DOI: 10.18372/2310-5461.63.18950
УДК 004.8

Т. В. Холявкіна, канд. техн. наук, доц.
Національний Авіаційний Університет
orcid.org/0000-0003-2595-9405
e-mail: holyavkina.t@gmail.com;

Я. В. Троцький
Національний Авіаційний Університет
orcid.org/0009-0000-4725-9544
e-mail: 4640007@stud.nau.edu.ua;

Ю. Б. Моденов, канд. техн. наук, доц.
Національний Авіаційний Університет
orcid.org/0000-0003-3898-4159
e-mail: yurii.modenov@npp.nau.edu.ua

АДАПТИВНІ АЛГОРИТМИ УПРАВЛІННЯ ТЕХНОЛОГІЇ IOT В УМОВАХ ОБМЕЖЕНИХ РЕСУРСІВ

Вступ

Швидкий розвиток Інтернету речей (Internet of Things - IoT) перетворює наш спосіб сприймання технологій і взаємодії з ними. За допомогою підключених пристроїв, датчиків та систем збірних даних, IoT відкриває безмежні можливості для вдосконалення різноманітних сфер життя, від побутового комфорту до промислового виробництва. Проте, разом із зростанням кількості підключених пристроїв і обсягу згенерованих даних виникають нові виклики, зокрема, ефективне управління цими масивами інформації та ресурсами.

IoT можна розглядати як мережеву концепцію, що складається з підключених пристроїв з вбудованими передавачами і програмним забезпеченням, які дозволяють передавати дані і обмінюватися ними між світом і комп'ютерними системами в автоматизованому режимі за допомогою стандартних протоколів зв'язку. Крім передавачів, мережі можуть також включати виконавчі пристрої, вбудовані у фізичні об'єкти і з'єднані між собою за допомогою дротових або бездротових мереж. Ці взаємопов'язані пристрої мають можливість зчитувати і приводити в дію, програмувати та ідентифікувати, а використання інтелектуальних інтерфейсів робить втручання людини непотрібним [1].

Адаптація в контексті Інтернету Речей (IoT) може розглядатися як важлива складова управління системами IoT. Вона включає в себе вміння системи адаптуватися до змінюваних умов і потреб користувача, а також до нових даних та інформації.

1. Адаптація до змінних умов

Динамічне налаштування: IoT пристрої повинні мати можливість автоматично коригувати свої параметри або поведінку в залежності від змінюваних умов навколишнього середовища чи вимог системи.

Обробка даних в реальному часі: Після збору даних з сенсорів, система може здійснювати адаптацію на основі аналізу цих даних, наприклад, коригуючи роботу пристроїв або змінюючи алгоритми обробки.

2. Адаптація до нових даних і сценаріїв використання

Машинне навчання: Використання алгоритмів машинного навчання для вдосконалення прогнозування і прийняття рішень. Система може навчатися на основі нових даних і поліпшувати свою ефективність з часом.

Гнучкість алгоритмів: Можливість адаптації алгоритмів управління або обробки даних для нових сценаріїв або завдань.

3. Адаптація до користувацьких потреб

Персоналізація: Пристрої можуть змінювати свої налаштування відповідно до вподобань користувача або історії використання.

Інтерфейси: Адаптація інтерфейсів для зручності користування, враховуючи індивідуальні уподобання та потреби.

4. Безпека і приватність

Адаптація до загроз: Системи повинні адаптувати свої механізми безпеки відповідно до нових загроз або вразливостей, які можуть з'являтися з часом.

Актуалізація політик безпеки: Впровадження нових політик захисту даних та конфіденційності на основі змін у нормативних вимогах або політиках.

У сучасному світі технології Інтернету речей (IoT) знаходять все більше застосувань у різних сферах. Наприклад:

1. Смарт-будинки: У сучасних смарт-будинках встановлюються різноманітні пристрої, такі як термостати, освітлення, дверні замки, відеокамери, датчики руху тощо. Ці пристрої взаємодіють між собою та з користувачем через мережу Інтернет, дозволяючи зручно контролювати та автоматизувати різні аспекти життя вдома.

2. Медицина: У галузі охорони здоров'я Інтернет речей використовується для розробки розумних медичних пристроїв, які можуть відстежувати стан здоров'я пацієнта в режимі реального часу і передавати дані лікареві для аналізу та діагностики. Приклади включають в себе носимі пристрої для вимірювання частоти серцевих скорочень, монітори артеріального тиску і глюкометри для вимірювання рівня глюкози в крові.

3. Промисловість (індустрія 4.0): У сфері промислового виробництва IoT використовується для створення «розумних» заводів, де підключені датчики та пристрої надають можливість відстежувати та контролювати процеси виробництва в реальному часі. Це може включати в себе моніторинг обладнання для передбачення збоїв, оптимізацію процесів виробництва для підвищення продуктивності та зниження витрат, а також впровадження систем автоматизації та роботизації.

4. Транспорт і логістика: У сфері транспорту та логістики IoT використовується для створення «розумних» транспортних систем, які дозволяють відстежувати рух транспортних засобів, оптимізувати маршрутизацію та управління логістичними процесами. Наприклад, системи відстеження вантажів, моніторинг стану транспортних засобів, системи управління трафіком тощо.

5. Сільське господарство: У галузі сільського господарства IoT використовується для створення "розумних" сільськогосподарських систем, які дозволяють відстежувати та контролювати різні аспекти сільського господарства. Це може включати в себе моніторинг стану ґрунту та рослин, автоматизацію поливу, відслідковування виробництва та доставки сільськогосподарської продукції тощо [2].

Обмеженість ресурсів є однією з ключових проблем, яку необхідно вирішувати при розробці технологій IoT. Зокрема, підключені пристрої часто мають обмежені ресурси енергії, обчислювальні можливості та мережеву пропускну здатність. Вирішення цих проблем вимагає розробки ефективних алгоритмів управління, які здатні адаптуватися до змінних умов і забезпечувати оптимальне використання доступних ресурсів.

Адаптивні алгоритми управління технологією IoT в умовах обмежених ресурсів – це програмні або апаратні методи та стратегії, які дозволяють оптимально використовувати обмежені ресурси, такі як енергія, обчислювальна потужність та мережева пропускна здатність, для ефективного управління системами IoT. Основна мета таких алгоритмів полягає в тому, щоб забезпечити високий рівень функціональності та ефективності системи при обмежених ресурсах [3].

Постановка завдання

Розвиток Інтернету речей (IoT) призводить до збільшення кількості пристроїв, що взаємодіють для збору, обміну та аналізу даних, але більшість з них мають обмеження щодо енергоспоживання, обчислювальних потужностей і пам'яті. Ці обмеження ускладнюють забезпечення стабільної роботи IoT систем, особливо з урахуванням вимог до масштабованості та продуктивності.

Відсутність ефективних механізмів управління ресурсами може спричинити затримки в роботі пристроїв, підвищене енергоспоживання та втрату зв'язку. Тому виникає необхідність створення алгоритмів, які б динамічно адаптували роботу IoT пристроїв до умов обмежених ресурсів, зберігаючи стабільну продуктивність та оптимізуючи використання енергії.

Аналіз останніх досліджень і публікацій

В останні роки відбулось помітне зміщення пристроїв, які раніше не вважались частиною технологій IoT. Прогнозують, що до 2030 року близько 75 % світових пристроїв будуть так чи інакше відноситись до технологій IoT (рис. 1).

