https://doi.org/10.31891/csit-2024-1-14
UDC 519.857

Eugene FEDOROV, Olga NECHYPORENKO
Cherkasy State Technological University
Tetiana NESKORODIEVA
Uman National University of Horticulture
Maryna LESHCHENKO
Cherkasy State Technological University

# DYNAMIC PROGRAMMING FOR THE INVENTORY MANAGEMENT PROBLEM SOLUTION IN LOGISTICS

*Currently, there is a problem of methods insufficient efficiency for finding solutions to the inventory management problem. The research object is the process of solving inventory management problems. The research subject is methods for finding a solution to the inventory management problem based on dynamic programming. The research goal is to increase the efficiency of finding a solution to the inventory management problem through dynamic programming. A method based on deterministic dynamic programming, a method based on stochastic dynamic programming, a method based on Q-learning, and a method based on SARSA were applicated for the inventory management problem. There are advantages of the methods. of Methods modification of deterministic and stochastic dynamic programming, Q-learning, and SARSA due to dynamic parameters makes it possible to increase the learning speed while maintaining the root-mean-square error of the method. The numerical study made it possible to evaluate the methods (for modifying the deterministic and stochastic dynamic programming methods, the number of iterations is close to the number of stages; for both methods of deterministic and stochastic dynamic programming, the root mean square error was 0.02; for modifying the Q-learning and SARSA methods, the number of iterations was 300, for both methods of Q-learning and SARSA, the root mean square error was 0.05). These methods make it possible to expand the scope of dynamic programming, which is confirmed by their adaptation to the inventory management problem and helps to increase the intelligent computer systems efficiency for general and special purposes. The application of these methods for a wide class of artificial intelligence problems are the prospects for further research.*

*Keywords: dynamic programming, reinforcement learning, inventory management, Q-learning method, SARSA method, parallel information processing*

Євген ФЕДОРОВ, Ольга НЕЧИПОРЕНКО
Черкаський державний технологічний університет
Тетяна НЕСКОРОДЄВА
Уманський національний університет садівництва
Марина ЛЕЩЕНКО
Черкаський державний технологічний університет

# ДИНАМІЧНЕ ПРОГРАМУВАННЯ ДЛЯ РІШЕННЯ ЗАВДАННЯ УПРАВЛІННЯ ЗАПАСАМИ У ЛОГІСТИЦІ

*В даний час існує проблема недостатньої ефективності методів пошуку вирішення задач управління запасами. Об'єктом дослідження є вирішення завдань управління запасами. Предметом дослідження є методи пошуку вирішення завдання управління запасами на основі динамічного програмування. Метою роботи є підвищення ефективності пошуку вирішення задач управління запасами за рахунок динамічного програмування. Для досягнення поставленої мети в роботі були створені: метод на основі детермінованого динамічного програмування, метод на основі динамічного стохастичного програмування, метод на основі Q-навчання, метод на основі SARSA для завдання управління запасами. До переваг запропонованих методів належить наступне. Модифікація методів детермінованого та стохастичного динамічного програмування, Q-навчання та SARSA за рахунок динамічних параметрів дозволяє підвищити швидкість навчання при збереженні середньоквадратичної помилки методу. Проведене чисельне дослідження дозволило оцінити запропоновані методи (для модифікації методів детермінованого та стохастичного динамічного програмування кількість ітерацій близька до кількості етапів, для обох методів детермінованого та стохастичного динамічного програмування середньоквадратична помилка склала 0.02, для модифікації методів Q-навчання та SAR методів Q-навчання та SARSA середньоквадратична помилка склала 0.05). Запропоновані методи дозволяють розширити сферу застосування динамічного програмування, що підтверджується їх адаптацією для завдання управління запасами та сприяє підвищенню ефективності інтелектуальних комп'ютерних систем загального та спеціального призначення. Перспективами подальших досліджень є дослідження запропонованих методів для широкого класу задач штучного інтелекту.*

*Ключові слова: динамічне програмування, навчання з підкріпленням, управління запасами, метод Q-навчання, метод SARSA, паралельна обробка інформації*

## Introduction

Modern companies are optimizing their business processes based on lean manufacturing technology and the theory of constraints technology. The method's relevance for intellectualizing the “lean production” and “theory of constraints” technology which are based on the solution of optimization problems, for example, the problem of inventory management, increases significantly nowadays [1-2].

Optimization methods that find an exact solution have high computational complexity [3-5]. Optimization methods that find an approximate solution through a directed search have a high probability of hitting a local

extremum [6-8]. Random search methods don't guarantee convergence [9-12]. The problem of optimization methods' insufficient efficiency should be solved in this regard.

**Related works**

Today, many optimization problems are solved using reinforcement learning methods, based on dynamic programming.

