МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота бакалавра

ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ БІОМЕТРИЧНИХ ДАНИХ ДЛЯ РОЗУМНИХ СПОРТИВНИХ ТРЕНУВАНЬ

Здобувач вищої освіти гр. ІН – 81 Артем ОВЧИННІКОВ

Науковий керівник,
кандидат технічних наук, доцент,
доцент кафедри комп'ютерних наук Ігор ШЕЛЕХОВ

Завідувач кафедри
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую
Зав. кафедри Анатолій ДОВБИШ
"_____" _____2022 р.

ЗАВДАННЯ

до кваліфікаційної роботи

здобувача вищої освіти четвертого курсу, групи ІН-81 спеціальності «122 – Комп'ютерні науки» денної форми навчання Овчиннікова Артема Ігоровича.

Тема: «ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ БІОМЕТРИЧНИХ ДАНИХ ДЛЯ РОЗУМНИХ СПОРТИВНИХ ТРЕНУВАНЬ»

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки:) інформаційний огляд; 2) постановка завдання; 3) визначення методів рішення завдання; 4) проектування інформаційної системи; 4) програмна реалізація; 5) аналіз результатів роботи.

Дата видачі завдання «_____»_____ 2022 р.
Керівник роботи _____ Ігор ШЕЛЕХОВ
Завдання прийняла до виконання _____ Артем ОВЧИННІКОВ

РЕФЕРАТ

Записка: 50 стор., 7 рис., 5 табл., 1 додаток, 41 джерел.

Об’єкт дослідження — процес інтелектуального аналізу біометричних даних.

Мета роботи — розробка системи інтелектуального аналізу біометричних даних для розумних спортивних тренувань.

Методи дослідження — методи інтелектуального аналізу даних, інформаційно-екстремальна інтелектуальна технологія.

Результати — розроблено та програмно реалізовано систему інтелектуального аналізу біометричних даних для розумних спортивних тренувань. В розробці застосовано математичні моделі і методи інформаційно-екстремальної інтелектуальної технології. Програмна реалізація виконана мовою програмування C#.

МЕТОД ФУНКЦІОНАЛЬНО-СТАТИСТИЧНИХ ВИПРОБУВАНЬ, КРИТЕРІЙ ФУНКЦІОНАЛЬНОЇ ЕФЕКТИВНОСТІ, ГІПЕРСФЕРИЧНИЙ КОНТЕЙНЕР КЛАСІВ РОЗПІЗНАВАННЯ

ЗМІСТ

ВСТУП

Швидкий розвиток інформаційних технологій (ІТ) вплинув майже на всі сфери нашого життя. Комп'ютери, смартфони, розумні годинники та інші мобільні та поширені технології змінюють наш спосіб роботи та те, як ми сприймаємо зовнішній світ. Безсумнівно, наша цивілізація має пристосуватися до багатьох змін, які є наслідком сучасних технологій. Спортивне тренування не є винятком, а також є цікавою сферою, де сучасні технології сприяли революційним змінам у тому, як спортсмени максимізують свою продуктивність і змагаються на більш високому рівні, ніж будь-коли раніше. Таким же чином, із зростанням тенденцій участі в масових спортивних заходах, а також залученням людей до спортивних заходів, виникає потреба в системах/додатках, які можуть направляти, допомагати та підтримувати спортсменів. Наприклад, багато людей у всьому світі не можуть найняти професійного спортивного тренера через багато причин, наприклад, фінансових. З іншого боку, масштабні дослідження, які пов'язують інструменти/методи інтелектуального аналізу даних зі спортивною наукою, створюють нові інтелектуальні рішення, які підтримують усі фази спортивного тренування. Розумне спортивне тренування (РСТ) – це тип спортивного тренування, який використовує пристрої Інтернету речей (IoT) та/або інтелектуальні методи та інструменти аналізу даних для підвищення ефективності тренування та/або зменшення навантаження, а також підтримки продуктивності тренувань. Реалізації РСТ варіюються від простих завдань, таких як виконання інтелектуального аналізу даних спортивного тренування, до набагато складніших реалізацій штучного тренера, де тренера замінює розумний агент, який керує всіма аспектами тренування, за винятком власне виконання пропонованих вправ. Розвиток робочого навантаження може стосуватися як спортсмена, так і його тренера. Для спортсмена вдосконалений план тренувань означає, що він може досягати кращих результатів з меншою кількістю тренувань, а для його тренера це означає, що допомога ІТ-технологій може автоматизувати частини його тренерської рутини.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Спортивне тренування

Спортивне тренування - це безперервний процес взаємодії між спортсменом і його тренером. Це педагогічно організований процес, у якому роль тренера полягає в керівництві діяльністю спортсмена та організації його тренувальних занять. Тренувальні вправи – це чітко визначені завдання, які вимагають фізичних зусиль і мають певним чином покращувати спортивні результати спортсмена. Декілька тренувальних вправ організовуються в повні блоки, які називаються тренувальними сесіями. Кінцевою метою тренування є вдосконалення здібностей спортсмена, тобто досягнення його природного потенціалу. Безперервний процес навчання можна коротко розбити на наступні чотири фази:

1. Планування відноситься до призначення відповідних одиниць вправ. Цикл спортивних тренувань орієнтований на календар змагань. Це етап, на якому тренер готує графік вправ для спортсмена.

2. Реалізація є етапом виконання підготовлених вправ. Ролі тренера на цьому етапі: підготовка (потенційного) обладнання, проведення психофізичної оцінки спортсмена перед заняттям, моніторинг інтенсивності заняття та покращення тактики в командних видах спорту. Дані вправ, необхідні для подальшого аналізу, обов’язково фіксуються на цьому кроці.

3. Контроль – це порівняння фактично виконаних спортсменом вправ із запланованими. Контроль може бути здійснено за допомогою аналізу відео та застосуванні сучасних обчислювальних технологій. В індивідуальних видах спорту можна проводити біометричний аналіз продуктивності, тоді як у командних видах спорту використовуються системи нотаційного аналізу.

4. Оцінка – це вимірювання продуктивності спортсмена. Існують два види оцінок: (1) оцінка окремого тренувального навантаження (короткостроковий аналіз ефективності) та (2) оцінка загального навантаження тренувального циклу (довгостроковий аналіз ефективності). Оцінка – це порівняння між

поставленими цілями та досягнутими результатами та кількістю запланованих і фактично виконаних вправ.

Кожен цикл повинен забезпечувати спортсмену кращі результати. Оскільки спортивне тренування є діяльністю принаймні двох сторін, зокрема тренера та спортсмена, різні обчислювальні підходи можуть бути використані, щоб допомогти тренеру приймати рішення, або взагалі замінити його, запровадивши віртуального помічника.

