

ГИБРИДНЫЙ АЛГОРИТМ ДЛЯ ПЛАНИРОВАНИЯ ЗАГРУЗКИ ТЕХНОЛОГИЧЕСКИХ МАШИН В ПРОИЗВОДСТВЕННЫХ ПРОЦЕССАХ

A HYBRID ALGORITHM FOR LOAD PLANNING OF TECHNOLOGICAL MACHINES IN PRODUCTION PROCESSES

А.А. АБУЗЯРОВ, А.А. МАКАРОВ

A.A. ABUZYAROV, A.A. MAKAROV

(Российский государственный университет имени А.Н. Косыгина (Технологии. Дизайн. Искусство))

(The Kosygin State University of Russia)

E-mail: theschoolofrock9219@gmail.com, makarov-aa@rguk.ru

В данной работе представлен гибридный алгоритм MCDDQ-SA, объединяющий метод Монте-Карло с деревом поиска и верхним доверительным интервалом (MCTS UCB), глубокое Q-обучение с двойной оценкой (DDQN) и имитацию отжига (SA), для решения задачи планирования загрузки технологических машин в текстильном производстве. Цель исследования – минимизация общего времени завершения всех задач (makespan) и времени простоя машин (idle time), а также обеспечение сбалансированной нагрузки между машинами. Алгоритм протестирован на модели с разным набором задач и машин и сравнен с традиционным методом смешанного целочисленного линейного программирования (MILP) и Жадным алгоритмом (Greedy). Результаты показывают, что MCDDQ-SA обеспечивает конкурентоспособные значения makespan и idle time при существенно меньшем времени вычислений, что делает его перспективным для применения в динамичных производственных условиях.

In this paper, a hybrid MCDDQ-SA algorithm combining Monte Carlo tree search method with upper confidence interval (MCTS UCB), deep Q-learning with double scoring (DDQN) and simulated annealing (SA) is presented for solving the problem of manufacturing process machine scheduling. The objective of the study is to minimize the total task completion time (makespan) and machine idle time (idle time), as well as to ensure load balancing among machines. The algorithm is tested on a model with 20 process machines and 30 tasks and compared with the traditional mixed integer linear programming (MILP) method. The results show that MCDDQ-SA provides competitive makespan and idle time values with significantly lower computation time, which makes it promising for application in dynamic manufacturing environments.

Ключевые слова: машинное обучение, Q-обучение, глубокие нейронные сети, обучение с подкреплением, MCTS UCB, имитация отжига, планирование, DDQN, Greedy.

Keywords: machine learning, Q-learning, deep neural networks, reinforcement learning, MCTS UCB, simulated annealing, scheduling, DDQN, Greedy.

Введение

Задача планирования загрузки технологических машин играет ключевую роль в оптимизации производственных процессов [1], где необходимо учитывать неоднородные характеристики оборудования и строгие временные ограничения. Традиционные методы, такие как смешанное целочисленное линейное программирование (MILP) [2], обеспечивают точные решения задач планирования, но их высокая вычислительная сложность ограничивает применимость в крупномасштабных или динамичных задачах. В последние годы эвристические и гибридные подходы [3], включая генетические алгоритмы [4], имитацию отжига и обучение с подкреплением, показали свою эффективность в нахождении субоптимальных решений за приемлемое время [5, 6].

В данной работе рассматривается задача оптимального распределения двух наборов задач (20, 30 задач и 10, 20 машин). Основные цели – минимизация makespan (общего времени завершения всех задач), idle time (времени простоя машин) и обеспечение сбалансированной нагрузки.

Для решения этой задачи разработан гибридный алгоритм MCDDQ-SA [7], объединяющий MCTS UCB [8], DDQN [9] и SA. Этот подход сочетает в себе исследование пространства решений, обучение оптимальной политике планирования и локальную оптимизацию, что делает его эффективным инструментом для реальных производственных сценариев.