Existing reinforcement learning methods have one or more of the following disadvantages:

– there is only an abstract method description or the method description is focused on solving only a specific problem [13-14];

– the method convergence isn't guaranteed [15-16];

– the iteration number influence on the finding a solution process isn't considered [17-18];

– there is no possibility of solving constrained optimization problems [19];

– insufficient accuracy of the method [20];

– the procedure for determining parameter values is not automated [21].

The task of constructing effective reinforcement learning methods is actually nowadays [22-24].

The research goal is to minimize costs of overproduction and inventory excess by creating effective optimization methods based on dynamic programming and parallel information processing technology.

**Research problem**

The efficiency-increasing problem of finding a solution to inventory management based on dynamic programming methods is presented as the problem of finding a solution $x^*$, where the goal function is $F(x^*) \to \min$ and $T \to \min$.

We proposed to use a combination of two functions as a goal function:

$$F(x,z) = F1(x,z) + F2(x,z) \to \min_x,$$

$$F1(x,z) = \sum_{m=1}^{M} w1 \cdot \max\{0, z^{\min} - (x_m + z_{m-1} - D_m)\},$$

$$F2(x,z) = \sum_{m=1}^{M} w2 \cdot \max\{0, x_m + z_{m-1} - D_m - z^{\max}\},$$

$$z_m = x_m + z_{m-1} - D_m,$$

where $F(\cdot)$ – a goal function, $F1(\cdot)$ – the costs of stockouts, $F2(\cdot)$ – storage costs of goods, $w1$ – the profit from the sale of one unit of goods (it is set), $w2$ – the cost of storing one unit of goods (it is set), $x_m$ – quantity of purchased goods from the supplier during the $m^{\text{th}}$ stage, $z_m$ – the amount of inventory of goods at the end of the $m^{\text{th}}$ stage, $z_0$ – the initial quantity of inventory of goods (it is set), $z^{\min}, z^{\max}$ – the minimum and maximum quantity of goods inventory at the end of each stage (it is set), $D_j$ – the quantity of goods sold during the $m^{\text{th}}$ stage (it is set), $M$ – number of stages.

$z^{\min}$ can be considered as the boundary between the black and red stock buffer zones. $z^{\max}$ can be considered as the boundary between the green and blue stock buffer zones. Getting into the inventory buffer black zone is described by the condition $F1(x,z) > 0$. Getting into the red/yellow/green inventory buffer zone is described by the condition $F1(x,z) + F2(x,z) = 0$. Getting into the blue inventory buffer zone is described by the condition $F2(x,z) > 0$.

There is a restriction on the quantity of purchased goods from the supplier, i.e.,

$$x^{\min} \le x_m \le x^{\max}, \; m \in \overline{1,M},$$

where $x^{\min}, x^{\max}$ – are the minimum and maximum quantity of purchased goods from the supplier during each stage.

**Method to solve the inventory management problem based on deterministic dynamic programming**

There was a proposed method to solve the problem of inventory management based on deterministic dynamic programming.

$$\varphi_1(z_1) = \min_{x_1} F_1(x_1, z_0),$$

$$\varphi_m(z_m) = \min_{x_m}\{F_m(x_m, z_{m-1}) + \varphi_{m-1}(z_{m-1})\},\ m \in \overline{2,M},$$

$$F_m(x_m, z_{m-1}) = F1_m(x_m, z_{m-1}) + F2_m(x_m, z_{m-1}),$$

$$F1_m(x_m, z_{m-1}) = w1 \cdot \max\{0, z^{\min} - (x_m + z_{m-1} - D_m)\},$$

$$F2_m(x_m, z_{m-1}) = w2 \cdot \max\{0, x_m + z_{m-1} - D_m - z^{\max}\},$$

$$z_m = x_m + z_{m-1} - D_m,$$

where $\varphi_m(z_m)$ – minimum costs for the amount of inventory goods $z_m$ at the end of the $m^{\text{th}}$ stage.

**Method to solve the inventory management problem based on stochastic dynamic programming**

According to this method, the inventory buffer can be in the black zone (state 1), red/yellow/green zone (state 2), and blue zone (state 3). The state diagram of the inventory buffer is shown in Fig. 1.

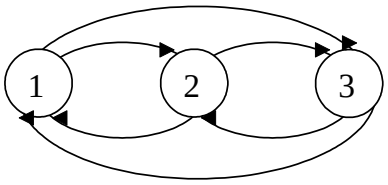


Fig. 1. The state diagram of the inventory buffer

$$\varphi_1(i) = \min_{x_1}\left\{\sum_{j=1}^{S} p_{ij}(x_1, z_0) r_{ij}(x_1, z_0)\right\},\ i \in \overline{1,S},$$

$$\varphi_m(i) = \min_{x_m}\left\{\sum_{j=1}^{S} p_{ij}(x_k, z_{m-1})(r_{ij}(x_k, z_{m-1}) + \varphi_{m-1}(j))\right\},\ i \in \overline{1,S},\ m \in \overline{2,M},$$