1.2 Методи інтелектуального аналізу даних розумного спортивного тренування

Розглянемо методи інтелектуального аналізу даних в області розумного спортивного тренування. На рис. 1.1 подано таксономію цих методів, що складається з п’яти основних груп. При цьому деякі алгоритми можуть бути входити в більш ніж одну групу, наприклад, штучні нейронні мережі. В цьому огляді штучні нейронні мережі було розміщено в групу машинного навчання, оскільки більшість досліджень з використанням штучних нейронних мереж, також використовували інші методи машинного навчання, наприклад, дерева рішень. Перелічимо ці групи:

- методи обчислювального інтелекту [1]:
  - еволюційні алгоритми
    - методи диференціальної еволюції[2]
  - ройові алгоритми
    - метод BAT [3]
    - метод рою часток [4]
  - нечіткі системи [5]
  - метод імітації відпалу [6]
- методи інтелектуального аналізу даних
  - алгоритм Apriori [7]
  - вузькоспеціалізовані методи аналізу даних [8]
  - машинне навчання

        - Методи AdaBoost [9]
        - Штучні нейронні мережі [14]
        - Дерева рішень [8]
        - Метод градієнтного бустінгу [11]
        - Ієрархічна кластеризація [15]
        - Кластеризація методом К-середніх [16]
        - Кластеризація методом К-найближчих сусідів [12]
        - Випадковий ліс [10]
        - Метод опорних векторів [13]
- глибоке навчання [17]
    - Згорткові нейронні мережі [20]
    - Довга короткочасна пам'ять [19]
    - Рекурентні нейронні мережі [18]
- Інші методи
    - Байєсівські мережі [23]
    - Умовиводи на основі прецедентів [21]
    - Алгоритм динамічної трансформації часової шкали [22]
    - Гауссівські процеси [26]
    - Узагальнені адитивні моделі [25]
    - Лінійний дискримінантний аналіз [29]
    - Лінійні регресії [27]
    - Марківські ланцюги [24]
    - Наївний баєсівський класифікатор [23]
    - Регуляризована логістична регресія [28]
    - Сплайн-інтерполяція [30]

Деякі дослідження чітко не зазначали використовувані алгоритми, а лише область, з якої вони були (наприклад, [31] або використовували спеціалізовані алгоритми (наприклад, [32]), у таких випадках використовуваний метод був ідентифікований як користувацький і поле

аналізу даних, з якого використовувався метод (наприклад, [32] користувацький алгоритм аналізу даних).

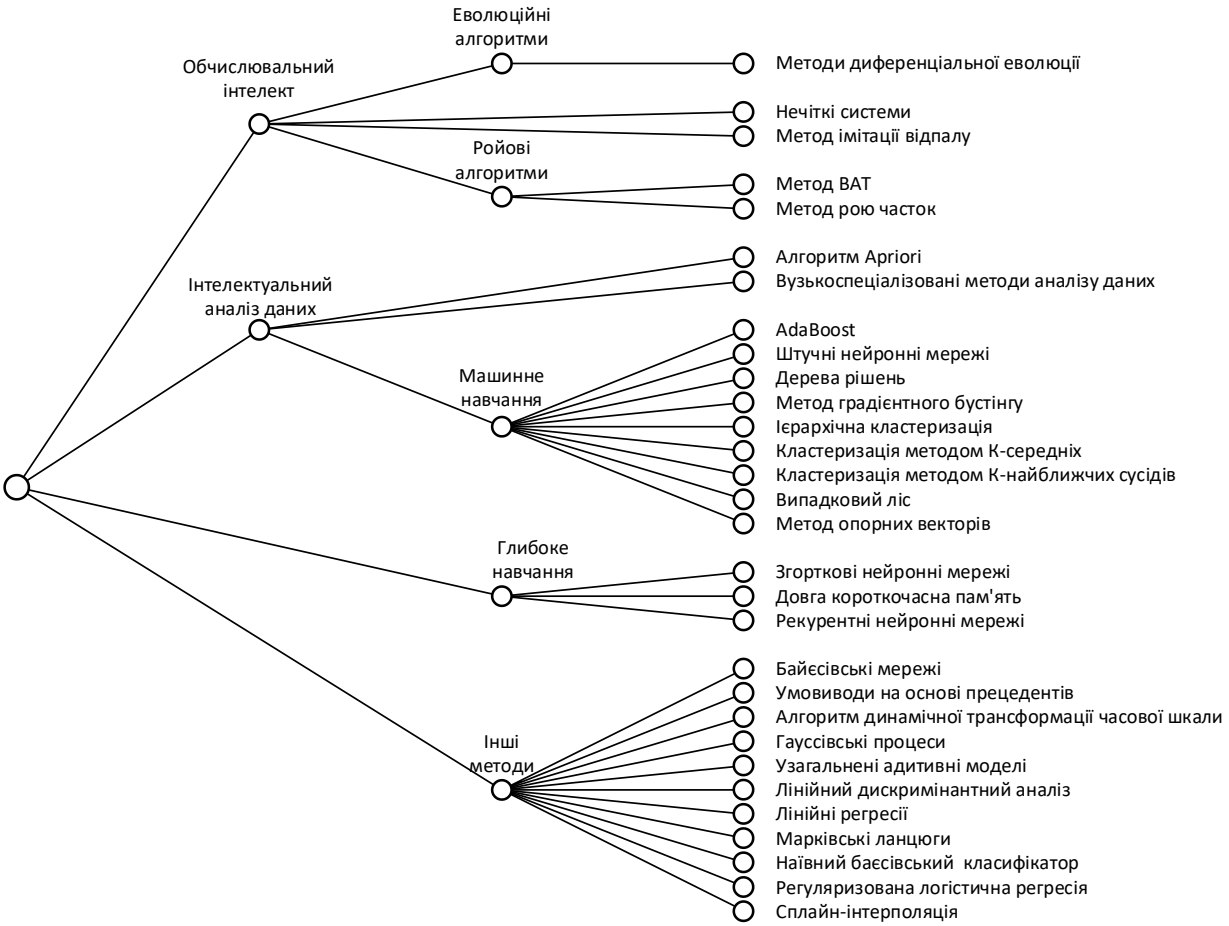


Рисунок 1.1 – Методи інтелектуального аналізу даних розумного спортивного тренування

Зазначені методи використовувалися для аналізу даних з таких видів спорту: айкідо, стрільба з лука, бадмінтон, баскетбол, скелелазіння, стрибки в висоту, крикет, їзда на велосипеді, фехтування, фітнес, американській футбол, гольф, метання молота, гандбол, хокей, стрибки в довжину, карате, кік-бокс, веслування, біг, їзда на велосипеді, стрільба, стрибки з трампліна, лижі, футбол, плавання, настільний теніс, тай-чі, теніс, триатлон, волейбол, важка атлетика та йога. Решта досліджень не стосувалися конкретної дисципліни і стосувалися спортивної підготовки в цілому.

Найбільш популярним серед розробників систем розумного тренування серед зазначених видів спорту вважається біг. У таблиці 1.1 представлено

дослідження розумних спортивних тренувань, проведених у сфері бігу. При цьому для усіх чотирьох етапів спортивного тренування дослідниками було застосовано певний метод інтелектуального аналізу даних. Були також представлені деякі методи для постаналізу продуктивності бігунів (наприклад, [33,34], щоб визначити, де швидкість бігу була недостатньою. Оскільки все тренування бігунів полягає в швидкому переміщенні по стадіону чи біговій доріжці, тренери обмежені у методах безпосереднього спостереження. Цю проблему вирішують за допомогою носимих датчиків, які миттєво надають зворотній зв'язок спортсмену і тренеру, як і у випадку виявлення втоми [35] або відхилення серцевого ритму [36].