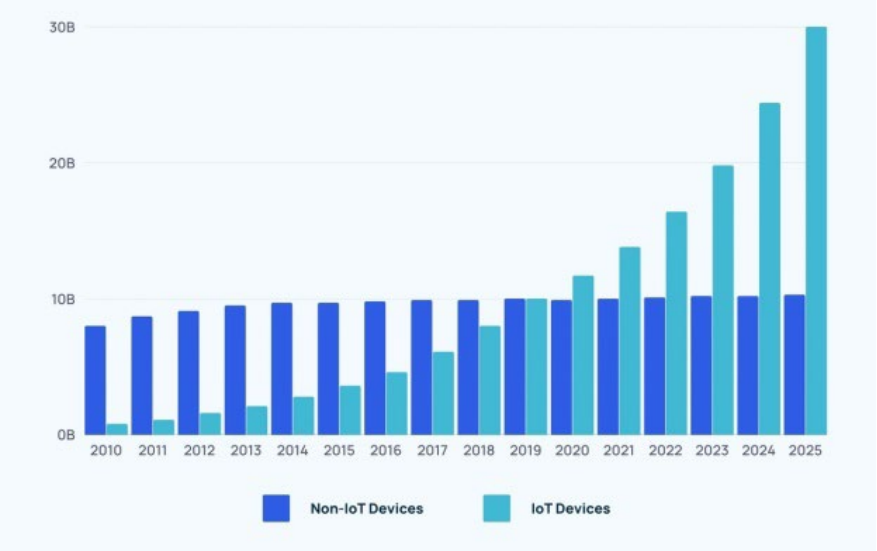


Рис. 1. Активність пристроїв, поза IoT та таких що відносяться до IoT в період з 2010 по 2025 роки [4]

Цю ж статистику можна відобразити у абсолютних числах (табл. 1).

Таблиця 1

Активність пристроїв у абсолютних числах

Рік	Пристрої поза IoT (мільярдів)	Пристрої IoT (мільярдів)	Відсоток пристроїв IoT по відношенню до загальної кількості пристроїв
2010	8	0,8	9 %
2011	8,7	1,1	11,2 %
2012	9,1	1,6	15 %
2013	9,5	2,1	18,1 %
2014	9,7	2,8	22,4 %
2015	9,7	3,6	27 %
2016	9,8	4,6	31,9 %
2017	9,9	6,1	38,1 %
2018	9,9	8	44,7 %
2019	10	10	50 %
2020	9,9	11,7	54,2 %
2021	10	13,8	58 %
2022	10,1	16,4	61,9 %
2023	10,2	19,8	66 %
2024	10,2	24,4	70,5 %
2025 (прогнозовані)	10,3	30,9	75 %

Споживчий сектор домінує в цій області з переважною часткою пристроїв IoT. Електроенергія, пальне, пара та змінний струм, а також виробництво складають значно менші (але відносно значні) частини галузі.

Також за останні роки зросла й кількість IoT платформ. У 2015 було відомо про 260 різних публічних IoT платформ. Вже у 2016 ця цифра зросла до 360 і до 460 за наступний після цього рік. До 2019 вже з'явилось 620 платформ, з них у 2021 залишались активними 426, навіть не зважаючи на глобальну кризу, серед них такі глобальні імена як Amazon та Microsoft.

Це ж відноситься і до витрат на технології IoT, які за останні роки тільки зростали (рис. 2).

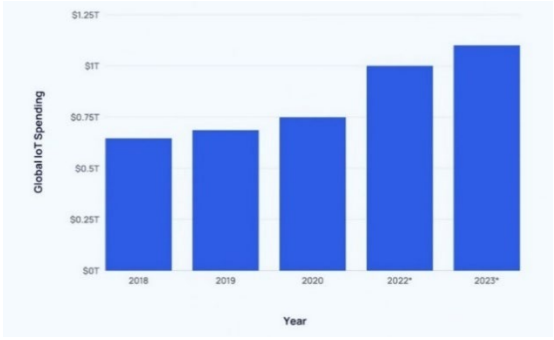


Рис. 2. Щорічні витрати сектору IoT [4]

Світові витрати IoT зросли щонайменше до $40 мільярдів з 2018 року щороку. В 2020 світ витратив близько $749 мільярдів, у 2022 вже $1 трильйон.

Цілком прогнозовано, що більша частина всього сектору IoT припадає на Азію, при цьому незважаючи на стрімкий ріст розмірів IoT, його розподіл залишається незмінним з року в рік (рис. 3).

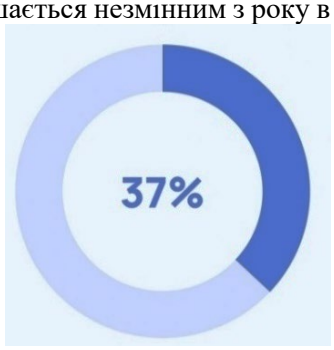


Рис. 3. У 2018 році 37% глобального IoT припало на Азію [4]

За розмірами наступним найбільшим регіоном є Північна Америка (29 %), ЄБСА (Європа, Близький Схід, Африка) (23 %). Японія (9 %) та Латинська Америка (9 %) складають решту ринку IoT.

Серед витрат IoT за категоріями першість займає категорія відео розваг. Ця категорія займає 64,6 % усього IoT, тобто більше аніж усі інші категорії разом взяті [4].

Подібна ситуація відноситься і до ринку Машинного навчання, який відчуває стабільне зростання протягом останніх років (рис. 4).

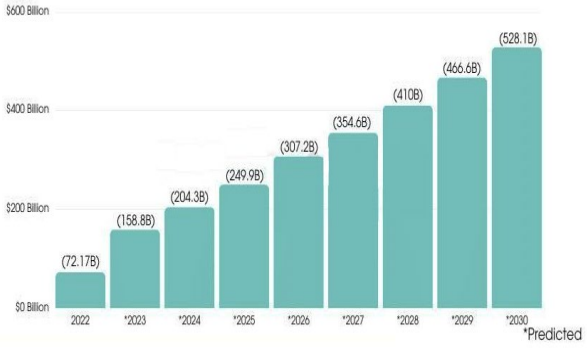


Рис. 4. Розмір ринку Машинного навчання [4]

Крім того планується зростання і до 2030 року, адже ці технології використовуються практично у всіх сферах життя [5].

Метою статті є розробка та дослідження адаптивних алгоритмів управління для IoT систем, що функціонують в умовах обмежених ресурсів. Основний акцент ставиться на дослідження існуючих підходів, які дозволять підвищити ефективність використання енергетичних, обчислювальних та мережевих ресурсів IoT пристроїв, забезпечуючи при цьому стабільну та надійну роботу систем в умовах змінних середовищ і динамічних потреб, а також на їх можливе вдосконалення. Стаття спрямована на впровадження рішень, які оптимізують роботу IoT пристроїв шляхом їх адаптації до обмежених ресурсів і змін в навколишньому середовищі, без втрати продуктивності та функціональності.

Підходи використання адаптивних алгоритмів управління технологіями IoT в умовах обмежених ресурсів

Приклади адаптивних алгоритмів управління технологією IoT в умовах обмежених ресурсів:

1. Q-LEARNING:

Q-learning - це алгоритм машинного навчання, який використовується для прийняття рішень в умовах невизначеності. У контексті IoT цей алгоритм може використовуватися для оптимізації ресурсів та прийняття рішень щодо енергозбереження, наприклад, управління енергоспоживанням пристроїв на основі динамічних змін у споживанні електроенергії [6].

2. FUZZY LOGIC (Нечітка логіка):

Нечітка логіка - це метод, який дозволяє моделювати нечіткі або невизначені величини та правила. У IoT вона може використовуватися для прийняття рішень в умовах обмежених ресурсів, коли точні значення можуть бути невизначені або нестабільні, наприклад, управління системою автономних роботів або маршрутизація даних у мережі IoT [7].