$$p_{ij}(x_m, z_{m-1}) = \begin{cases} & (F1_m(x_m, z_{m-1}) > 0 \wedge j = 1) \vee \\ 1, & (F1_m(x_m, z_{m-1}) + F2_m(x_m, z_{m-1}) = 0 \wedge j = 2) \vee \\ & (F2_m(x_m, z_{m-1}) > 0 \wedge j = 3) \\ 0, & otherwise \end{cases},\ i \in \overline{1,S},$$

$$r_{ij}(x_m, z_{m-1}) = \begin{cases} F1_m(x_m, z_{m-1}), & j = 1 \\ 0, & j = 2,\ i \in \overline{1,S}, \\ F2_m(x_m, z_{m-1}), & j = 3 \end{cases}$$

$$F1_m(x_m, z_{m-1}) = w1 \cdot \max\{0, z^{\min} - (x_m + z_{m-1} - D_m)\},$$

$$F2_m(x_m, z_{m-1}) = w2 \cdot \max\{0, x_m + z_{m-1} - D_m - z^{\max}\},$$

$$z_m = x_m + z_{m-1} - D_m,$$

where $\varphi_m(i)$ – is a minimum cost when the inventory buffer state transitions from state *i* at the end of the $m^{\text{th}}$ stage, $p_{ij}(x_m, z_{m-1})$ – is the probability of the inventory buffer transition from state *i* to state *j* in the case of the goods purchased amount from the supplier $x_m$ and the inventory goods amount $z_{m-1}$, $r_{ij}(x_m, z_{m-1})$ – are costs that arise when the inventory buffer moves from state *i* to state *j* in the case of the goods purchased amount from the supplier $x_m$ and the inventory goods amount $z_{m-1}$, *S* – is a number of inventory buffer states, *S*=3.

The stochastic dynamic programming advantage is the ability to describe the dynamic inventory buffer management model in the inventory buffers state visual diagram.

**Method to solve the inventory management problem based on Q-learning with dynamic parameters**

The method based on Q-learning with dynamic parameters consists of the following steps:

1. Initialization.

1.1. To set the maximum number of iterations *N*, the number of stages *M*, the profit from the sale of one goods unit $w1$, the cost of storing one goods unit $w2$, the minimum and maximum goods amounts purchased from the supplier $x^{\min}, x^{\max}$, the minimum and maximum amounts of goods inventories at the end of each stage

$z^{\min}, z^{\max}$, the initial amount of goods inventories $z_0$, the amount of goods sold are set $D_m$, $m \in \overline{1,M}$, parameters $\rho^{\min}, \rho^{\max}$ (to control the learning rate), $0 < \rho^{\min} < \rho^{\max} < 1$, parameters $\varepsilon^{\min}, \varepsilon^{\max}$ for the ε-greedy approach, $0 < \varepsilon^{\min} < \varepsilon^{\max} < 1$, parameters $\theta^{\min}, \theta^{\max}$ (to determine the importance of the future reward), $0 < \theta^{\min} < \theta^{\max} < 1$.

1.2. To initialize the reward table $Q = [Q(i, j)]$, $Q(i, j) = 0$, $i, j \in \overline{1,M}$.

2. Iteration number *n*=1.

3. To calculate the parameters:

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

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

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

4. To set the initial state (initial amount of goods inventory) $s = z_0$.

5. To set the stage starting number $m = 1$.

6. To select an *a* action (the amount of goods purchased from the supplier), it is necessary to move from state *s*, using the ε-greedy approach (if $U(0,1) < \varepsilon(n)$, then to select an action a randomly from the allowed actions set $\{x^{\min}, x^{\max}\}$, otherwise to select an *a* action: $a = \arg\max_b Q(s,b)$, $b \in \{x^{\min}, x^{\max}\}$.The *a* action becomes a new component of the vector of the amount of goods purchased from the supplier vector, i.e. $x_m = a$ .

7. If the last stage $m = M$, then go to step 13, otherwise $m = m + 1$.

8. To calculate the element of the current reward table $R(s,a)$:

$$R(s,a) = -(w1 \cdot \max\{0, z^{\min} - (a + s - D_m)\} + w2 \cdot \max\{0, a + s - D_m - z^{\max}\}).$$

9. There are a new state (the amount of the product inventory) $e = a + s - D_m$.

10. To calculate the element of the reward table $Q(s,a)$:

$$Q(s,a) = (1 - \rho(n))Q(s,a) + \rho(n)\left(R(s,a) + \theta(n)\max_b Q(e,b)\right), \quad b \in \{x^{\min}, x^{\max}\}.$$

11. To set the current state (amount of product inventory) $s = e$. Go to step 6.

12. If the best value of the goal function at the current iteration is less than the best value of the goal function for all previous iterations, i.e., $F(x) < F(x^*)$, then replace the best vector of the amount of goods purchased from the supplier, i.e., $x^* = x$.