Таблиця 1.1 – Методи інтелектуального аналізу даних розумних спортивних тренувань з бігу

Метод	Короткий опис	Етапи тренування			
		Планування	Реалізація	Контроль	Оцінка
1	2	3	4	5	6
Метод імітації відпалу	Планування оптимальної швидкості бігу спортсмена шляхом оцінки фізичних зусиль, необхідних на кожній частині змагань і тренувань. Надано опис підходу до розробки адаптивної системи підтримки прийняття рішень спортивного тренування на основі оцінки оптимальної швидкості бігу спортсмена.[38]	Повністю	Повністю	Частково	Частково
Комплекс методів	Запропоновано персональна система тренувань на свіжому повітрі з урахуванням контексту та адаптивними рекомендаціями користувача на основі чуттєвого контексту, моделі користувача та знань, отриманих від персонального тренера та спортивного фізіолога.[37]	Частково	Частково	Частково	Частково
Методи диференціальної еволюції	Постійний аналіз спортивних біометричних даних в ході марафонського бігу [34], в тому числі серцевого ритму спортсмена [33]	-	-	-	Повністю

Продовження табл. 1.1

1	2	3	4	5	6
Кластеризація методом К-найближчих сусідів	Реалізація інтелектуальної системи, яка застосовується для занять біговими видами спорту на відкритому повітрі, з підтримкою персоналізованого зворотного зв'язку в реальному часі. [39]	-	Повністю	Повністю	-
Штучні нейронні мережі Метод градієнтного бустінгу Лінійні регресії	Система виявлення втоми та попередження для запобігання травм під час тренування. Запропоновано модель визначення швидкість сприйнятого навантаження [35]	-	Повністю	Повністю	-
Метод градієнтного бустінгу	Прогнозування фінішного часу спортсменів на дистанціях 800 та 5000 м на основі даних про вправи та харчування [40]	-	Повністю	Повністю	Повністю
Байєсівські мережі	інтелектуальна система, яка ефективно використовує метод оптимізації інтервалів тренування для кожної людини за допомогою схем інтелекту даних щодо атрибутів, умов і даних, зібраних під час сеансів фізичних вправ. [41]	Повністю	Повністю	Повністю	-
Метод опорних векторів Кластеризація методом К-найближчих сусідів Сплайн-інтерполяція	Демонстрація доцільності використання технологій штучного інтелекту для занять спортом на свіжому повітрі. Система, яка вибрала користувальницький режим тренувань для регулювання складності бігу на основі аналізу пульсу під час тренування.[36]	-	Повністю	-	-

1.3 Постановка задачі

Результати проведеного аналітичного огляду доводять актуальність практичної задачі розробки інформаційної інтелектуальної системи інтелектуального аналізу даних розумних спортивних тренувань з бігу. Для досягнення поставленої мети в роботі запропоновано використання інформаційно-екстремальної інтелектуальної технології машинного навчання та розпізнавання образів.

При цьому основні завдання роботи включають:

1) Формування вхідного математичного опису системи інтелектуального аналізу даних розумних спортивних тренувань з бігу

2) Розробка математичної моделі навчання інтелектуального аналізу даних розумних спортивних тренувань з бігу.
3) Вибір критерію функціональної ефективності параметрів системи інтелектуального аналізу даних розумних спортивних тренувань з бігу
4) Розробка та програмна реалізація алгоритмів навчання системи інтелектуального аналізу даних розумних спортивних тренувань з бігу
5) Перевірка працездатності системи інтелектуального аналізу даних розумних спортивних тренувань з бігу.

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Основні положення інформаційно-екстремальної інтелектуальної технології

Інформаційно-екстремальна інтелектуальна технологія (IEI-технологія) базується на методі функціонально-статистичних випробувань (МФСВ). Цей непараметричний метод інформаційного аналізу та синтезу інтелектуальної системи (ІС) виконує пряму оцінку її інформаційної здатності в умовах нечіткого компактного розподілу реалізацій образу та обмеженого обсягу навчальної вибірки.

Два основних дистанційних принципах МФСВ:

1. максимально-дистанційний, за яким максимізується міжцентрова кодова відстань між класами;
2. мінімально-дистанційний, за яким мінімізується усереднена кодова відстань від реалізацій до центру відповідного класу.

За IEI-технологією клас розпізнавання або образ X_m^o – це відбиття ознак m-го стану об’єкта, що діагностується, і відношень між його елементами. В просторі ознак розпізнавання (ОР) це топологічна категорія – область $X_m^o \subset \Omega_Б$.

Система нормованих допусків $\{\delta_{H,i} \mid i = \overline{1,N}\}$ вводиться так, щоб значення i–ї ОР належало цьому полю з імовірністю p_i=1 або p_i=0, якщо стан об’єкта, що діагностується, відноситься до базового класу X_1^o.

Система контрольних допусків (СКД) $\{\delta_{K,i} \mid i = \overline{1,N}\}$ вводиться так, щоб значення i–ї ОР належало цьому полю з імовірністю $0 < p_i < 1$, якщо стан об’єкта, що діагностується, відноситься до базового класу X_1^o.

Реалізація образу $x_m^{(j)} \in X_m^o$ – це структурований вектор ОР

$$x_m^{(j)} = < x_{m,1}^{(j)}, \ldots, x_{m,i}^{(j)}, \ldots, x_{m,N}^{(j)} >, \; j = 1, \ldots, n_{\min},$$

де $x_{m,i}^{(j)}$ – приймає значення 1, якщо значення i-ї ОР знаходиться в СКД $\delta_{K,i}$, і значення 0, в іншому випадку; $n_{\min}$ – мінімальна кількість реалізацій, яка забезпечує репрезентативність навчальної вибірки.

Еталонний вектор (ЕВ) x_m – це центр розподілу реалізацій класу X_m^o.

$$x_m = \left\langle \frac{\sum_{j=1}^{n_{\min}} x_{m,1}^{(j)}}{n_{\min}}, \ldots, \frac{\sum_{j=1}^{n_{\min}} x_{m,i}^{(j)}}{n_{\min}}, \ldots, \frac{\sum_{j=1}^{n_{\min}} x_{m,N}^{(j)}}{n_{\min}} \right\rangle$$

Нехай задано алфавіт класів розпізнавання $\{X_m^o \mid m = \overline{1,M}\}$, кожний з яких характеризується репрезентативною множиною зображень об'єкту відповідної патології. У процесі оброблення зображень сформовано шляхом зчитування в пікселях рецепторного поля значень градацій яскравості вхідну навчальну матрицю яскравості зображень $\| y_{m,i}^{(j)} \mid i = \overline{1,N}, j = \overline{1,n} \|$, де N, n – кількість ознак розпізнавання в багатовимірному векторі-реалізації (далі просто реалізація) образу і кількість реалізацій, яка дорівнює кількості зображень однієї патології, що обробляються. Таким чином, i-й стовпчик матриці $\| y_{m,i}^{(j)} \|$ містить значення навчальної вибірки, а j-й рядок є реалізацією з N діагностичними ознаками.