Методы исследования

Проблема параллельного планирования загрузки технологических машин (Parallel Machine Scheduling Problems) – это классическая задача комбинаторной оптимизации, цель которой заключается в распределении набора заданий по множеству технологических машин с учетом их неоднородных характеристик. Основная цель – минимизация общего времени выполнения (makespan), времени простоя (idle time) и обеспечение сбалансированной нагрузки между машинами. Задача моделируется как марковский процесс принятия решений (MDP) [10], определяемый кортежем (S, A, R, γ):

- S: Пространство состояний, включающее назначения задач, загрузку технологических машин и их состояние (отказы, время восстановления). Чем выше вероятность отказа машины, тем ниже вероятность назначения задачи на машину.
- A: Пространство действий, представляющее выбор пары задача-технологическая машина.
- R: Функция награды, учитывающая makespan, idle time и дисперсию нагрузки.
- γ : Дисконтный фактор, определяющий значимость будущих наград.

Такая формализация позволяет применять методы RL для поиска оптимальной политики планирования.

Формальное описание

- Набор из n задач $T = \{T_1, T_2, \dots, T_n\}$, каждая с длительностью d_i ($i = 1, \dots, n$).
- Набор из m технологических машин $M = \{M_1, M_2, \dots, M_m\}$, каждая с производительностью (скоростью) s_j ($j = 1, \dots, m$).
- Вероятность отказа технологической машины p_f , зависящая от текущей нагрузки, и максимальное время восстановления t_r .

Цель

Назначить каждую задачу T_i на одну технологическую машину M_j , минимизируя многокритериальную целевую функцию, включающую:

- Makespan: $C_{max} = \max_j \{C_j\}$, где C_j – время завершения всех задач на машине M_j .
- Idle Time: Суммарное время простоя, $\sum_j \max(0, C_{max} - C_j)$, с учетом штрафов за отказы.
- Load Variance: Дисперсия времени работы технологических машин, $var(C_j)$, отражающая сбалансированность нагрузки оборудования. При высоком дисбалансе нагрузки происходит более сильный износ технологической машины.

Награда (Целевая функция)

- Промежуточная награда: $r = -0.05 * load_variance - 0.1 * current_makespan - 0.2 * failure_risk - 0.2 * current_idle_time$, учитывающая текущие метрики, где $failure_risk$ – вероятность отказа машины.
- Итоговая награда: $r = -makespan - 3.0 * idle_time - 0.1 * load_variance$, отражающая финальное качество расписания.

Ограничения

- Каждая задача назначается ровно на одну технологическую машину.
- Машина в состоянии отказа недоступна до завершения восстановления.
- Время обработки задачи на машине M_j равно d_i/s_j .

Особенности задачи

- NP-трудность: Комбинаторная природа задачи делает ее вычислительно сложной, особенно при большом числе заданий и технологических машин.
- Неоднородность технологических машин: Скорости машин различаются, что требует поиска компромисса между использованием быстрых машин и равномерным распределением нагрузки.
- Многокритериальность: Решение задачи направлено на достижение краткосрочной эффективности (минимизация makespan) и долгосрочной сбалансированности (минимизация idle time и дисперсии нагрузки).

Работа модели начинается с инициализации, где все операции не назначены, а времена работы машин равны нулю. На каждом шаге алгоритм назначает операцию на технологическую машину, обновляя состояние и вычисляя метрики (makespan, idle time, дисперсию нагрузки). Модель предоставляет награды для оптимизации, а также поддерживает визуализацию текущего состояния.

MILP

MILP (метод смешанного целочисленного программирования) – это метод оптимизации, где часть переменных должна быть целыми числами, а часть – непрерывными. Задача состоит в том, чтобы найти

значения переменных, которые минимизируют или максимизируют линейную функцию (см. Награда), соблюдая линейные ограничения (отказы и т. д.)

MILP, реализованный с использованием библиотеки PuLP, формулирует задачу планирования как задачу оптимизации. Бинарные переменные x_{ij} обозначают назначение задачи j на машину i . Время работы машины t_i вычисляется как $t_i = \sum_j x_{ij} * \frac{d_j}{s_i}$. Целевая функция минимизирует основные критерии (см. Награда).

Особенности:

- Точный метод, решающий задачу до глобального оптимума.
- Учитывает время простоя в целевой функции, что соответствует метрикам среды Parallel Machine Scheduling Env.