3. SWARM INTELLIGENCE (Інтелект рою):

Інтелект рою – це підхід, що моделює поведінку рою агентів для розв’язання складних завдань. У IoT цей метод може використовуватися для координації дій та оптимізації розподілу ресурсів між пристроями, що входять до мережі IoT [8].

Ці алгоритми надають можливість ефективного управління технологією IoT в умовах обмежених ресурсів, що дозволяє підтримувати високу продуктивність.

Q-learning: варіант безмодельного навчання

Q-навчання – це безмодельний алгоритм навчання з підкріпленням; метою *Q*-навчання є вивчення стратегій, які диктують, яку дію агент повинен виконати в будь-якій ситуації; *Q*-навчання не потребує моделі навколишнього середовища (звідси і кваліфікатор «безмодельний») має і може вирішувати проблеми зі стохастичними переходами та винагородами без необхідності адаптації.

Даний алгоритм використовує систему винагород та покарань для покращення майбутніх результатів. Винагородою, або ж покаранням є абстрактне число, яке показує ефективність рішення, прийнятого алгоритмом. Загальний варіант алгоритму виглядає як множина вже прийнятих рішень та нагород виданих агенту (1).

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) * Q(s_t, a_t) + \alpha * \\ * (r_t + \gamma * max^a Q(s_{t+1}, a)), \quad (1)$$

де: $Q^{new}(s_t, a_t)$ – значення Q-функції, яке поверне стан s_tпри виборі дії a_t; α – швидкість навчання; $Q(s_t, a_t)$ – старе значення Q-функції; r_t – винагорода; γ – «фактор знижки», тобто наскільки важливою буде майбутня нагорода; $max^a Q(s_{t+1}, a)$ – прогнозоване найкраще майбутнє значення Q-функції

Можна розбити процес прийняття рішень Q-learning на п'ять простих кроків:

1. Агент знаходиться в нульовому стані середовища.

2. Він виконуватиме дії на основі певної стратегії.

3. За цю дію він отримає винагороду або покарання.

4. Вивчаючи попередні дії та оптимізуючи стратегію.

5. Процес повторюватиметься, доки не буде знайдено оптимальну стратегію.

Набір значень Q в даному випадку називають Q-таблицею, саме на її основі алгоритм і приймає рішення.

Загалом є два методи отримання Q-значення:

- Темпоральна різниця – формула, що обчислює Q-значення шляхом включення значення поточного стану та дії шляхом порівняння відмінностей із попереднім станом та дією.
- Рівняння Белмана – Математик Річард Беллман винайшов це рівняння в 1957 році як рекурсивну формулу для прийняття оптимальних рішень. У контексті q-навчання рівняння Беллмана використовується для обчислення значення даного стану та оцінки його відносного положення. Стан із найвищим значенням вважається оптимальним [9].

На сьогодні алгоритм *Q-learning* знайшов чимало застосувань у повсякденному житті, особливо у місцях де точність або продуктивність не є критично важливими:

- новинні рекомендації;
- контроль мережевого трафіку;
- реклама в мережі Інтернет.

Переваги використання алгоритму *Q-learning*:

1. Безмодельність.

2. Оптимізація «поза політикою» (вважається що цей алгоритм можна оптимізувати в будь-якій площині без прив'язки до конкретної предметної області, що надає більше свободи для імплементації алгоритму).

3. Гнучкість.

4. Офлайн навчання (Модель Q-learning можна розгорнути на попередньо зібраних офлайн наборах даних).

Недоліки використання алгоритму *Q-learning*:

1. Компроміс навчання та використання – Для моделі *Q-learning* може бути важко знайти правильний баланс між пробуванням нових дій і дотриманням того, що вже відомо. Це дилема, яку зазвичай називають компромісом дослідження проти експлуатації для навчання з підкріпленням.

2. «Прокляття розмірності» – *Q-learning* потенційно може зіткнутися з ризиком машинного навчання, відомим як прокляття розмірності. Це проблема з даними великої розмірності, коли кількість даних, необхідних для представлення розподілу, зростає експоненціально. Це може призвести до обчислювальних проблем і зниження точності.

3. Переоцінка – алгоритм *Q-learning* іноді може бути занадто оптимістичним, переоцінюючи наскільки ефективними можуть бути рішення та їх результати.

4. Продуктивність – якщо є кілька способів вирішення проблеми, *Q-learning* може зайняти багато часу, щоб знайти оптимальне рішення.

Підходячи до питання використання алгоритму *Q-learning* в технологіях IoT, особливо в площині обмежених ресурсах потрібно завжди враховувати контекст в якому даний алгоритм планується застосовувати. Наприклад, чи необхідно застосувати його в «розумному» будинку або ж в системі моніторингу енергомережі (яку теж можна розглядати як частину технологій IoT). В даному випадку основними проблемами даного алгоритму будуть прокляття розмірності та переоцінка результатів, адже деякі області IoT вимагають не просто продуктивності чи гнучкості, а саме точності, адже неправильно імплементований алгоритм може не просто не допомогти, а й, навіть, нашкодити.

Проте все-таки даний алгоритм можна використати для даних задач:

1. Оптимізація стратегій управління: *Q-learning* може бути використаний для навчання агента приймати оптимальні рішення щодо управління технологією IoT, такі як вибір маршрутів передачі даних, розподіл ресурсів чи управління енергоспоживанням. Агент може навчитися приймати рішення, що максимізують продуктивність мережі та ефективність використання ресурсів.

2. Адаптивне управління: *Q-learning* дозволяє агенту навчитися адаптуватися до змінних умов роботи мережі IoT та ресурсів, що стає особливо важливим у динамічних середовищах. Наприклад, агент може автоматично адаптувати стратегії управління в залежності від змін споживання енергії чи доступності ресурсів.

3. Оптимізація з врахуванням часу: *Q-learning* може бути використаний для оптимізації управ-

ління технологією IoT з урахуванням часових обмежень. Агент може навчитися приймати рішення, що мінімізують часові затримки у передачі даних або максимізують час роботи системи.

4. Підтримка мультиагентних систем: *Q-learning* може бути використаний для навчання кількох агентів у мультиагентних системах IoT співпрацювати між собою та координувати свої дії для досягнення спільної мети.

5. Управління в умовах невизначеності: *Q-learning* може бути корисним у вирішенні проблеми управління технологією IoT в умовах невизначеності, коли точні параметри середовища можуть бути невідомими чи змінюватися з часом.

Нечітка логіка: нечіткі залежності

Нечітка логіка – це розділ математики, який є узагальненням класичної логіки та теорії множин. Він вивчає об'єкти, які мають функціональність елементів, що належать до наборів, які приймають значення в діапазоні [0, 1]. На основі цієї концепції вводяться логічні операції над нечіткими множинами та сформульовано поняття лінгвістичних змінних як нечітких множин.

Даний алгоритм є досить простим: його робота полягає в перетворенні цифрових даних у лінгвістичні (наприклад параметри у відрізках «високі», «середні», «низькі») (рис. 5) [10].

relative_humidity	temperature	heat_index
74.0	25.0	25.5

relative_humidity	temperature	heat_index
mid	mid	high

Рис. 5. Результат перетворення числових значень на лінгвістичні

Після перетворення даних даний алгоритм надає один з найпотужніших інструментів моделювання: правило ЯКЩО-ТО. Наприклад, ЯКЩО відносна вологість є середньою то температура є середньою ТО індекс тепла є високим. Проаналізувавши необхідні правила, які алгоритм складе він може виконувати необхідні дії.