Відомий вектор параметрів функціонування

$$g_m = < x_m, \delta >, \qquad (2.1)$$

де x_m – еталонний вектор-реалізація, який визначає центр контейнера класу розпізнавання X_m^o;

δ – параметр поля контрольних допусків на ознаки розпізнавання;

У рамках IEI-технології на параметри функціонування накладаються такі обмеження:

- $x_m \in [0; d(x_m \oplus x_c) - 1]$, де x_c – еталонний вектор-реалізація класу, який є найближчим до класу X_m^o;
- $\delta \in \left[0; {\delta_H}/{2}\right]$, де δ_H – нормоване значення контрольних допусків на ознаки розпізнавання;.

На етапі навчання необхідно оптимізувати (тут і далі в інформаційному розумінні) значення координат вектору (2.1) параметрів функціонування, які забезпечують максимальне усереднене за алфавітом $\{X_m^o\}$ значення інформаційного КФЕ навчання діагностичної СППР, обчислене в робочій (допустимій() області визначення функції критерію оптимізації:

$$\overline{E}^* = \frac{1}{M} \sum_{m=1}^{M} \max_{<\{k\}>} E_m, \qquad (2.2)$$

де E_m – інформаційний КФЕ навчання СППР розпізнавати реалізації класу X_m^o;

$<\{k\}>$ – впорядкована множина кроків навчання (відновлення контейнерів класів розпізнавання).

На етапі екзамену необхідно прийняти з достовірністю, наближеною до максимальної асимптотичної, рішення про належність реалізації зображення, що розпізнається, до відповідного класу розпізнавання із сформованого на етапі навчання алфавіту класів $\{X_m^o\}$.

2.2 Математична інформаційно-екстремальна модель навчання

Базова категорійна модель інформаційно-екстремального навчання показана на рис. 2.1 діаграма відображення множин, що застосовуються в процесі машинного навчання.

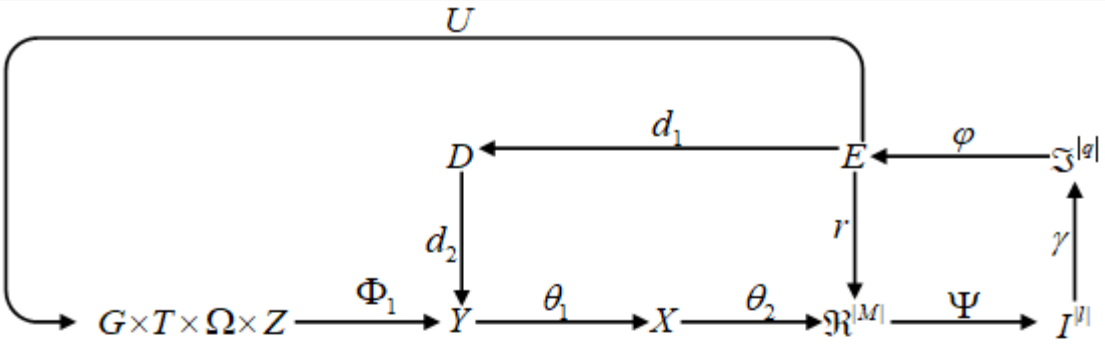


Рисунок 2.1 – Категорійна модель інформаційно-екстремального навчання СППР з оптимізацією контрольних допусків на ознаки розпізнавання

На рис. 2.1 прийнято такі позначення:

Φ_1 – оператор формування вхідної вибіркової множини Y, яка формує вхідну у загальному випадку дійсну багатовимірну навчальну матрицю $\| y_{m,i}^{(j)} \|$, де $m = \overline{1,M}$, $i = \overline{1,N}$, $j = \overline{1,n}$ – змінні кількості класів розпізнавання, діагностичних ознак і реалізацій образів відповідно;

Φ_2 – оператор формування бінарної навчальної матриці $\| x_{m,i}^{(j)} \|$;

θ – оператор побудови в загальному випадку нечіткого розбиття $\tilde{\Re}^{|M|}$ простору ознак розпізнавання на M класів;

Ψ – оператор перевірки основної статистичної гіпотези $\gamma_1 : x_m^{(j)} \in X_m^o$;

$I^{|l|}$ – множина статистичних гіпотез;

$\Im^{|q|}$ – множина точнісних характеристик оцінки статистичних рішень, $q = l^2$;

E – терм-множина значень інформаційного критерію функціональної ефективності (КФЕ) навчання СППР;

D – терм-множина значень системи контрольних допусків на ознаки розпізнавання;

$U : E \to G \times T \times \Omega \times Z$ – оператор регламентації процесу навчання СППР.

Як КФЕ навчання СППР може використовуватися будь-який інформаційний статистичний критерій. Найбільшого поширення в рамках IEI-технології знайшли ентропійні (за Шенноном) критерії

$$E^{(k)} = 1 + \frac{1}{2}\left(\frac{\alpha^{(k)}}{\alpha^{(k)} + D_2^{(k)}} \log_2 \frac{\alpha^{(k)}}{\alpha^{(k)} + D_2^{(k)}} + \right.$$
$$+ \frac{\beta^{(k)}}{D_1^{(k)} + \beta^{(k)}} \log_2 \frac{\beta^{(k)}}{D_1^{(k)} + \beta^{(k)}} +$$
$$+ \frac{D_1^{(k)}}{D_1^{(k)} + \beta^{(k)}} \log_2 \frac{D_1^{(k)}}{D_1^{(k)} + \beta^{(k)}} +$$
$$\left. + \frac{D_2^{(k)}}{\alpha^{(k)} + D_2^{(k)}} \log_2 \frac{D_2^{(k)}}{\alpha^{(k)} + D_2^{(k)}} \right) \quad (2.3)$$

та інформаційна міра Кульбака [25].

$$E^{(k)} = \log_2 \left(\frac{2 - \left(\alpha^{(k)} + \beta^{(k)}\right) + 10^{-r}}{\alpha^{(k)} + \beta^{(k)} + 10^{-r}} \right) * \left[2 - \left(\alpha^{(k)} + \beta^{(k)}\right) \right]. \quad (2.4)$$

Де $\alpha^{(k)}$, $\beta^{(k)}$ - помилки першого та другого роду, $D_1^{(k)}$, $D_2^{(k)}$ перша та друга достовірності, k – крок навчання, r – константа, яка дорівнює 0,01.

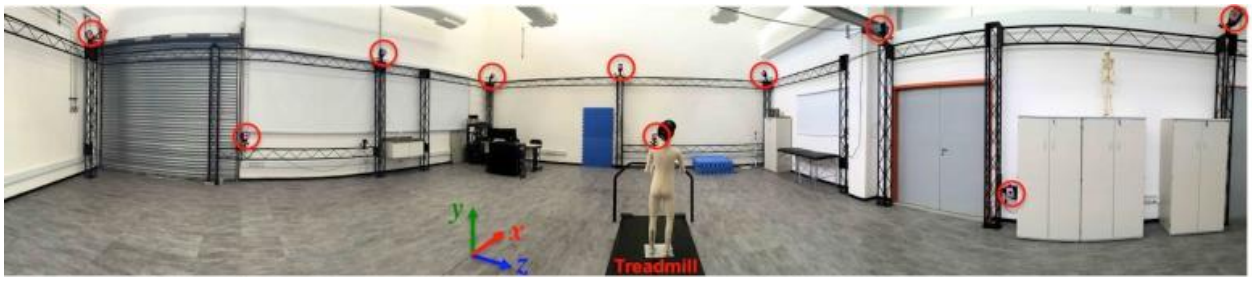
При функціонуванні діагностичної СППР в режимі екзамену, тобто безпосереднього розпізнавання зображень, основною її функцією є прийняття рішення про належність або не належність реалізації, що розпізнається до одного з класів алфавіту, тобто етап екзамену.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