Преимущества:

- MILP гарантирует нахождение глобально оптимального решения при выполнении всех заданных ограничений.
- Высокая точность, что идеально для эталонного сравнения.

Ограничения:

- Задача относится к классу NP-трудных, и время решения экспоненциально растет с увеличением числа заданий и машин.
- Для больших задач или динамических систем MILP может быть слишком медленным, что делает его неприменимым в производственных условиях с частыми изменениями.

Greedy

Greedy (Жадный алгоритм) – алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе при допущении, что конечное решение также окажется оптимальным.

Процесс назначения:

- Для каждой задачи (в порядке от 0 до T-1).
- Перебираются все машины (от 0 до M-1).
- Вычисляется время завершения задачи на каждой машине.
- Выбирается машина с минимальным временем завершения.
- Задача назначается на эту машину.

Особенности:

- Локальная оптимизация без учета долгосрочных последствий.

Преимущества:

- Низкая вычислительная сложность.
- Простота реализации.

Ограничения:

- Субоптимальные решения.
- Низкая устойчивость к отказам из-за отсутствия глобального планирования.

MCDDQ-SA

MCDDQ-SA – это гибридный алгоритм, объединяющий Monte Carlo Tree Search (MCTS) с Upper Confidence Bound (UCB), Deep Double Q-Network (DDQN) и Simulated Annealing (SA), для решения задачи параллельного планирования машин. Он эффективно балансирует глобальный поиск оптимальных решений и локальную оптимизацию.

Процесс принятия решений:

- DDQN вычисляет Q-значения для доступных действий (назначение задачи на машину).
- SA оценивает действия на основе энергии, учитывающей загрузку, вероятность отказов и дисперсию нагрузки, и задает вероятности выбора.
- MCTS использует UCB, комбинируя Q-значения и вероятности SA, для выбора действия, балансируя глобальный и локальный поиск.

После обучения агента DDQN на этапе выполнения (поиска решения) алгоритм работает следующим образом:

- MCTS выполняет симуляции (rollouts) для оценки действий, балансируя исследование и эксплуатацию с помощью формулы Upper Confidence Bound (UCB):

$$UCB = 0.7 * Q(s, a) + 0.3 * E(s, a) + w_{exp} * \sqrt{\frac{\ln N(s)}{N(s, a)}}$$

где $Q(s, a)$ – Q-значение от DDQN; $E(s, a)$ – энергия действия от SA; $N(s)$ и $N(s, a)$ – количество посещений состояния и действия; $w_{exp} = 30 * (1 - step/num_tasks)$ – адаптивный вес исследования.

- При отсутствии улучшений makespan в течение 20 итераций температура сбрасывается до $0.5 * T_{initial}$, что предотвращает застревание в локальных минимумах.

Гибридный выбор действий:

- Действия выбираются с учетом веса SA ($w_{sa} = \max(0.3, T/T_{initial})$).
- Если $T > 0.7 * T_{initial}$ или с вероятностью 0.01, действие выбирается вероятностно по $P(a)$ (SA-режим).
- Иначе выбирается действие с максимальным комбинированным значением: $(1 - w_{sa}) * UCB + w_{sa} * P(a)$.
- Это обеспечивает баланс между детерминированным выбором (UCB) и вероятностным (SA).
- Лучшее действие определяется на основе среднего значения наград, скорректированного бонусом SA ($0.2 * P(a)$), что дополнительно учитывает вероятности SA при обновлении статистики MCTS.

Результаты и обсуждения

Сравнение проводилось на трех конфигурациях в два этапа:

- Первый этап (Тестирование при вероятности отказа $p = 0$): 20 задач/10 машин и 30 задач/20 машин.
- Второй этап (Тестирование при вероятности отказа $p = 0.05$): 20 задач/10 машин.

Машины имеют различные характеристики (скорость обработки, время наладки), а задания различаются по длительности и требованиям.

Для количественной оценки производительности алгоритмов используются следующие метрики, каждая из которых отражает ключевые аспекты качества планирования:

- Makespan: максимальное время завершения всех задач. Минимизация makespan является основной целью планирования.
- Idle Time: суммарное время простоя машин, включая время, вызванное отказами.
- Время выполнения: время работы алгоритма.