Нечітка логіка має багато сфер застосування, насправді в будь-якій сфері можна застосувати алгоритм нечіткої логіки:

- Аерокосмічна галузь.
- Контроль швидкості транспортних засобів.
- Системи підтримки прийняття рішень.
- Хімічна індустрія.

Переваги використання нечіткої логіки:

1. Моделювання нечітких залежностей: Алгоритм нечіткої логіки дозволяє моделювати нечіткі, невизначені або розмиті залежності між вхідними та вихідними змінними, що дозволяє більш точно враховувати реальні умови та вимоги.

2. Толерантність до шуму та непередбачуваності: Алгоритм нечіткої логіки дозволяє ефективно працювати з даними, які містять шум або непередбачувані зміни, оскільки він дозволяє враховувати неоднорідність даних та враховувати їх варіативність.

3. Простота використання та розуміння: Алгоритм нечіткої логіки досить простий у використанні та розумінні, що робить його доступним для використання без спеціальних знань у галузі математики чи логіки.

4. Здатність до використання експертних знань: Алгоритм нечіткої логіки дозволяє використовувати експертні знання та досвід у формі лінгвістичних правил для прийняття рішень, що дозволяє враховувати інтуїтивні аспекти в процесі прийняття рішень.

Недоліки використання нечіткої логіки:

1. Неоднорідність та нечіткість результатів: Використання алгоритму нечіткої логіки може призвести до неоднорідності та нечіткості результатів, оскільки він враховує непередбачуваність даних та може призвести до неоднозначних висновків.

2. Складність налаштування параметрів: Деякі алгоритми нечіткої логіки можуть мати велику кількість параметрів, які потребують налаштування, що може вимагати додаткового часу та експертної експертизи.

3. Висока обчислювальна складність: Деякі алгоритми нечіткої логіки можуть бути обчислювально витратними, особливо при роботі з великими обсягами даних, що може призвести до затримок у роботі системи.

4. Потреба в додатковому навчанні: Використання алгоритму нечіткої логіки може вимагати додаткового часу та зусиль для навчання персоналу та розуміння принципів роботи алгоритму.

Хоча алгоритми нечіткої логіки і є відносно простим концептом, вони також є і досить ефективним рішенням, особливо якщо точність не є важливим критерієм.

***Swarm intelligence*: колективний інтелект**

Колективні знання – це термін, який використовується для опису складної колективної поведінки децентралізованих систем, що самоорганізуються. З точки зору комп'ютерних наук, колективний інтелект є предметом досліджень у галузі розробки та вивчення ефективних чисельних методів розв'язання задач, подібних до поведінки біологічних «груп». Результатами в цій галузі є власне розроблені алгоритми, які в основному використовуються для розв'язання задач комбінаторної оптимізації та задач комівояжера [11].

Концепція алгоритмів колективного інтелекту ідеально підходить для управління технологіями IoT, адже кожна маленька частина такого алгоритму може знаходитися в, або керувати кожним окремим пристроєм. Звичайно, є багато видів алгоритмів колективного алгоритму, такі як штучний алгоритм бджолиної сім'ї (ABC), штучна імунна система, алгоритм зозулі та інші.

Основним застосуванням алгоритмів колективного інтелекту є, звичайно, задачі оптимізації, будь це кластеризація даних (що у випадку обмежених ресурсів можна розглянути як нестачу місця для зберігання та обробки інформації), мінімізація функції та інші. Крім того, дані алгоритми мають властивості класичних алгоритмів з «нагородами», а тому можуть робити все що і останні (симуляція, прогнозне моделювання).

Для конкретики можна розглянути алгоритм Штучної імунної системи. Даний алгоритм моделює роботу природної імунної системи людини з метою вирішення обчислювальних завдань, таких як класифікація даних, виявлення аномалій, оптимізація тощо. Основна ідея полягає в тому, що штучна імунна система використовує принципи та механізми природної імунної системи для вирішення обчислювальних завдань. Якщо бути точним, то у випадку керування технологіями IoT в умовах обмежених ресурсів задачу можна поставити наступним чином: необхідно класифікувати які саме ресурси обмежені, або ж виявити аномалії в системах IoT. Працює даний алгоритм наступним чином:

1. Ініціалізація: Спочатку система ініціалізує початкову популяцію антитіл.

2. Антигени: Антигени представляють собою дані, які потрібно класифікувати. Наприклад, це можуть бути зображення для розпізнавання об'єктів або тексти для визначення категорії.

3. Взаємодія антитіла з антигенами: Антитіла в популяції взаємодіють з антигенами, пробуючи класифікувати їх. Це може включати порівняння антигенів з певними критеріями, які визначають класи або категорії.

4. Оцінка ефективності антитіл: Після взаємодії кожного антитіла з антигенами оцінюється його ефективність за допомогою функції оцінки (фітнес-функції). Антитіла, які виявилися найбільш ефективними у класифікації антигенів, зберігаються або переходять до наступного покоління.

5. Мутація та селекція: Для збереження різноманітності та пошуку оптимальних рішень в популяції відбувається мутація антитіл, а також відбираються найбільш ефективні антитіла для наступного покоління.

6. Збільшення ефективності: Процес мутації та селекції повторюється протягом кількох поколінь, поки не буде досягнуто певного критерію зупинки або не буде досягнута задовільна ефективність класифікації.

Як видно, даний алгоритм можна розглянути як змішування алгоритмів навчання з підкріпленням та еволюційних алгоритмів. Усі ці процеси можна імплементувати в різні вузли систем IoT, наприклад, антигени – це дані з різних датчиків (це можуть бути дані про стан електромережі, або ж дані про стан пристроїв накопичення інформації), а антитіла та обчислювальні потужності самого алгоритму можуть або знаходитись на центральному обчислювальному пристрої, або ж розподілені між усіма пристроями IoT мережі (хоча в даному випадку потужності алгоритму будуть обмежені найменш потужним пристроєм меережі) (рис. 6).

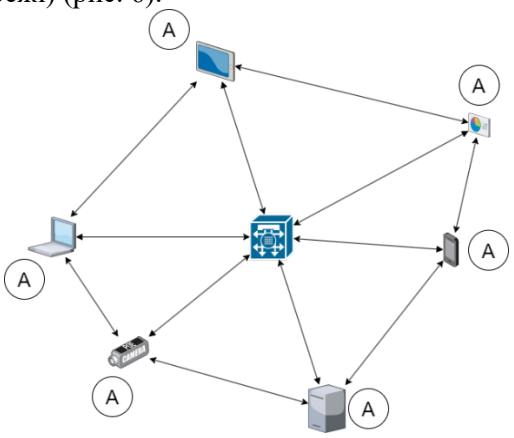


Рис. 6. Приклад IoT мережі, де A – це антигени, в центрі основні обчислювальні потужності для алгоритму

Даний варіант є найкращим варіантом, адже алгоритм зможе бачити і контролювати усі аспекти мережі, необхідні для ефективної роботи мережі, при цьому не бути обмеженим в продуктивності найменш потужним пристроєм мережі [12].

Переваги використання алгоритмів колективного розуму:

- Адаптивність: AIC може адаптуватися до змін у середовищі і пристосовуватися до нових умов без необхідності повного перепрограмування;
- Ефективність у вирішенні складних задач: Цей алгоритм може бути досить ефективним у вирішенні складних задач оптимізації, класифікації або виявлення аномалій;
- Відсутність потреби в знанні: AIC може працювати з наборами даних без попередніх знань про їх структуру або особливості;
- Робота з неконкретними проблемами: Цей алгоритм може застосовуватися для вирішення

проблем, які важко формалізувати або визначити чітко.