13. If it isn't the last iteration, i.e., $n < N$, then go to step 3.

**SARSA-based method to solve the inventory management problem with dynamic parameters**

The SARSA (State Action Reward State Action) method with dynamic parameters consists of the following steps.

1. Initialization

1.1. To set the maximum number of iterations $N$, the number of stages $M$, the profit from the sale of one unit of goods $w1$, the cost of storing one unit of goods $w2$, the minimum and maximum amounts of goods purchased from the supplier $x^{\min}, x^{\max}$, the minimum and maximum amounts of goods inventories at the end of each stage $z^{\min}, z^{\max}$, the initial amount of goods inventories $z_0$, the amount of goods sold by stages $D_m$, $m \in \overline{1,M}$, parameters $\rho^{\min}, \rho^{\max}$ (to control the learning rate), $0 < \rho^{\min} < \rho^{\max} < 1$, parameters $\varepsilon^{\min}, \varepsilon^{\max}$ for the ε-greedy approach, $0 < \varepsilon^{\min} < \varepsilon^{\max} < 1$, parameters $\theta^{\min}, \theta^{\max}$ (to determine the importance of the future reward), $0 < \theta^{\min} < \theta^{\max} < 1$.

1.2. To initialize the reward table: $Q = [Q(i,j)]$, $Q(i,j) = 0$, $i, j \in \overline{1,M}$.

2. Iteration number *n*=1.

3. To calculate the parameters:

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

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

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

4. To set the initial state (initial amount of goods inventory) $s = z_0$.

5. To set the stage starting number $m = 1$.

6. To select an *a* action (the amount of goods purchased from the supplier), it is necessary to move from states, using the ε-greedy approach (if $U(0,1) < \varepsilon(n)$, then to select an action *a* randomly from the allowed actions set $\{x^{\min}, x^{\max}\}$, otherwise to select an *a* action: $a = \arg\max_b Q(s,b)$, $b \in \{x^{\min}, x^{\max}\}$.The *a* action becomes a new component of the vector of the amount of goods purchased from the supplier vector, i.e. $x_m = a$.

7. If the last stage $m = M$, then go to step 13, otherwise $m = m+1$.

8. To calculate the element of the current reward table $R(s,a)$:

$$R(s,a) = -(w1 \cdot \max\{0, z^{\min} - (a+s-D_m)\} + w2 \cdot \max\{0, a+s-D_m - z^{\max}\}).$$

9. There are a new state (the amount of the product inventory) $e = a + s - D_m$.

10. To select an *c* action (the amount of goods purchased from the supplier), it is necessary to move from state *e*, using the ε-greedy approach (if $U(0,1) < \varepsilon(n)$, then to select an action *c* randomly from the allowed actions set $\{x^{\min}, x^{\max}\}$, otherwise to select an *c* action: $c = \arg\max_b Q(e,b)$, $b \in \{x^{\min}, x^{\max}\}$.The *c* action becomes a new component of the vector of the amount of goods purchased from the supplier vector, i.e. $x_m = c$.

11. To calculate the element of the current reward table $Q(s,a)$:

$$Q(s,a) = (1-\rho(n))Q(s,a) + \rho(n)\big(R(s,a) + \theta(n)Q(e,c)\big).$$

12. To set the current action (the amount of goods purchased from the supplier) $a = c$. Go to step 7.

13. If the best value of the goal function at the current iteration is less than the best value of the goal function for all previous iterations, i.e., $F(x) < F(x^*)$, then replace the best vector of the amount of goods purchased from the supplier, i.e., $x^* = x$.

13. If it isn't the last iteration, i.e., $n < N$, then go to step 3.

**Algorithm to solve the inventory management problem based on deterministic dynamic programming**

An algorithm has been developed for implementation on a GPU using the CUDA parallel information processing technology. It is presented in Fig. 2. This block diagram has the following steps.

Step 1 – To specify a discrete set of the number of goods purchased from the supplier by the operator $X = \{x^{\min}, ..., x^{\max}\}$ with capacity $K$, the amount of goods sold during all stages $D = (D_{k1}, ..., D_{kM})$, the initial quantity of goods inventories $z_0$.

Step 2 – To set the function value $\varphi_0(z_0) = 0$.

Step 3 – To set the stage number $m = 1$.

Step 4 – To compute a functions combination $F_m(x_k, z_{m-1}) + \varphi_{m-1}(z_{m-1})$ using $K$ GPU threads, which are grouped into 1 block. Each thread calculates the function value $\psi_m(x_k, z_{m-1})$.

Step 5 – To determine the best quantity of the number of goods purchased from the supplier during the $m^{\text{th}}$ stage $x_m^*$, using $K$ GPU threads that are grouped into 1 block based on parallel reduction $x_m^* = \arg\min_{x_k} \psi_m(x_k, z_{m-1})$.