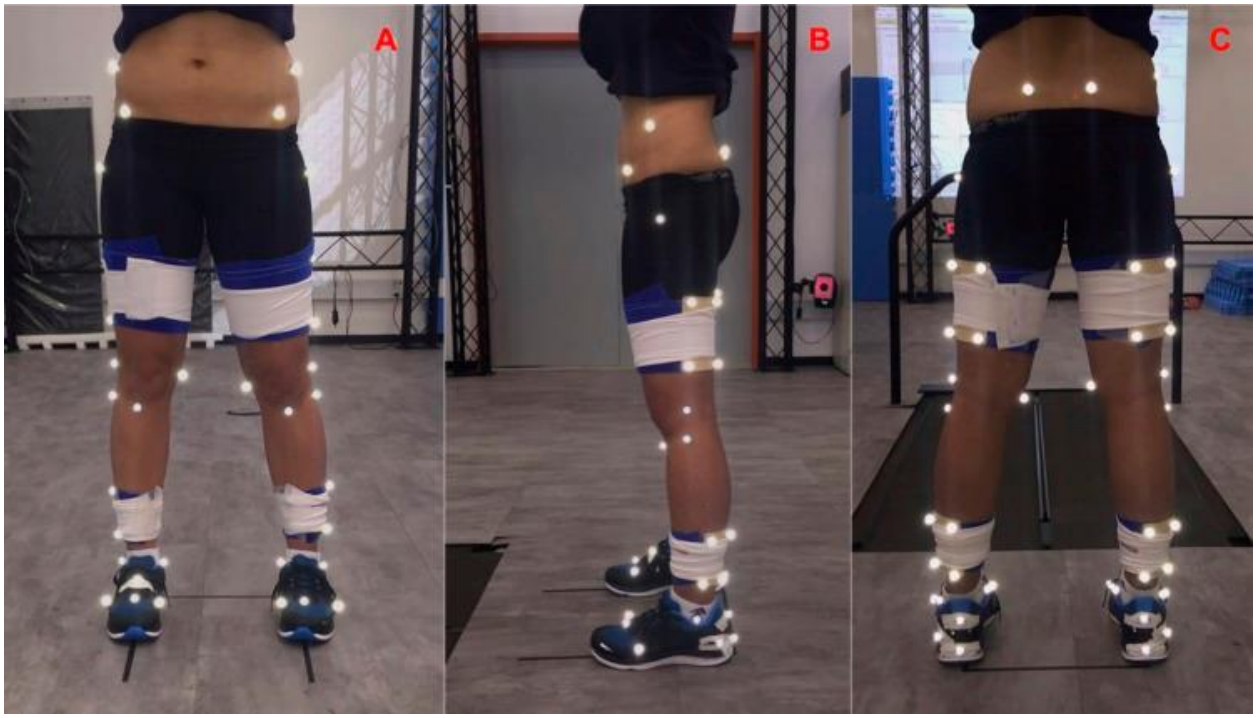
3.1 Опис вхідних даних

При формуванні вхідних даних було використано загально доступну базу біомеханічних показників бігунів різного класу майстерності лабораторії BMClab бразильського федерального університету ABC.

При формуванні даних використовувалося обладнання для фіксації руху в 3D форматі за допомогою 12 камер з роздільною здатністю 4 Мб і програмним забезпеченням Cortex 6.0. При цьому камери було розподіллено по приміщенню лабораторії (рис. 3.1 а)



а



б

Рисунок 3.1 – Розташування обладнання для формування вхідних даних: а) камер, б) маркерів

Програмне забезпечення Cortex 6.0 було використано для

- калібрування маркерів захоплення руху;
- захоплення та ідентифікація світловідбиваючих маркерів;
- підготувати дані та експортувати їх у формат файлу c3d.

У дослідженні було використано 48 технічних та анатомічних світловідбиваючих маркерів (рис. 3.1 б). На сегментах стегна та гомілки використовували кластери з чотирма технічними маркерами, розміщеними в жорсткій оболонці.

В таблиці 3.1 наведено перелік показників, що формувалися при зчитуванні положення маркерів під час бігу.

Таблиця 3.1 – Опис маркерів

№	**Назва маркеру**	**Опис**
1	R.ASIS	Права передня верхня клубова кістка
2	L.ASIS	Ліва передня верхня клубова кістка
3	R.PSIS	Права задня клубова кістка
4	L.PSIS	Задня ліва клубова кістка
5	R.Iliac.Crest	Правий гребінь клубової кістки
6	L.Iliac.Crest	Лівий гребінь клубової кістки
7	R.Thigh.Top.Lateral	Верхній бічний маркер правого стегна
8	R.Thigh.Bottom.Lateral	Бічний маркер знизу правого стегна
9	R.Thigh.Top.Medial	Верхній медіальний маркер правого стегна
10	R.Thigh.Bottom.Medial	Медіальний маркер знизу правого стегна
11	R.Shank.Top.Lateral	Верхній бічний маркер правого хвостовика
12	R.Shank.Bottom.Lateral	Нижній бічний маркер правого хвостовика
13	R.Shank.Top.Medial	Верхній медіальний маркер правої гомілки
14	R.Shank.Bottom.Medial	Нижній медіальний маркер правої гомілки
15	R.Heel.Top	Верх правого каблука
16	R.Heel.Bottom	Правий каблук знизу

Продовження табл. 3.1

№	Назва маркеру	Опис
17	R.Heel.Lateral	Права п’ята бічна
18	L.Thigh.Top.Lateral	Верхній бічний маркер лівого стегна
19	L.Thigh.Bottom.Lateral	Бічний маркер нижньої частини стегна
20	L.Thigh.Top.Medial	Медіальний маркер верхнього лівого стегна
21	L.Thigh.Bottom.Medial	Медіальний маркер знизу лівого стегна
22	L.Shank.Top.Lateral	Верхній бічний маркер лівого хвостовика
23	L.Shank.Bottom.Lateral	Нижній бічний маркер лівого хвостовика
24	L.Shank.Top.Medial	Верхній медіальний маркер лівої гомілки
25	L.Shank.Bottom.Medial	Нижній медіальний маркер лівої гомілки
26	L.Heel.Top	Верх лівого каблука
27	L.Heel.Bottom	Нижня частина лівого каблука
28	L.Heel.Lateral	Ліва п’ята збоку
29	R.GTR	Правий великий вертел
30	R.Knee	Праве коліно
31	R.Knee.Medial	Праве коліно медіальне
32	R.HF	Права головка малогомілкової кістки
33	R.TT	Горбик правої великогомілкової кістки
34	R.Ankle	Права щиколотка
35	R.Ankle.Medial	Права щиколотка медіальна
36	R.MT1	Права 1 плесна
37	R.MT5	Права 5 плесна
38	R.MT2	Права 2 плесна
39	L.GTR	Лівий великий вертел
40	L.Knee	Ліве коліно
41	L.Knee.Medial	Медіальне ліве коліно
42	L.HF	Ліва головка малогомілкової кістки
43	L.TT	Горбистість лівої великогомілкової кістки
44	L.Ankle	Ліва щиколотка
45	L.Ankle.Medial	Медіально ліву гомілку
46	L.MT1	Ліва 1 плесна
47	L.MT5	Ліва 5 плесна
48	L.MT2	Ліва 2 плесна

В роботі використовувалися навчальні матриці двох класів – показники спортсмена – професіонала і аматора. Ці навчальні матриці складалися з 48 ознак розпізнавання і 4500 реалізацій, що були сформовані в ході тренування.

