internet: A comprehensive survey, IEEE Communications Surveys & Tutorials, 24 (2) (2022) 1240–1279. doi:10.1109/COMST.2022.3160697.

[22] L. M. Schmidt, J. Brosig, A. Plinge, B. M. Eskofier, C. Mutschler, An introduction to multi-agent reinforcement learning and review of its application to autonomous mobility, in: IEEE 25th International Conference on Intelligent Transportation Systems, 2022, pp. 1342–1349.

[23] P. Yadav, A. Mishra, S. Kim, A comprehensive survey on multi-agent reinforcement learning for connected and automated vehicles, Sensors, 23 (10) (2023) 4710. doi:10.3390/s23104710.

[24] J. Orr, A. Dutta, Multi-agent deep reinforcement learning for multi-robot applications: A survey, Sensors, 23 (7) (2023) 3625. doi.org/10.3390/s23073625.

[25] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, S. Spanò, Multi-agent reinforcement learning: A review of challenges and applications, Applied Sciences, 11 (11) (2021) 4948. doi: 10.3390/app11114948.

[26] Z. Xu, H. van Hasselt, M. Hessel, J. Oh, S. Singh, D. Silver, Meta-gradient reinforcement learning with an objective discovered, arXiv:2007.08433, 2020. doi:10.48550/arXiv.2007.08433.

[27] H. Wang , E. Miahi, M. White, M. C. Machado, Z. Abbas, R. Kumaraswamy, V. Liu, A. White, Investigating the properties of neural network representations in reinforcement learning, Artificial Intelligence, 330 (2024) 1-24. doi: 10.1016/j.artint.2024.104100.

[28] F. Robertazzi, M. Vissani, G. Schillaci, E. Falotico, Brain-inspired meta-reinforcement learning cognitive control in conflictual inhibition decision-making task for artificial agents, Neural Networks, 154 (2022) 283–302. doi: 10.1016/j.neunet.2022.06.020.