Step 6 – To calculate the function value: $\varphi_m(z_m) = \psi_m(x_m^*, z_{m-1})$.

Step 7 – To calculate the number of product inventory at the end of the $m^{\text{th}}$ stage $z_m = x_m^* + z_{m-1} - D_m$.

Step 8 – The stop conditions: If $m < M$, then $m = m+1$, and go to step 4.

Step 9 – To write the resulting global best position to the database.

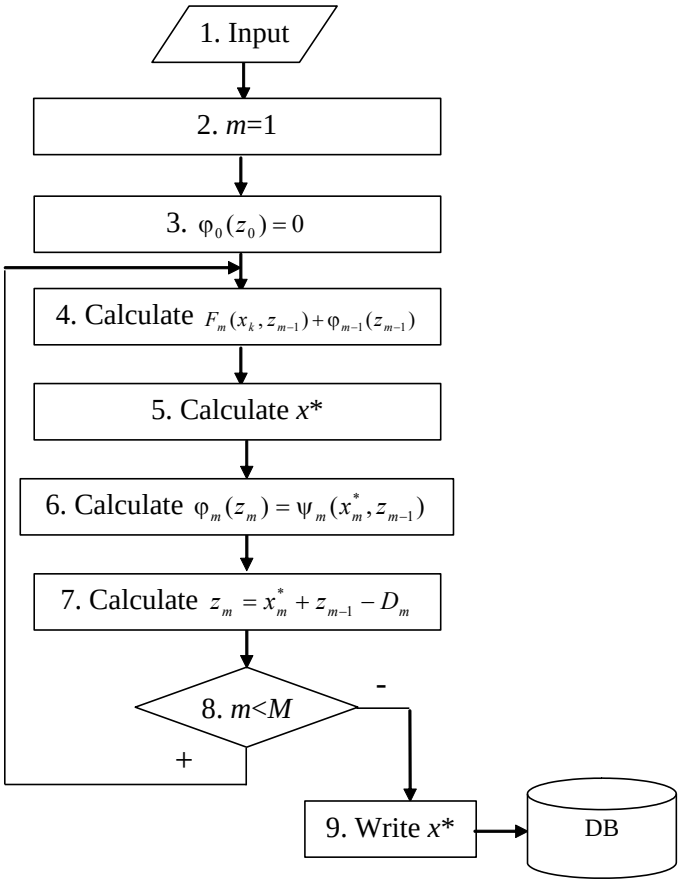


**Fig 2. Flowchart of the algorithm for solving the inventory management problem based on deterministic dynamic programming**

**Algorithm to solve the inventory management problem based on based on stochastic dynamic programming**

An algorithm is intended to be implemented on a GPU using the CUDA parallel information processing technology and is similar to the algorithm in Fig. 2. This block diagram has the following steps.

Step 1 – To specify a discrete set of the number of goods purchased from the supplier by the operator $X = \{x^{\min},...,x^{\max}\}$ with capacity $K$, the amount of goods sold during all stages $D = (D_{k1},...,D_{kM})$, the initial quantity of goods inventories $z_0$.

Step 2 – To set the function value: $\varphi_0(j) = 0$, $j \in \overline{1,3}$.

Step 3 – To set the stage number $m = 1$.

Step 4 – To compute a functions combination: $\sum_{j=1}^{3} p_{ij}(x_k, z_{m-1})(r_{ij}(x_k, z_{m-1}) + \varphi_{m-1}(j))$ using $K$ GPU threads, which are grouped into 1 block. Each thread calculates the function value $\psi_m(i, x_k, z_{m-1})$.

Step 5 – To determine the best quantity of the number of goods purchased from the supplier during the $m^{\text{th}}$ stage $x_m^*$, using $K$ GPU threads that are grouped into 1 block based on parallel reduction $x_m^* = \arg\min_{x_k} \psi_m(i, x_k, z_{m-1})$.

Step 6 – To calculate the function value: $\varphi_m(i) = \psi_m(i, x_m^*, z_{m-1})$, $i \in \overline{1,3}$.

Step 7 – To calculate the number of product inventory at the end of the $m^{\text{th}}$ stage: $z_m = x_m^* + z_{m-1} - D_m$

Step 8 – The stop conditions: If $m < M$, then $m = m+1$, and go to step 4.

Step 9 – To write the resulting global best position to the database.

**Experiments**

A numerical study was carried out using the Python package. For Q-learning methods, SARSA, dynamic parameters, the value of the parameters is $\rho^{min} = 0.1, \rho^{max} = 0.9$ (to control the learning rate), the value of the parameters is $\varepsilon^{min} = 0.1, \varepsilon^{max} = 0.9$ for the ε-greedy approach, the value of the parameters is $\theta^{min} = 0.1, \theta^{max} = 0.9$ (to determine the importance of the future reward).