Недоліки використання алгоритмів колективного розуму:

- Час виконання: AIC може вимагати значних обчислювальних ресурсів через ітеративний процес мутації, селекції та оцінки;
- Схильність до «застрягання» в локальних мінімумах: Як і багато інших еволюційних алгоритмів, AIC може «застрягати» в локальних мінімумах, не досягнувши глобального оптимуму;
- Складність настройки параметрів: Для досягнення оптимальних результатів може знадобитися тонке налаштування параметрів алгоритму, що може бути часом і ресурсозатратним;
- Неявність пояснювальної здатності: У багатьох випадках складно інтерпретувати, як саме працює алгоритм і чому прийняті рішення, що робить його менш зрозумілим для людей.

Оптимальний алгоритм

Алгоритм нечіткої логіки, як і алгоритм колективного інтелекту є досить непоганими рішеннями, проте дещо обмеженими та неефективними у деяких ситуаціях. Саме тому є сенс спробувати «схрестити» ці два алгоритми.

Об'єднання алгоритмів можна провести за двома шляхами:

1. Використання нечітких правил *Fuzzy logic* для визначення параметрів агентів у системі *Swarm Intelligence*.

2. Використання вихідних даних алгоритму *Swarm Intelligence* для підтримки прийняття рішень на основі алгоритму *Fuzzy logic*.

Хоча обидва варіанти є прийнятними, використати є сенс перший, адже було встановлено що саме алгоритми колективного інтелекту є дещо більш гнучкими, хоча і поступаються в ефективності алгоритму нечіткої логіки, саме тому ми будемо підтримувати один алгоритм за допомогою другого.

Загальний алгоритм буде виглядати наступним чином:

1. Розробка нечітких правил: Спочатку потрібно визначити нечіткі правила, які будуть використовуватися для визначення параметрів агентів у системі *Swarm Intelligence*. Ці правила можуть бути базовані на експертних знаннях або визначені на основі аналізу даних.

2. Визначення вхідних та вихідних змінних: Встановлюються, які вхідні змінні впливають на параметри агентів, а також які вихідні параметри будуть отримані в результаті.

3. Побудова нечіткої логіки: Розробка нечітких правил та механізмів вирішення конфліктів для кожної вхідної змінної.

4. Використання нечітких правил у системі *Swarm Intelligence*: Імплементація нечітких правил в систему *Swarm Intelligence*. Це може включати визначення поведінки агентів на основі вхідних даних та встановлення відповідних параметрів з урахуванням цих правил.

5. Тестування та налаштування: Проведення тестування системи з використанням різних наборів вхідних даних, щоб переконатися, що вона працює ефективно.

6. Оцінка результатів: Це допоможе визначити ефективність та працездатність рішення.

Даний алгоритм можна розбити на дві частини: визначення параметрів агентів та прийняття рішень. Першим займається алгоритм *Fuzzy logic*, він визначає рівень членства для кожного агента на основі вхідних параметрів. Цей рівень членства вказує, наскільки сильно агент відноситься до певного класу (низький, середній, високий) в залежності від цих параметрів. Прийняттям рішень займається алгоритм swarm intelligence і так як ми плануємо використовувати даний алгоритм на розлогій мережі різноманітних пристроїв візьмемо конкретний механізм обміну даних. Скомбінувавши це все ми і отримаємо новий алгоритм який можна назвати «*Fuzzy Swarm Intelligence*», або «Нечіткий колективний розум».

Перш за все необхідно створити Агента, який буде представляти з себе по суті пристрої IoT мережі (рис. 7).

Коли Агента було розроблено необхідно придумати правила для алгоритму Нечіткої логіки. Для простоти показу візьмемо приклад з обмеженим дисковим простором. В такому прикладу всі Агенти повинні «спілкуватися» між собою, передаючи інформацію про свій дисковий простір і у разі його нестачі тимчасово переймати на себе обов'язки зі зберігання даних Агента, що запитує про це. Проте це вже прийняття рішень, яке повинно бути у частині колективного інтелекту. *Fuzzy Logic* просто обчислює рівень вільної пам'яті Агентів та як швидко він заповнюється (рис. 8).

```
9 references
public class Agent
{
    public double _membershipDegree;
    4 references
    public double MembershipDegree
    {
        get { return _membershipDegree; }
        set
        {
            _membershipDegree = Math.Max(0, Math.Min(1, value));
        }
    }

    4 references
    public string GetMembershipLevelDescription()
    {
        MembershipLevel level = GetMembershipLevel();
        return level.ToString();
    }

    1 reference
    private MembershipLevel GetMembershipLevel()
    {
        if (MembershipDegree >= 0.8)
        {
            return MembershipLevel.High;
        }
        else if (MembershipDegree < 0.8 && MembershipDegree >= 0.5)
        {
            return MembershipLevel.Medium;
        }
        else
        {
            return MembershipLevel.Low;
        }
    }

    4 references
    public double ChangeRate { get; set; }
    3 references
    public double FreeSpacePercentage { get; set; }

    // Додамо поле для зберігання індексу агента для спрощення ідентифікації
    7 references
    public int Index { get; set; }
}
```

Рис. 7. Приклад Агента для алгоритму що керує дисковим простором

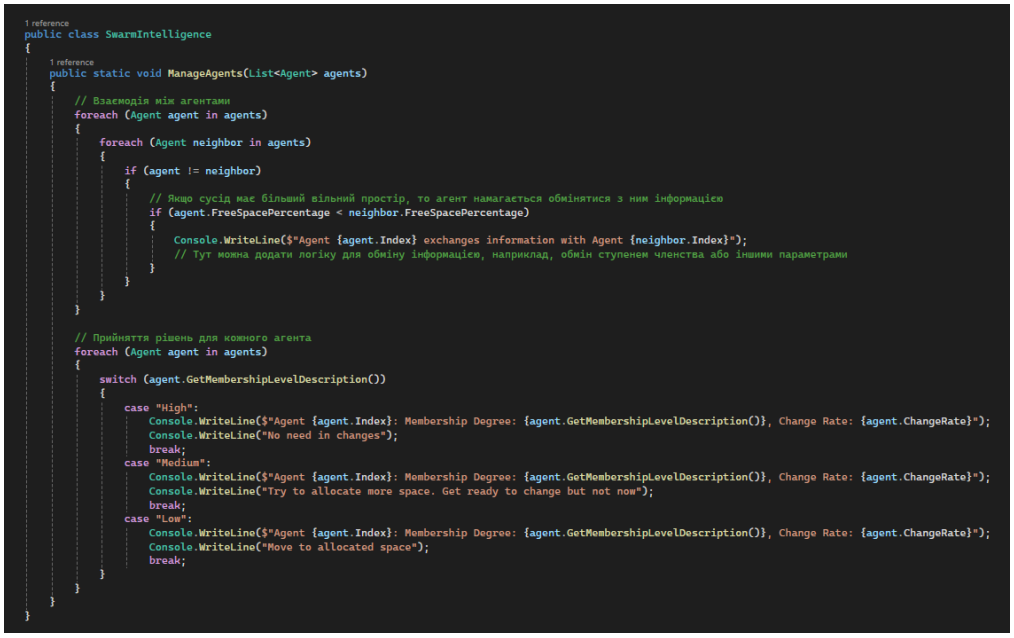
```
2 references
public class FuzzyLogic
{
    2 references
    public static void DetermineParameters(Agent agent, double freeSpacePercentage)
    {
        agent.MembershipDegree = CalculateMembershipDegree(freeSpacePercentage);
        agent.ChangeRate = CalculateChangeRate();
        agent.FreeSpacePercentage = freeSpacePercentage; // Зберігаємо вільний простір для обміну з іншими агентами
    }

    1 reference
    private static double CalculateMembershipDegree(double freeSpacePercentage)
    {
        if (freeSpacePercentage >= 80)
        {
            return 1.0;
        }
        else if (freeSpacePercentage < 80 && freeSpacePercentage >= 50)
        {
            return (80 - freeSpacePercentage) / 30;
        }
        else
        {
            return 0.0;
        }
    }

    1 reference
    private static double CalculateChangeRate()
    {
        return 0.1;
    }
}
```