3.2 Алгоритм навчання

В ході навчання системи були задіяні контури оптимізації геометричних параметрів вирішальних правил та системи контрольних допусків на ознаки розпізнавання. Відповідні алгоритми подамо у вигляді блок-схем (рис. 3.2-3.3)

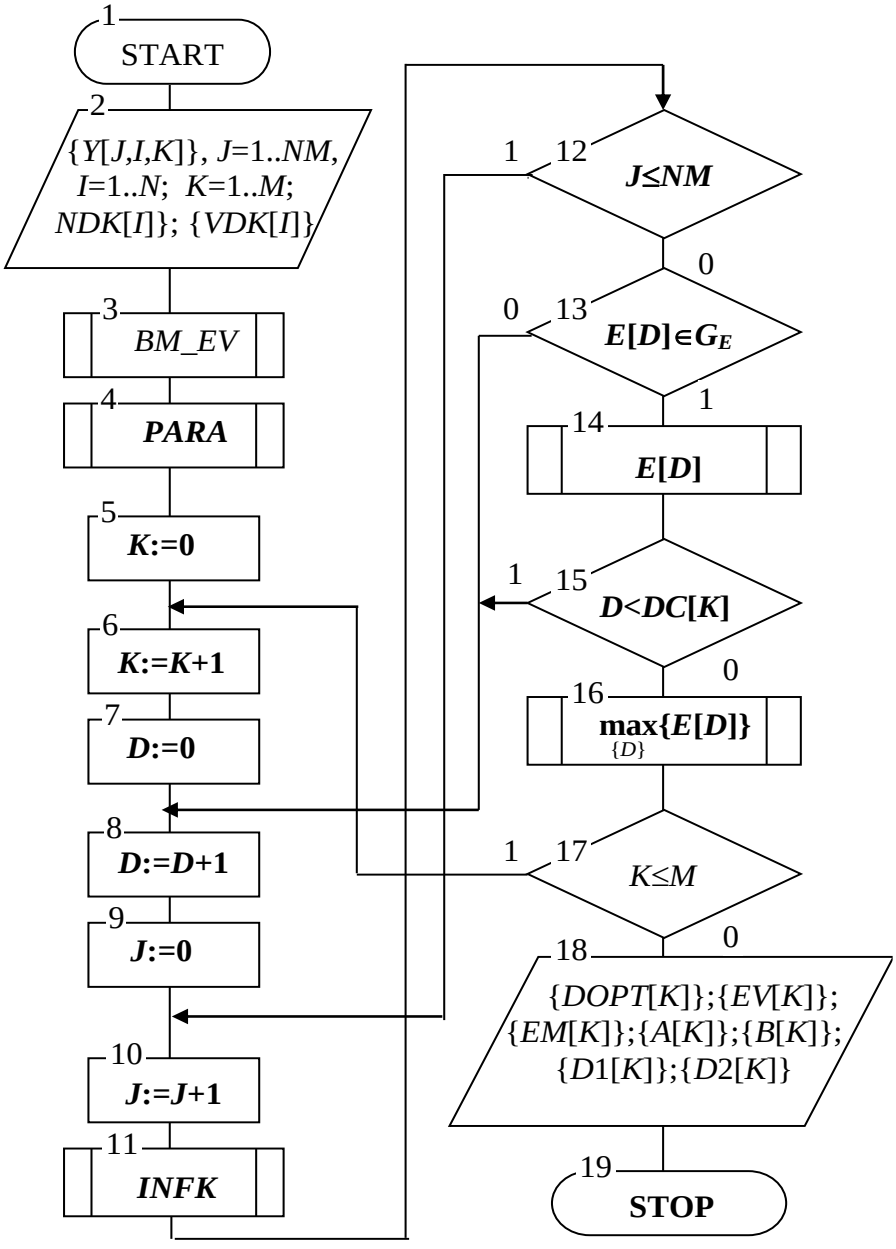


Рисунок 3.2 – Алгоритм оптимізації геометричних параметрів вирішальних правил

На вхід цього алгоритму подаються навчальна матриця *Y* та система контрольних допусків на ознаки розпізнавання *NDK* і *VDK*.

На виході формуються вирішальні правила для кожного класу розпізнавання, геометричні параметри яких – центри *EV* та радіуси *DOPT* – оптимізовано. Відповідні значення КФЕ і точносних характеристик додаються до вихідних даних алгоритму.

Основними блоками алгоритму поданого на рис. 3.2 є:

- блок 3 – перетворення навчальної матриці *Y* на бінарну навчальну матрицю і формування масиву одного з геометричних параметрів вирішальних правил – еталонних векторів;

- блок 4 – визначення для кожного класу найближчого (сусіднього) класу і обчислення відстані до нього;

- блоки 8-16 – визначення оптимального значення іншого геометричного параметру вирішальних правил – радіуса – для кожного класу;

- блок 11 – оцінка функціональної ефективності поточних значень геометричних параметрів вирішальних правил;

- блок 14 – формування таблиці, що містить множину можливих радіусів і результати оцінки їх функціональної ефективності для кожного класу;