Все метрики вычисляются для каждого запуска эксперимента и усредняются по сериям запусков для учета случайных факторов.

Тестирование

Первый этап: 10 машин, 20 задач

- Каждый алгоритм запускался 10 раз для получения статистически значимых данных. Использовались фиксированные значения скоростей машин и длительностей задач.

- $machine_speeds = [1.28, 0.79, 1.35, 1.31, 0.73, 1.25, 0.53, 0.79, 1.46, 0.66]$ – скорости машин.
- $task_durations = [76.54, 29.02, 84.37, 79.23, 23.56, 71.99, 12.44, 29.03, 91.98, 20.32, 55.85, 89.91, 37.37, 48.92, 91.36, 47.27, 32.85, 70.48, 87.38, 16.12]$ – длительности задач.
- Вероятность отказа $p = 0$.

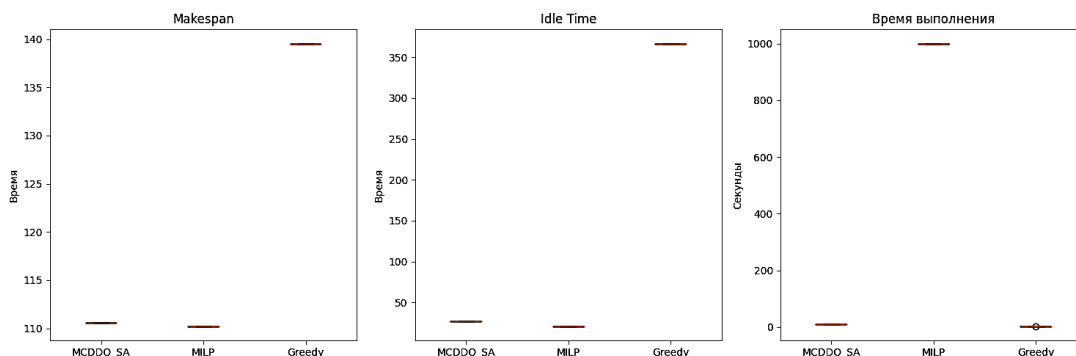


Рис.1

Таблица 1

Алгоритм	Makespan	Idle Time	Время выполнения, с
MCDDQ_SA (DDQN + MCTS+ SA)	110.61 ± 0.00	26.60 ± 0.00	10.78
Greedy	139.55 ± 0.00	366.72 ± 0.00	2.18
MILP	110.2 ± 0.0	20.55 ± 0.0	1000.1

Из рис. 1 и табл. 1 видно, что:

- Makespan: MILP (110.2) незначительно лидирует, но MCDDQ-SA (110.61) близок и превосходит Greedy (139.55).
- Idle Time: MILP (20.55) также незначительно опережает MCDDQ-SA (26.6), но MCDDQ-SA значительно лучше Greedy (366.72).
- Время выполнения: MCDDQ-SA значительно быстрее MILP (10.78 секунд против 1000.1 секунд), но уступает в скорости Greedy (2.18 секунд).

Первый этап: 20 машин, 30 задач

- Каждый алгоритм запускался также 10 раз для получения статистически значимых данных.
- $machine_speeds = [1.28, 0.79, 1.35, 1.31, 0.73, 1.25, 0.53, 0.79, 1.46, 0.66, 0.95, 1.10, 0.85, 1.20, 0.70, 1.30, 0.90, 1.05, 0.80, 1.15]$.

- $task_durations = [76.54, 29.02, 84.37, 79.23, 23.56, 71.99, 12.44, 29.03, 91.98, 20.32, 55.85, 89.91, 37.37, 48.92, 91.36, 47.27, 32.85, 70.48, 87.38, 16.12, 65.20, 40.15, 80.75, 25.60, 50.45, 95.30, 35.80, 60.10, 45.25, 70.90]$.
- Вероятность отказа $p = 0$.