Рис. 8. Нечітка логіка, яка визначає параметри Агентів

Після того як усі Агенти було створено, а їх параметри визначено, наступає черга Колективного інтелекту. Вся ідея в тому щоб будь-які агенти могли б вільно обмінюватися своєю інформацією з іншими агентами, таким чином прийняття рішень буде відбуватися простіше і швидше (рис. 9).



```
1 reference
public class SwarmIntelligence
{
    1 reference
    public static void ManageAgents(List<Agent> agents)
    {
        // Взаємодія між агентами
        foreach (Agent agent in agents)
        {
            foreach (Agent neighbor in agents)
            {
                if (agent != neighbor)
                {
                    // Якщо сусід має більший вільний простір, то агент намагається обмінятися з ним інформацією
                    if (agent.FreeSpacePercentage < neighbor.FreeSpacePercentage)
                    {
                        Console.WriteLine($"Agent {agent.Index} exchanges information with Agent {neighbor.Index}");
                        // Тут можна додати логіку для обміну інформацією, наприклад, обмін ступенем членства або іншими параметрами
                    }
                }
            }
        }

        // Прийняття рішень для кожного агента
        foreach (Agent agent in agents)
        {
            switch (agent.GetMembershipLevelDescription())
            {
                case "High":
                    Console.WriteLine($"Agent {agent.Index}: Membership Degree: {agent.GetMembershipLevelDescription()}, Change Rate: {agent.ChangeRate}");
                    Console.WriteLine("No need in changes");
                    break;
                case "Medium":
                    Console.WriteLine($"Agent {agent.Index}: Membership Degree: {agent.GetMembershipLevelDescription()}, Change Rate: {agent.ChangeRate}");
                    Console.WriteLine("Try to allocate more space. Get ready to change but not now");
                    break;
                case "Low":
                    Console.WriteLine($"Agent {agent.Index}: Membership Degree: {agent.GetMembershipLevelDescription()}, Change Rate: {agent.ChangeRate}");
                    Console.WriteLine("Move to allocated space");
                    break;
            }
        }
    }
}
```

Рис. 9. Колективний інтелект, який приймає відповідні рішення

Представлений приклад є дуже простим, базовим. Його можна розширяти і видозмінювати. Додати механізми які дозволяють «ділитися» пам'яттю, надати колективному інтелекту можливість перенаправляти живлення з непотрібних, або ж неактивних пристроїв на пристрої яким ця енергія необхідна. Можна додати можливість читання даних з логів, що дозволить моніторити роботу алгоритму.

Такий алгоритм фактично залишився б без не достатків двох використаних. Він комбінує ефективність колективного інтелекту з гнучкістю нечіткої логіки. Окрім того, залежно від задач можна використати різні підходи колективного інтелекту, наприклад «*Ant colony*», або «Оптимізацію рою часток (PSO)». Показаний приклад – лише «скелет», який демонструє базовий функціонал алгоритму, який можна модифікувати залежно від IoT мережі в якій його буде розгорнуто, наприклад, якщо ми розглядаємо мережу пристроїв контролю дорожнього трафіку, то можна використати підхід «PSO», якщо розглянемо «розумний будинок», то краще використати «*Ant colony*».

Як видно, є багато варіантів розробки та застосування даного алгоритму, єдине питання: де конкретно його буде використано?

Порівняльні результати створеного алгоритму з алгоритмом *Fuzzy logic*

Для визначення результатів, тобто ефективності нового алгоритму можна симулювати деяке IoT-середовище. Скажімо, у нас є мережа з 50 IoT пристроїв, кожен з яких може мати обмежені ресурси – енергія, обчислювальні потужності та комунікаційні мережі. Ці пристрої повинні слідкувати та керувати різними аспектами інфраструктури «розумного міста», будь то вологість, температура чи якість повітря, або потік транспорту.

Для цього ми будемо слідкувати за рядом характеристик, таких як: мінімізація споживання енергії, зменшення затримки сигналу, максимізація загальної якості обслуговування (*Quality of Service, QoS*). Порівняємо створений алгоритм зі звичайним алгоритмом Нечіткої логіки.

Якщо взяти мережу то можна вважати, що кожен вузол(пристрій) мережі – власна сутність зі своїми характеристиками. Симуляція пройде 1000 кроків, кожен крок представляє реальну операцію довжиною в одну хвилину.

Проте деякі параметри необхідно нормалізувати, наприклад:

- споживання енергії – по шкалі від 0 до 1, де 0 – найкраще споживання енергії, а 1 – найгірше споживання енергії. Даний показник нормалізуємо діленням знайденого споживання на показник максимально можливого споживання;
- затримка сигналу – логіка така само: 0 – найменша затримка, 1 – найбільша затримка. І, враховуючи параметри нашої системи ми отримаємо дане значення поділивши знайдену затримку на максимальну (1000 мс);
- QoS – даний показник є найсуб'єктивнішим, де 0 – найгірший сервіс, а 1 – найкращий показник.

Ось результати які вийшло симулювати:

- Алгоритм *Fuzzy Logic*:

Енергоеефективність: 0,65
Затримка: 0,25
QoS: 0,82

Середнє використання ресурсів: 0,57

- Створений алгоритм:

Енергоефективність: 0,45
Затримка: 0,18
QoS: 0,92
Середнє використання ресурсів: 0,42 (рис. 10).

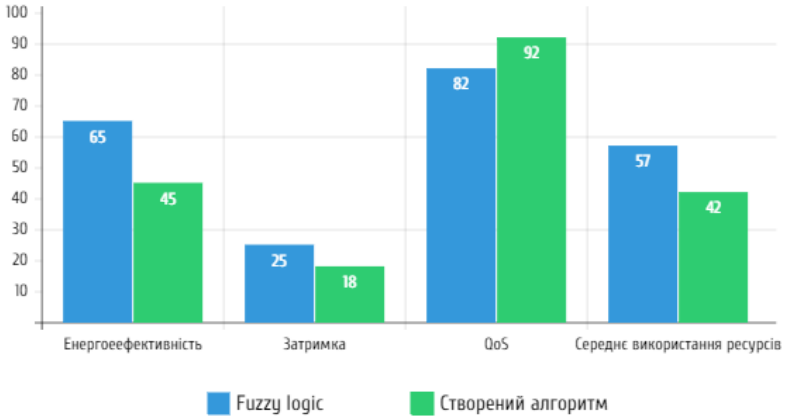


Рис. 10. Порівняння ефективності роботи алгоритмів

Інший варіант оцінки ефективності алгоритму: *F1-score*. Даний показник обчислюється через влучність та повноту тесту. Припустимо, що всі пристрої в мережі формують потоки даних різної важливості, таким чином ми зможемо розділити їх на 3 типи: критичні(наприклад, термінові повідомлення), важливі(запити на ремонт) та неважливі(періодичні оновлення станів).

Симулюючи створення 1000 потоків (300 критичних, 400 важливих та 300 неважливих), можна отримати наступні *F1-score*:

- Алгоритм *Fuzzy Logic*:

F1 score (Критичне): 0.85
F1 score (Важливе): 0.78
F1 score (Неважливе): 0.92
F1 score (Зважене середнє): 0.83

- Створений алгоритм:

F1 score (Критичне): 0.92
F1 score (Важливе): 0.85
F1 score (Неважливе): 0.95
F1 score (Зважене середнє): 0.90 (рис. 11).