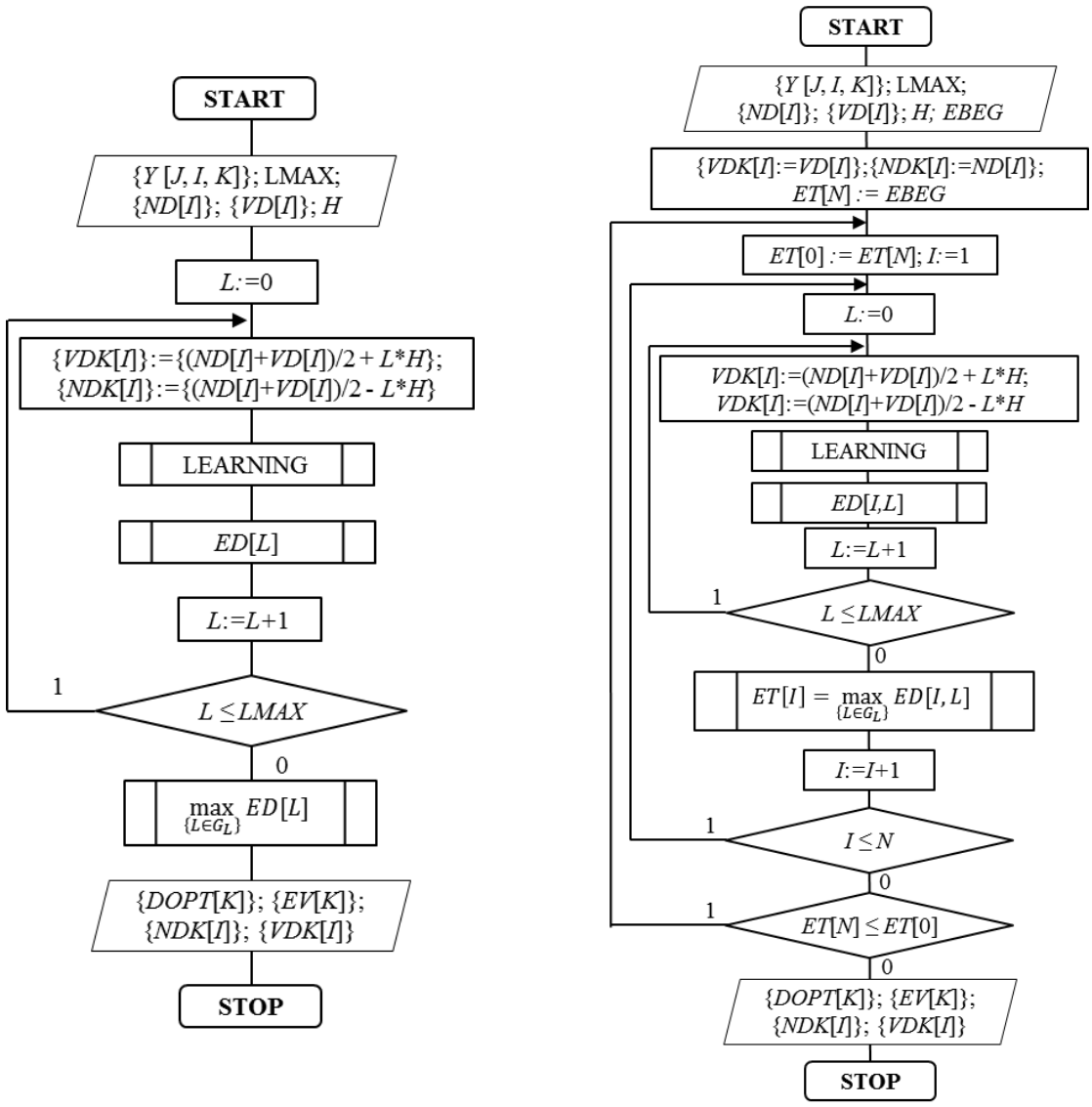
- блок 16 – пошук максимального значення КФЕ і оптимального значення радіуса для поточного класу.

Алгоритм оптимізації СКД можливо реалізувати у вигляді паралельного та послідовного алгоритму. Відповідні блок-схеми подано на рис. 3.3.

На вхід цих алгоритмів подаються навчальна матриця *Y*, система нормованих допусків на ознаки розпізнавання *ND* і *VD* та крок зміни значень системи допусків *H*. Для послідовного алгоритму на вхід додатково подається значення оптимізованого за паралельним алгоритмом параметра ширини поля контрольних допусків.

На виході ці алгоритми формують оптимізовану систему контрольних допусків і вирішальні правила для кожного класу, що були відтворені за такої системи допусків.

Основними блоками як паралельного, так і послідовного алгоритму оптимізації СКД є блок LEARNING, що реалізує алгоритм оптимізації геометричних параметрів вирішальних правил, блок формування таблиці, що містить множину можливих систем контрольних допусків і результати оцінки їх функціональної ефективності (Цей блок розташовано одразу після блоку LEARNING) та блок пошук максимального значення усередненого КФЕ і оптимального значення СКД.



а) б)

Рисунок 3.3 – Блок схеми оптимізації СКД: а) за паралельним алгоритмом та б) послідовним алгоритмом

Основна відмінність алгоритмів – паралельний алгоритм оптимізує поле допусків для всіх ознак одночасно, а послідовний – для кожної ознаки окремо. Тому в паралельному алгоритмі є лише один цикл, за яким змінюється та оцінюється значення ширини поля контрольних допусків всіх ознак одночасно, а в послідовній – три, що дозволяють виконувати аналогічні дії для кожної ознаки окремо. Крім того, у алгоритмів різні критерії зупину – паралельний алгоритм завершується, якщо було досягнуто максимально можливої ширини поля допусків, а послідовний – якщо не змінним залишається усереднене значення КФЕ перед оптимізацією ширини поля допусків першої і після оптимізації ширини поля допусків останньої ознаки.

Алгоритми екзамену за МФСВ можуть мати різну структуру залежно від розподілу реалізацій образу, що розпізнаються. Обов'язковою умовою їх реалізації є забезпечення однакових структурованості і параметрів формування як для навчальної, так і для екзаменаційної матриць.

За наявності чіткого розбиття, яке було утворено на етапі навчання, алгоритм екзамену за МФСВ має такі вхідні дані:

M – кількість класів, які СПР навчена розпізнавати;

$\{x_m^* \mid m = \overline{1, M}\}$ – масив еталонних двійкових векторів, які визначають центри відповідних оптимальних контейнерів класів розпізнавання, побудованих на етапі навчання;

$\{d_m^*\}$ – масив оптимальних радіусів побудованих на етапі навчання відповідних контейнерів;

$\{x^{(j)} \mid j = \overline{1, n}\}$ – масив двійкових векторів-реалізацій образу, що розпізнається;

$\{\delta_{k,i}^* \mid i = \overline{1, N}\}$ – оптимальна СКД на ознаки розпізнавання, яку визначено на етапі навчання.

За умовчанням приймається рівень селекції $\rho_m = 0{,}5$.

Розглянемо кроки реалізації алгоритму екзамену при застосуванні гіпотези чіткої компактності реалізацій образу:

1. Формування лічильника класів розпізнавання: $m := m + 1.$

2. Формування лічильника числа реалізацій, що розпізнаються: $j := j + 1.$

3. Порівняння: якщо $x^{(j)} \in X_m^o$, то виконується крок 4, інакше – крок 5.

4. Формування лічильника $k := k + 1$ позитивних результатів порівняння.

5. Порівняння: якщо $j \le n$, то виконується крок 2, інакше – крок 6.

6. Порівняння: якщо $k > j / 2$, то виконується крок 8, інакше –крок 7.

7. Порівняння: якщо $m \le M$, то виконується крок 1, інакше – крок 8.

8. Визначення класу X_m^o, до якого належить екзаменаційна матриця.

Для нечіткого розбиття алгоритм екзамену за МФСВ ґрунтується на аналізі значень функції належності, що обчислюється для кожної реалізації, що розпізнається. Розглянемо кроки реалізації алгоритму екзамену при нечіткому розбитті:

1. Формування лічильника $m := m + 1$ класів розпізнавання.

2. Формування лічильника числа реалізацій, що розпізнаються: $j := j + 1.$

3. Обчислення кодової відстані $d(x_m^* \oplus x^{(j)})$.

4. Обчислення функції належності за виразом:

$$\mu_m = 1 - \frac{d(x_m^* \oplus x^{(j)})}{d_m^*}.$$

5. Порівняння: якщо $j \le n$, то виконується крок 2, інакше – крок 6.

6. Порівняння: якщо $m \le M$, то виконується крок 1, інакше – крок 7.