The solution search was based on data from the logistics company “Ekol Ukraine”.

The parameter dependence $\theta(n)$ is defined as $\theta(n) = \theta^{min} + (\theta^{max} - \theta^{min})\frac{n-1}{N-1}$ (Fig. 3).

The parameter dependence $\theta(n)$ (Fig. 3) on the iteration number *n* shows that its share increases with the iteration number.

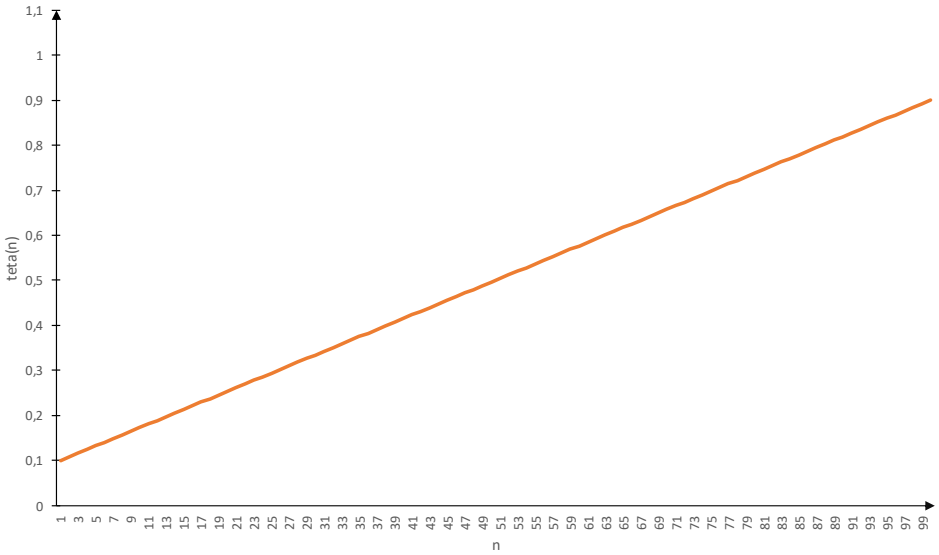


**Fig. 3. The dependence parameter $\theta(n)$ on the iteration number *n***

The parameters dependence $\rho(n)$ and $\varepsilon(n)$ is defined $\rho(n) = \rho^{max} - (\rho^{max} - \rho^{min})\frac{n-1}{N-1}$, and $\varepsilon(n) = \varepsilon^{max} - (\varepsilon^{max} - \varepsilon^{min})\frac{n-1}{N-1}$ (Fig. 4).

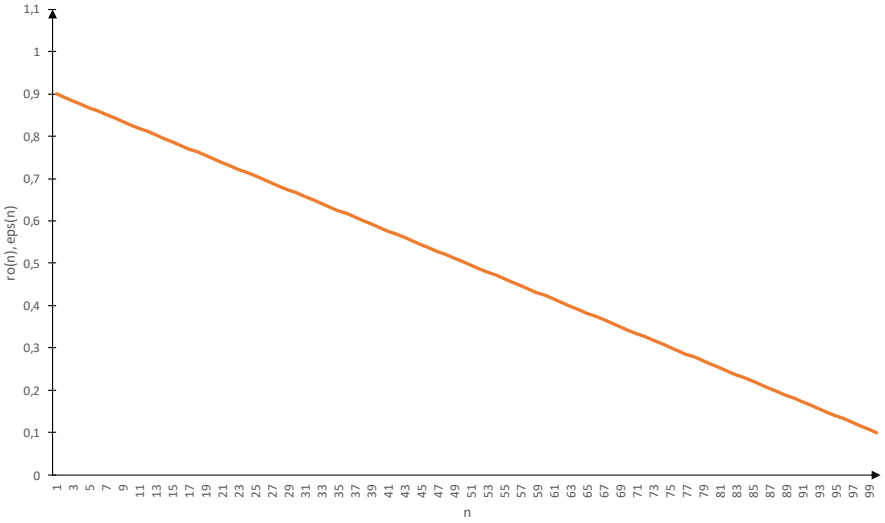


**Fig.4. The dependence parameters $\theta(n)$ and $\varepsilon(n)$ on the iteration number *n***

The parameters dependence $\rho(n)$ and $\varepsilon(n)$ (Fig. 4) on the iteration number $n$ shows that their share decreases with increasing iteration number.

The comparison results in methods of deterministic and stochastic dynamic programming based on CUDA technology (DDP and SDP with CUDA) with traditional methods of deterministic and stochastic dynamic programming (DDP and SDP without CUDA) using the computational complexity criterion (T) are presented in Table 1 ( $K$ – is the power of a discrete set of number of goods purchased from the supplier, $M$ – is the number of stages, $S$ – is the number of states of the inventory buffer, S=3.