Из рис. 2 и табл. 2 видно, что:

- Makespan: MILP демонстрирует лучшее значение (91.83 против 93.21), но разница незначительна.
- Idle Time: MILP также превосходит MCDDQ-SA (185.05 против 212.17), однако MCDDQ-SA остается конкурентоспособным, значительно опережая Greedy.
- Время выполнения: MCDDQ-SA в 58 раз быстрее MILP (20.70 секунд против 1200.1 секунд), но все также уступает Greedy.

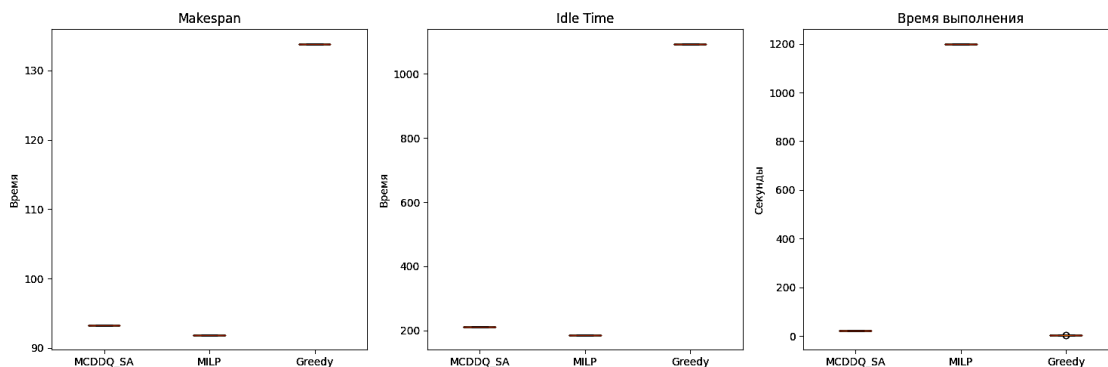


Рис.2

Т а б л и ц а 2

Алгоритм	Makespan	Idle Time	Время выполнения, с
MCDDQ-SA (DDQN + MCTS + SA)	93.21 ± 0.00	212.17 ± 0.00	20.70
MILP	91.83 ± 0.00	185.05 ± 0.00	1200.1
Greedy	133.83 ± 0.00	1093.81 ± 0.00	3.30

Второй этап (Отказ Машины 3): 10 машин, 20 задач

- Каждый алгоритм запускался также 10 раз для получения статистически значимых данных.
- machine_speeds = [1.28, 0.79, 1.35, 1.31, 0.73, 1.25, 0.53, 0.79, 1.46, 0.66] – скорости машин.

- task_durations = [76.54, 29.02, 84.37, 79.23, 23.56, 71.99, 12.44, 29.03, 91.98, 20.32, 55.85, 89.91, 37.37, 48.92, 91.36, 47.27, 32.85, 70.48, 87.38, 16.12] – длительности задач.
- Вероятность отказа $p = 0.05$ для машины 3.

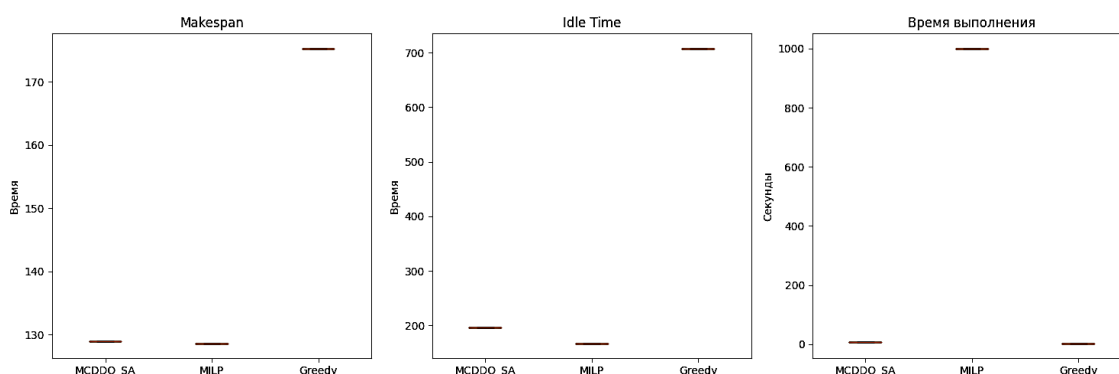


Рис.3

Т а б л и ц а 3

Алгоритм	Makespan, мин	Idle Time, мин	Время выполнения, с
MCDDQ-SA (DDQN + MCTS + SA)	128.96 ± 0.00	195.93 ± 0.00	7.64x2
Greedy	175.27 ± 0.00	707.67 ± 0.00	2.17x2
MILP	128.56 ± 0.00	166.93 ± 0.00	1000.5x2