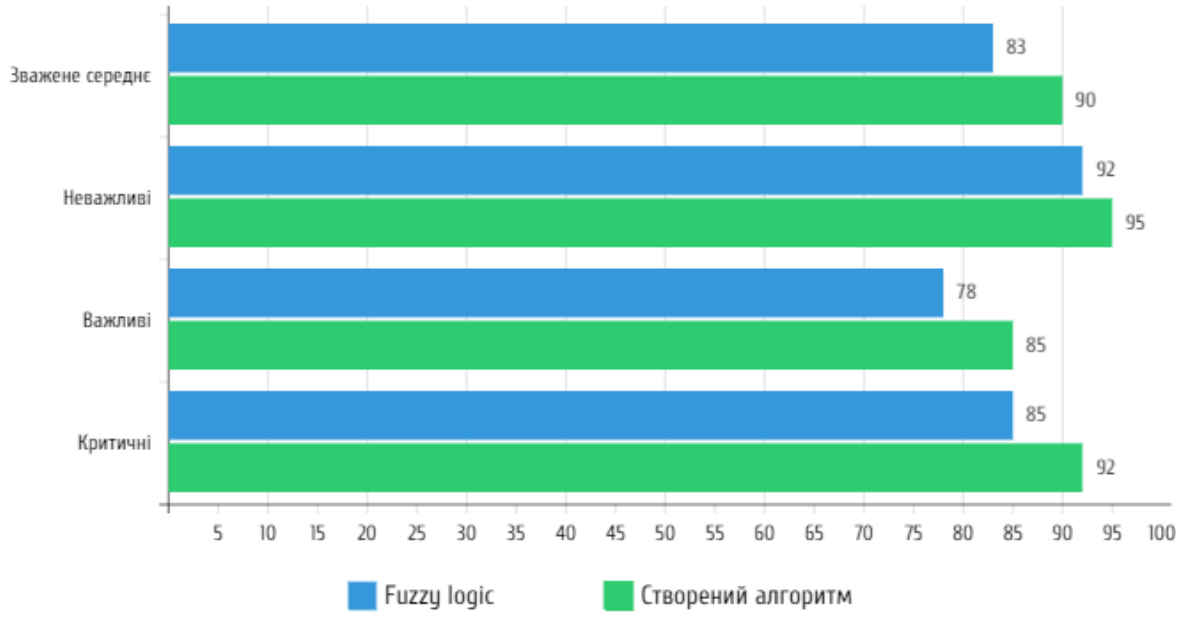


Рис. 11. Порівняння F1-score алгоритмів

Такі дані є лише поверхневим аналізом алгоритмів, вони будуть відрізнятися, залежно від ситуації та мережі, де алгоритм буде розгорнуто. Проте, навіть симулюючи роботу цих алгоритмів можна зрозуміти наступне: навіть враховуючи відносну ефективність алгоритму *Fuzzy logic*, створений алгоритм є на 10–25 % ефективнішим в керуванні мережами IoT пристроїв в умовах обмежених ресурсів.

Аналіз отриманих результатів

Досліджені алгоритми, хоч і всі мають свої переваги, та не всі можна ефективно використати для вирішення проблем керування IoT в умовах обмежених ресурсів.

Найкраще підходить (якщо розглядати мережу IoT як розподілену архітектуру) алгоритм колективного інтелекту, наприклад розглянутий алгоритм штучної імунної системи. Якщо розглядати мережу IoT як неподільну одиницю всередині якої циркулюють дані, найкраще буде підходити алгоритм нечіткої логіки. І хоча алгоритм *Q-learning* може підійти до обох ситуацій, він буде обмеженим і, потенційно, неефективним, адже його нові рішення базуються на оцінці успішних та, що важливо, провальних попередніх рішень.

Алгоритм колективного інтелекту, зберігаючи прийнятну продуктивність та точність прийнятих рішень, є дуже гнучким: його можна використати для одиничної системи, де кожна окрема частина алгоритму хоч і буде виконуватися на одній машині, буде відповідати за різні ділянки мережі, другий варіант – розділити алгоритм на частини та «запустити» його в різні пристрої IoT мережі. Так, якщо взяти другий варіант продуктивність подібного алгоритму буде визначено найменш потужним пристроєм в мережі, проте він буде мультизадачним і головна частина алгоритму буде зайнята тільки обробкою даних та створенням нових рішень, перекладаючи задачі моніторингу ресурсів на інші частини алгоритму.

Плюс нечіткої логіки – її здатність до прийняття рішень, коли вхідні дані не є чіткими, а також можливість інтерпретації експертних рішень, що по суті дозволяє використовувати в алгоритмі базу знань та, потенційно, експертні системи.

Q-learning хоча і є теж потенційно ефективним рішенням, все-таки в даному аспекті не є оптимальним. Потрібно зауважити, що його можна вважати таким тільки в аспекті обмеженості ресурсів. При недостатній кількості ресурсів нам потрібно прийняти не стільки швидке, скільки ефективне рішення, а тому алгоритми які можуть прийняти неефективне рішення не підходять для керування такими технологіями. Хоча і тут є «але»: це застосовується тільки для ненавчених алгоритмів, або для алгоритмів які покладаються на множину попередніх рішень для прийняття наступних.

Отже, у результаті дослідження виявлено, що найефективнішим алгоритмом для керування технологіями IoT в умовах обмежених ресурсів є алгоритм колективного інтелекту, який надає достатній рівень гнучкості, продуктивності, точності прийняття рішень та ефективності. Проте через проблеми його масштабування та використання пропонується покращити його за допомогою введення обробки інформації шляхом використання *Fuzzy logic*, що дозволить покращити роботу даного алгоритму в середньому на 10–25 % (рис. 12).

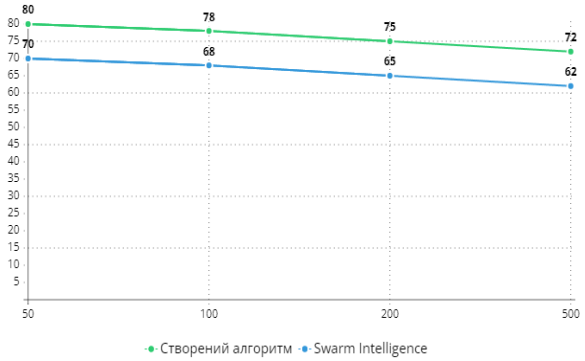


Рис. 12. Порівняння F1-score створеного алгоритму та класичного алгоритму *Swarm intelligence* (*Ant colony*)

Переваги використання створеного алгоритму:

- Адаптивність до змінних умов:

Нечітка логіка дозволяє алгоритму адаптуватися до різних умов роботи мережі, таких як зміна навантаження, коливання якості з'єднання, енергоспоживання та інші непередбачувані фактори.

- Гнучкість у налаштуванні:

Нечіткі правила дозволяють створювати алгоритми, які враховують широкий спектр вхідних параметрів та налаштовують поведінку агентів більш гнучко порівняно з жорсткими правилами класичних алгоритмів.

- Покращена продуктивність:

Використання нечітких правил може покращити точність і повноту класифікації або ухвалення рішень в IoT мережі. Це означає, що алгоритм з *Fuzzy Logic* має кращу збалансованість між точністю (*precision*) і повнотою (*recall*).

- Стійкість до шуму і неточностей:

Нечітка логіка добре працює в умовах невизначеності та шуму, що часто характерно для реальних IoT середовищ, де дані можуть бути неточними або неповними.