Table 1

**Comparison of the methods based on CUDA technology with traditional ones**

| Criteria | Dynamic programming methods | | | |
|---|---|---|---|---|
| | DDP with CUDA | DDP without CUDA | SDP with CUDA | SDP without CUDA |
| $T$ | $M \cdot \log_2 K$ | $M \cdot K$ | $S^2 M \cdot \log_2 K$ | $S^2 M \cdot K$ |

The comparing results of the proposed Q-learning method with dynamic parameters and the traditional Q-learning method based on the mean square error criterion and the iterations number to solve the inventory management problem are presented in Table 2. Similar results were obtained for the proposed SARSA method with dynamic parameters and with the traditional SARSA method.

Table 2

**Comparison of the optimization method with the traditional Q-learning method based on the mean square error criterion and the iterations number to solve the inventory management problem**

| Method's mean square error | | Iterations number | |
|---|---|---|---|
| the proposed method | the current method | the proposed method | the current method |
| 0.05 | 0.05 | 300 | 2000 |

There are some advantages of the proposed methods.

1. The searching minimum process reduces the computational complexity due to parallel reduction (Table 1). The method's mean square error for all four methods was 0.02.SDP allows constructing a states visual diagram of the inventory buffer, but has greater computational complexity than DDP.

2. The Q-learning and SARSA methods modification makes it possible to increase the learning speed due to dynamic parameters while maintaining the method's mean square error (Table 2).

## Conclusions

1. The problem of inventory management as an integral part of effective supply chain management is examined in the research. Deterministic and stochastic dynamic programming methods were chosen to improve the quality-solving problem. Parallel algorithms based on CUDA technology were proposed for these methods. It was possible to ensure the high speed and accuracy of the solution, as well as to construct a states visual diagram of the inventory buffer.

2. The modification of Q-learning and SARSA reinforcement learning methods using dynamic programming with dynamic parameters in the reward table update rule was proposed which allows increasing the learning speed. The Q-learning and SARSA methods modification allows to ensure solution high accuracy due to the entire search space exploration in the initial iterations and the search direction in the final iterations.

3. The methods are intended for software implementation in the Matlab package using Parallel Computing Toolbox. This speeds up the finding solution process. The software according to methods implementation was developed and researched based on data from the logistics company "Ekol Ukraine". The experiments confirmed the performance of the developed software and allowed us to recommend it for solving supply chain management problems. are Methods testing on a wider set of test databases are the prospects for further research.

## References

1. Intellectualization of Lean Production Logistic Technology Based on Fuzzy Expert System and Multi-agent Metaheuristics. / E. Fedorov et al. *Lecture Notes in Networks and Systems*. 2022. Vol. 461. P. 447-462. Doi: 10.1007/978-981-19-2130-8_36.

2. Grygor O., Fedorov E., Nechyporenko O., Grygorian M. Neural network forecasting method for inventory management in the supply chain. *CEUR Workshop Proceedings*. 2022. Vol. 3137. P. 14-27.

3. Yang X.-S. Nature-inspired Algorithms and Applied Optimization. Charm: Springer, 2018. 330 p.

4. Yang X.-S. Optimization Techniques and Applications with Examples. Hoboken, New Jersey: Wiley & Sons, 2018. 364 p.

5. Optimization method based on the synthesis of clonal selection and annealing simulation algorithms / O. Grygor et al. *Radio Electronics, Computer Science, Control*. 2019. Vol. 2. P. 90-99.

6. Cetin E., Ball P. J., Roberts S., Celiktutan O. Stabilizing off-policy deep reinforcement learning from pixels. *Proceedings of the 39th International Conference on Machine Learning.* 2022. Vol. 162. P. 2784-2810.

7. Martí R., Pardalos P. M., Resende M. G. C. Handbook of Heuristics. Charm: Springer, 2018. 1289 p.

8. Bozorg-Haddad O., Solgi M., Loaiciga H. Meta-heuristic and Evolutionary Algorithms for Engineering Optimization. Hoboken, New Jersey: Wiley & Sons, 2017. 293 p.