Из рис. 3 и табл. 3 видно, что:

- Makespan: MILP лидирует, за ним следует MCDDQ-SA, Greedy.
- Idle Time: MILP опережает MCDDQ-SA, но MCDDQ-SA значительно лучше Greedy. SA улучшает адаптивность к отказам.

- Время выполнения: MCDDQ-SA значительно быстрее MILP, но немного медленнее Greedy.

Алгоритм приближается к оптимальным результатам MILP. Нулевое стандартное отклонение в большинстве сценариев указывает на стабильность результатов.

зывает на высокую надежность и стабильность MCDDQ-SA.

Основные результаты исследования

- Качество решений: MCDDQ-SA демонстрирует хорошую производительность, приближаясь к MILP по makespan и idle time и превосходя Greedy во всех сценариях, особенно при отказах машин. Его стабильность и адаптивность делают его перспективным для производственных систем с динамическими отказами.
- Скорость вычислений: Время выполнения MCDDQ-SA составило 20.70 секунд (при задаче 20 машин, 30 задач), что в 58 раз быстрее, чем 1200.1 секунд у MILP. Это делает алгоритм пригодным для использования в реальном времени, что особенно важно в условиях динамического производства.
- Адаптивность подхода: Комбинация глобального поиска (MCTS и DDQN) и локальной оптимизации (SA) позволяет MCDDQ-SA эффективно справляться с NP-трудными задачами, включая неоднородные машины и сложные ограничения. Такой подход демонстрирует гибкость, которая может быть полезна в различных производственных и непроизводственных сценариях.

Практическая значимость

- Применение в реальном времени: Высокая скорость MCDDQ-SA позволяет оперативно реагировать на изменения в производственных процессах, такие как сбои оборудования или срочные заказы. Это делает его ценным инструментом для промышленности, где время играет критическую роль. При тестировании этапа 2 (отказ Машины 3) особенно видно преимущество алгоритма MCDDQ-SA перед MILP. Алгоритм тратит на первичное построение плана 7.64 секунд против 1000.1 секунд (MILP) и на перестроение плана после отказа Машины 3 все те же 7.64 секунд против 1000.1 секунд (всего 15.28 секунд против 2000.2 секунд), что ком-

пенсировать небольшое отклонение от результатов Makespan и Idle Time.

- Снижение вычислительных затрат: В отличие от MILP, требующего значительных ресурсов для крупных задач, MCDDQ-SA обеспечивает экономию времени и вычислительных мощностей, что особенно актуально для предприятий с ограниченными техническими возможностями.
- Универсальность: Успех алгоритма в задаче планирования указывает на его потенциал для адаптации к другим комбинаторным задачам, включая управление запасами, планирование персонала или оптимизацию производственных линий.

Заключение

Разработан и протестирован гибридный алгоритм MCDDQ-SA, сочетающий метод Монте-Карло с деревом поиска (MCTS), глубокое Q-обучение с двойной оценкой (DDQN) и имитацию отжига (SA). Алгоритм применен к задаче планирования загрузки технологических машин в производственном процессе, где целью являлась минимизация общего времени завершения задач (makespan), времени простоя машин (idle time) и обеспечение сбалансированной нагрузки. Установлено, что гибридный алгоритм MCDDQ-SA представляет собой эффективное решение для задачи загрузки технологических машин, обеспечивая баланс между качеством и скоростью вычислений. Его практическая значимость заключается в возможности применения в реальном времени, что особенно ценно для динамичных производственных сред.