- Покращене використання ресурсів:

Алгоритми з нечіткою логікою можуть ефективніше використовувати обмежені ресурси IoT пристроїв (батарею, обчислювальні потужності, пам'ять), оптимізуючи енергоспоживання та збільшуючи тривалість роботи мережі.

- Легкість інтеграції з іншими системами:

Нечітка логіка може бути легко інтегрована з іншими методами штучного інтелекту та машинного навчання, що дозволяє створювати гібридні системи для більш складних завдань.

- Вища стабільність при збільшенні масштабів мережі:

Алгоритм з нечіткою логікою демонструє менше зниження продуктивності (*F1-score*) при збільшенні кількості пристроїв у мережі, що вказує на його кращу масштабованість.

Висновки

У статті розглянуто три різних адаптивні алгоритми, які надають різний функціонал для вирішення проблеми керування технологіями IoT в умовах обмежених ресурсів. Вирішення такої задачі є затребуваним, адже системи IoT стають все більш розповсюдженими, так як все більше вже існуючих систем підпадають під визначення Інтернету речей. Впровадження адаптивних алгоритмів може підвищити ефективність технологій IoT та автоматизувати багато процесів.

У дослідженні показано, що найефективнішим алгоритмом для керування технологіями IoT в умовах обмежених ресурсів є алгоритм колективного інтелекту, який надає достатній рівень гнучкості, продуктивності, точності прийняття рішень та ефективності. Запропоновано покращити його за допомогою введення обробки інформації шляхом використання *Fuzzy logic*, що дозволить покращити роботу даного алгоритму в середньому на 10–25 %.

ЛІТЕРАТУРА

[1] Wikipedia. Internet of things. URL: https://en.wikipedia.org/wiki/Internet_of_things (дата звернення: 29.04.2024).

[2] IEEE Xplore. The research and implement of smart home system based on Internet of Things. URL: https://ieeexplore.ieee.org/document/6066672 (дата звернення: 29.04.2024)

[3] Big Data Statistics 2023: How Much Data is in The World? URL: https://firstsiteguide.com/big-data-stats/ (дата звернення: 29.04.2024)

[4] Exploding Topics. 80+ Amazing IoT Statistics. URL: https://explodingtopics.com/blog/iot-stats#iot-industry-size (дата звернення: 07.05.2024)

[5] What's the big data. Top Machine Learning Statistics to know. URL: https://whatsthebigdata.com/top-machine-learning-statistics/ (дата звернення: 07.05.2024)

[6] Watkins, C.J.C.H. Learning from Delayed Rewards. PhD thesis. 1989. P. 220–228.

[7] Fuzzy logic. URL: http://www.scholarpedia.org/article/Fuzzy_logic (дата звернення: 29.04.2024)

[8] Hassanien E. Swarm Intelligence for Resource Management in Internet of Things. 2020. P. 1–19.

[9] Wikipedia. Q-навчання. URL: https://uk.wikipedia.org/wiki/Q-%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F (дата звернення: 07.05.2024)

[10] Medium. Machine Learning with Fuzzy Logic. URL: https://towardsdatascience.com/machine-learning-with-fuzzy-logic-52c85b46bfe4 (дата звернення: 07.05.2024)

[11] Wikipedia. Колективний інтелект. URL: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82 (дата звернення: 09.05.2024)

[12] LinkedIn. How does Swarm Intelligence work and what are its potential applications? URL: https://www.linkedin.com/pulse/how-does-swarm-intelligence-work-what-its-potential-giovanni-sisinna (дата звернення: 09.05.2024)

Холявкіна Т. В., Троцький Я. В., Моденов Ю. Б.

АДАПТИВНІ АЛГОРИТМИ УПРАВЛІННЯ ТЕХНОЛОГІЇ IOT В УМОВАХ ОБМЕЖЕНИХ РЕСУРСІВ

Стаття присвячена дослідженню та аналізу застосування адаптивних алгоритмів управління технологією IoT в умовах обмежених ресурсів. В контексті швидко зростаючої популярності IoT, ефективне управління цією технологією в умовах обмежених обчислювальних, енергетичних та мережевих ресурсів стає вирішальним завданням. У роботі пропонуються методи адаптивного управління, спрямовані на оптимізацію використання ресурсів та підвищення продуктивності системи IoT.

Проведено аналіз трьох адаптивних алгоритмів управління IoT, а саме: Q- learning, алгоритм нечіткої логіки та алгоритм колективного інтелекту. Досліджуються особливості, ефективність та придатність цих алгоритмів для застосування в умовах обмежених ресурсів, таких як обчислювальна потужність та енергоефективність.

Як вирішення існуючої проблеми в статті запропоновано новий алгоритм: гібрид алгоритмів нечіткої логіки та колективного інтелекту. Новий алгоритм створює нові можливості для керування IoT-пристроями в умовах обмежених ресурсів, адже має кращі сторони обох алгоритмів, не маючи їх явних недоліків. Запропонований алгоритм має потенціал підвищення ефективності IoT-систем в середньому на 10–25 %.

Проте для досягнення кращих результатів, важливо також враховувати різні сценарії використання IoT-систем, оскільки ефективність алгоритмів може змінюватися залежно від конкретних умов. Наприклад, пристрої, що працюють у віддалених або важкодоступних місцях, вимагають більшої автономності та стійкості до обмежень мережевих ресурсів. Подібні фактори відіграють ключову роль у забезпеченні стабільної та надійної роботи IoT-систем у реальних умовах.

Висновки статті спрямовані на визначення найбільш ефективного алгоритму управління IoT в умовах обмежених ресурсів. Результати досліджень можуть бути корисними для розробників та впроваджувачів IoT систем, спрямованих на оптимізацію використання ресурсів та підвищення продуктивності систем.

Ключові слова: адаптивні алгоритми, інтернет речей, IoT, обмежені ресурси, управління, ефективність.

Kholyavkina T., Trotskyi Y., Modenov U.

ADAPTIVE ALGORITHMS FOR IOT TECHNOLOGY MANAGEMENT IN RESOURCE-CONSTRAINED ENVIRONMENTS

The article is dedicated to the research and analysis of the application of adaptive algorithms in managing IoT technology under resource-constrained conditions. In the context of the rapidly growing popularity of IoT, effective management of this technology in conditions of limited computational, energy, and network resources becomes a crucial task. Article proposes adaptive management methods aimed at optimizing resource utilization and improving the performance of IoT systems.

An analysis of three adaptive IoT management algorithms, namely Q-learning, fuzzy logic algorithm, and swarm intelligence algorithm, is presented. The features, efficiency, and suitability of these algorithms for use under resource-constrained conditions, such as computational power and energy efficiency, are examined.

As a solution to the existing problem, the article proposes a new algorithm: a hybrid of fuzzy logic and swarm intelligence algorithms. This new algorithm opens up new possibilities for managing IoT devices under limited resource conditions, as it combines the best features of both algorithms while avoiding their obvious drawbacks. The proposed algorithm has the potential to increase the efficiency of IoT systems by an average of 10–25 %.

However, to achieve better results, it is also important to consider various IoT use cases, as the effectiveness of the algorithms may vary depending on specific conditions. For example, devices operating in remote or hard-to-reach locations require greater autonomy and resilience to network resource limitations. Such factors play a crucial role in ensuring stable and reliable IoT system operation in real-world conditions.

The conclusions of the article aim to determine the most effective IoT management algorithm under resource-constrained conditions. The research results may be useful for developers and implementers of IoT systems focused on optimizing resource usage and improving system performance.

Keywords: adaptive algorithms, internet of things, IoT, resource constraints, management, efficiency.

Стаття надійшла до редакції 22.05.2024 р.
Прийнято до друку 11.09.2024 р.