9. Chopard B., Tomassini M. An Introduction to Metaheuristics for Optimization. New York: Springer, 2018. 230 p.
10. Radosavljević J. Metaheuristic Optimization in Power Engineering. New York: The Institution of Engineering and Technology, 2018. 536 p.
11. Bjorck N., Gomes C. P., Weinberger K. Q. Towards deeper deep reinforcement learning with spectral normalization. *35th Conference on Neural Information Processing Systems*. 2021. Vol. 34. P. 8242-8255.
12. Deep reinforcement learning at the edge of the statistical precipice. / R. Agarwal et al. *35th Conference on Neural Information Processing Systems*. 2021. Vol. 34. P. 29304-29320.
13. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play / D. Silver et al. *Science.* 2018. Vol. 362. P. 1140-1144. Doi: 10.1126/science.aar6404.
14. Gershman S. J., Daw N. D. Reinforcement learning and episodic memory in humans and animals: an integrative framework. *Annual Review of Psychology*. 2017. Vol. 68. P. 101-128. Doi: 10.1146/annurev-psych-122414-033625.
15. Vikbladh O., Shohamy D., Daw, N. Episodic contributions to model-based reinforcement learning. *Annual Conference on Cognitive Computational Neuroscience CCN*. 2017. P. 1-2. URL: https://ccneuro.org/abstracts/abstract_3000200.pdf.
16. Lukianykhin O., Bogodorova T. ModelicaGym: Applying Reinforcement Learning to Modelica Models. *9th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools.* 2019. Doi: 10.1145/3365984.3365985.
17. Reinforcement learning, fast and slow / M. Botvinick et al. *Trends in Cognitive Sciences.* Vol. 23 (5). P. 408-422. Doi: 10.1016/j.tics.2019.02.006 .
18. Moriyama T., De Magistris G., Tatsubori M., Pham T.-H., Munawar A., Tachibana R. Reinforcement Learning Testbed for Power-Consumption Optimization. *Asian Simulation Conference*. 2018. P. 45–59. Doi: 10.1007/978-981-13-2853-4_4.
19. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. 2nd ed. The MIT Press, 2018. 552 p.
20. Learning to reinforcement learn / J. X. Wang et al. Cornell University: Computer Science, Machine Learning, 2017. 17 p. arXiv:1611.05763. URL: https://doi.org/10.48550/arXiv.1611.05763.
21. Meta-gradient reinforcement learning with an objective discovered / Z. Xu et al. Cornell University: Computer Science, Machine Learning, 2020. 18 p. arXiv:2007.08433. URL: https://doi.org/10.48550/arXiv.2007.08433.
22. Barkovska O., Kholiev V. Neural Network Architecture for Text Decoding Based on Speaker's Lip Movements. *Computer Systems and Information Technologies*. 2023. Vol. 4. P. 52-59. Doi: 10.31891/csit-2023-4-7.
23. Mochurad L. CUDA-Based Parallelization of Gradient Boosting and Bagging Algorithm for Diagnosing Diabetes. *Computer Systems and Information Technologies*. 2023. Vol. 2. P. 6-16. Doi: 10.31891/csit-2023-2-1.
24. Boyko N., Kovalchuk R. Data Update Algorithms in the Machine Learning System. *Computer Systems and Information Technologies*. 2023. Vol. 1. P. 6-13. Doi: 10.31891/csit-2023-1-1.



| | | |
|---|---|---|
| **Eugene FEDOROV**<br>**Євген ФЕДОРОВ** | Doctor of Technical Science, Professor, Professor of Department Statistics and Applied Mathematics, Cherkasy State Technological University, Cherkasy, Ukraine<br>e-mail: fedorovee75@ukr.net<br>https://orcid.org/0000-0003-3841-7373<br>Scopus author ID: 47161155900, ResearcherID: AAO-6744-2020 | доктор технічних наук, професор, професор кафедри статистики та прикладної математики Черкаського державного технологічного університету, Черкаси, Україна |
| **Olga NECHYPORENKO**<br>**Ольга НЕЧИПОРЕНКО** | PhD, Associate Professor, Associate Professor of Department Robotics and Specialized Computer Systems, Cherkasy State Technological University, Cherkasy, Ukraine<br>e-mail: olne@ukr.net<br>https://orcid.org/0000-0002-3954-3796<br>Scopus author ID: 57216488854,<br>ResearcherID: ABB-2418-2021 | кандидат технічних наук, доцент, доцент кафедри робототехніки та спеціалізованих комп'ютерних систем Черкаського державного технологічного університету, Черкаси, Україна |
| **Tetiana NESKORODIEVA**<br>**Тетяна НЕСКОРОДЄВА** | Doctor of Technical Science, Associate Professor, Associate Professor of Department Mathematics and Physics, Uman National University of Horticulture, Uman, Ukraine<br>e-mail: tvnesk1@gmail.com<br>https://orcid.org/0000-0003-2474-7697<br>Scopus author ID: 57218242548, ResearcherID: S-5190-2017 | доктор технічних наук, доцент, професор кафедри математики і фізики Уманського національного університету садівництва, Умань, Україна |
| **Maryna LESHCHENKO**<br>**Марина ЛЕЩЕНКО** | PhD, Associate Professor, Associate Professor of International economics and business, Cherkasy State Technological University, Cherkasy, Ukraine<br>e-mail: mari.leshchenko@gmail.com<br>https://orcid.org/0000-0002-0210-9582<br>Scopus author ID: 57210600995, ResearcherID: AAH-6585-2021 | кандидат технічних наук, доцент, доцент кафедри міжнародної економіки та бізнесу Черкаського державного технологічного університету, Черкаси, Україна |