ЛИТЕРАТУРА

1. Arnaout J. P., Rabadi G., Musa R. A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times // Journal of Intelligent Manufacturing. 2010. Vol. 21, № 6. P. 693...701.
2. Bellman R. Dynamic Programming. – Princeton University Press, 1957. 342 с.
3. Anghinolfi D., Paolucci M. A hybrid metaheuristic for the parallel machine scheduling problem with sequence dependent setup times // Journal of Scheduling. 2009. Vol. 12, № 4. P. 357...372.
4. Zhong W., Xu J. A hybrid genetic algorithm for parallel machine scheduling with sequence-dependent

setup times // Journal of Intelligent Manufacturing. 2019. Vol. 30, № 4. P. 1573...1586.

5. *Deng J., Li K., Irwin G. W.* Fast automatic tuning of a simulated annealing algorithm // IEEE Transactions on Automation Science and Engineering. 2012. Vol. 9, № 2. P. 305...315.

6. *Sutton R.S., Barto A.G.* Reinforcement Learning: An Introduction. 2nd ed. – MIT Press, 2018. 552 с.

7. *Абузяров А.А., Макаров А.А.* Сравнение алгоритмов МСТS, МСDDQ, МСDDQ-SA, Greedy в рамках задачи параллельного планирования загрузки машин на производстве // Инженерный вестник Дона. 2025. №6. – <http://www.ivdon.ru/magazine/archive/n6y2025/10139>

8. *Browne C.B., Powley E., Whitehouse D. et al.* A survey of Monte Carlo Tree Search methods // IEEE Transactions on Computational Intelligence and AI in Games. 2012. Vol. 4, № 1. P. 1...43.

9. *Liu Wentao, Tang Tao, Wang Xi.* A DQN-based intelligent control method for heavy haul trains on long steep downhill section, Knowledge Based Systems, 2021. vol. 129. – <https://www.sciencedirect.com/science/article/abs/pii/S0968090X2100262X>.

10. *Puterman M.L.* Markov Decision Processes: Discrete Stochastic Dynamic Programming. – John Wiley & Sons, 2014. 684 с.

REFERENCES

1. *Arnaout J.P., Rabadi G., Musa R.* A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times // Journal of Intelligent Manufacturing. 2010. Vol. 21, № 6. P. 693...701.

2. *Bellman R.* Dynamic Programming. – Princeton University Press, 1957. 342 с.

3. *Anghinolfi D., Paolucci M.* A hybrid metaheuristic for the parallel machine scheduling problem with sequence dependent setup times // Journal of Scheduling. 2009. Vol. 12, № 4. P. 357...372.

4. *Zhong W., Xu J.* A hybrid genetic algorithm for parallel machine scheduling with sequence-dependent setup times // Journal of Intelligent Manufacturing. 2019. Vol. 30, № 4. P. 1573...1586.

5. *Deng J., Li K., Irwin G.W.* Fast automatic tuning of a simulated annealing algorithm // IEEE Transactions on Automation Science and Engineering. 2012. Vol. 9, № 2. P. 305...315.

6. *Sutton R.S., Barto A.G.* Reinforcement Learning: An Introduction. 2nd ed. – MIT Press, 2018. 552 с.

7. *Абузяров А.А., Макаров А.А.* Comparison of МСТS, МСDDQ, МСDDQ-SA, Greedy algorithms in the context of the problem of parallel planning of machine loading in production // Engineering journal of Don. 2025. №6. – <http://ivdon.ru/en/magazine/archive/n6y2025/10139>

8. *Browne C.B., Powley E., Whitehouse D. et al.* A survey of Monte Carlo Tree Search methods // IEEE Transactions on Computational Intelligence and AI in Games. 2012. Vol. 4, № 1. P. 1...43.

9. *Liu Wentao, Tang Tao, Wang Xi.* A DQN-based intelligent control method for heavy haul trains on long steep downhill section, Knowledge Based Systems, 2021. vol. 129. – <https://www.sciencedirect.com/science/article/abs/pii/S0968090X2100262X>.

10. *Puterman M.L.* Markov Decision Processes: Discrete Stochastic Dynamic Programming. – John Wiley & Sons, 2014. 684 с.

Рекомендована кафедрой автоматизации и промышленной электроники РГУ им. А.Н. Косыгина. Поступила 10.06.25.