7. Визначення класу X_m^o, до якого належить екзаменаційна реалізація, наприклад, за умови $\overline{\mu}_m^* = \max_{\{m\}} \overline{\mu}_m$, де $\overline{\mu}_m = \frac{1}{n}\sum_{j=1}^{n} \mu_{m,j}$ – усереднене значення функцій належності для реалізацій класу X_m^o, або видача повідомлення: «Клас не визначено», якщо $\overline{\mu}_m^* \le c$. Тут c – порогове значення.

3.3 Короткий опис програмної реалізації

Для програмної реалізації було використано мову C#. При цьому програмний додаток не мав графічного інтерфейсу, запускався з командного рядка, а результати його роботи завантажувалися в текстові файли, аналіз яких і побудова графіків виконувалася в MS Excel. Основні функції додатку наведено в таблиці (табл. 3.2)

Таблиця 3.2 – Основні функції

Назва	Короткий опис
`LoadClass(int num)`	Загрузка реалізацій класу з номером num
`MakeSD2()`	Формування СКД для окремої ознак
`MakeSD()`	Формування СКД для всіх ознак
`MakeBM()`	Перетворення навчальних матриць в бінарну форму
`MakeEV()`	Формування еталонів або центрів контейнерів
`MakeSK(int num)`	Визначення кодових відстаней від еталонів до всіх реалізацій класу num
`INFK(int t1_, int t_, ref double t_D1, ref double t_betta)`	Обчислення КФЕ Кульбака
`SaveClass(int num)`	Збереження основних і проміжних результатів формування вирішального правила для класу num
`MakeDO()`	Пошук радіусів контейнерів класів
`Opt_delta()`	Оптимізація СКД

Повний код наведено в Додатку.

3.4 Аналіз результатів

На першому етапі було застосовано алгоритм паралельної оптимізації СКД (рис. 3.3 а). При цьому ширину поля допусків обмежимо інтервалом від 0 до 100 і будемо змінювати її з кроком 1. Результати такої оптимізації СКД подано на рис. 3.4.

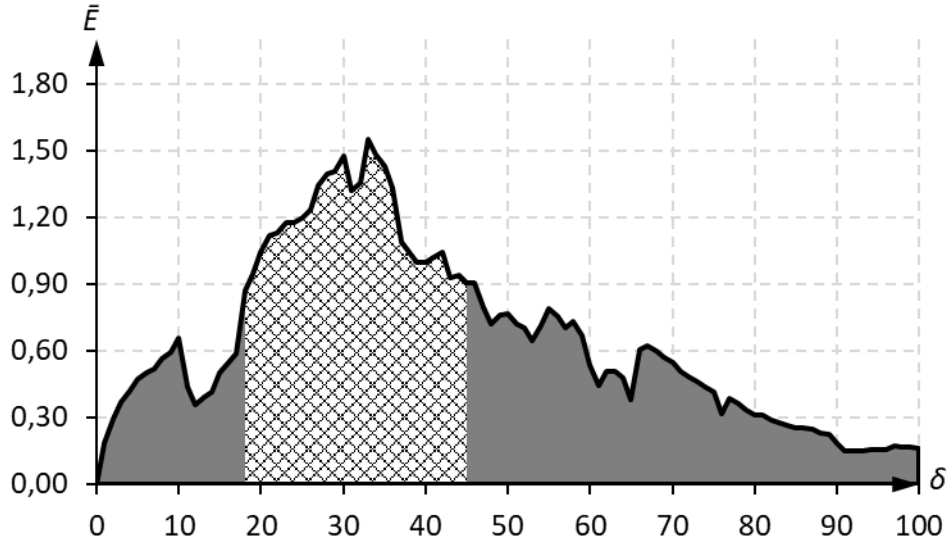
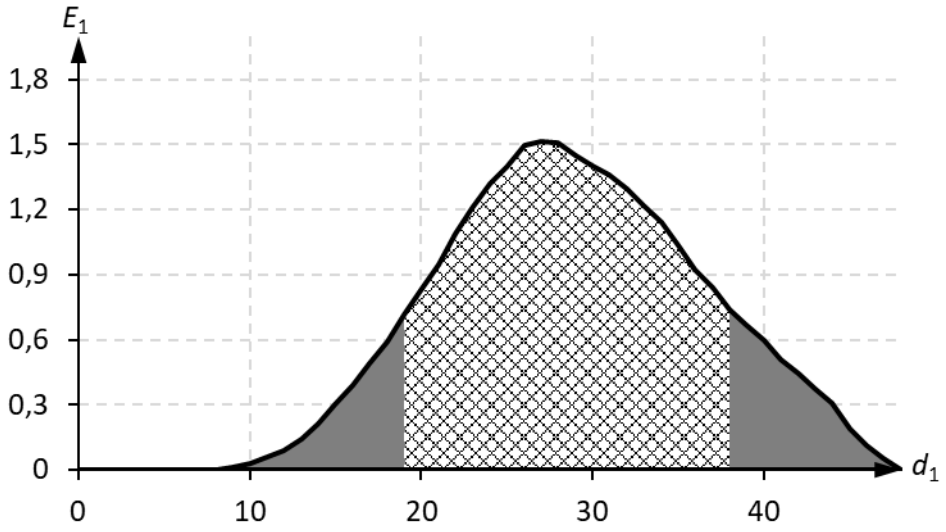
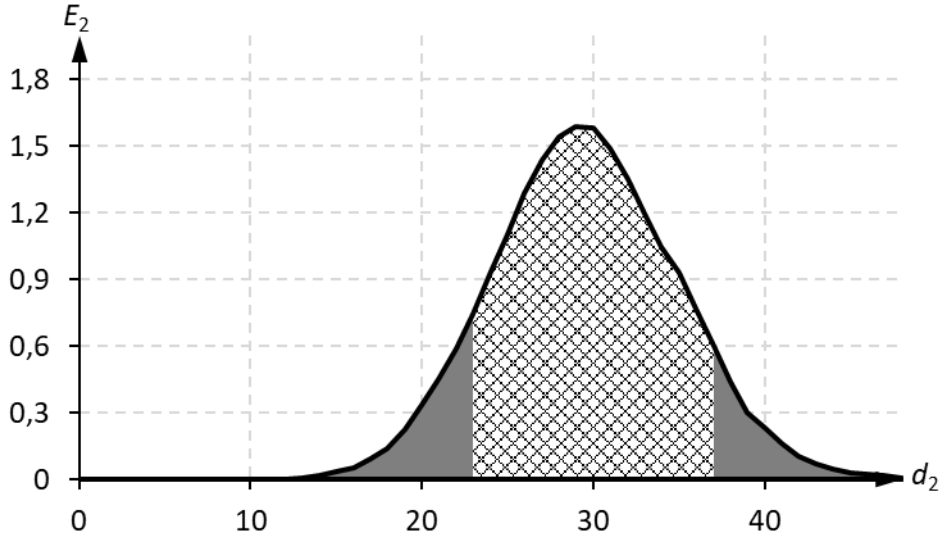


Рисунок 3.4 – Графік залежності усередненого критерію функціональної ефективності від ширини поля допусків при її оптимізації за паралельним алгоритмом

Аналіз рис. 3.4 показує, що максимальне значення усередненого критерію функціональної ефективності дорівнює 1,55232 на 33 кроці оптимізації. Оптимізація геометричних параметрів за такої СКД подана на рис. 3.5.



а)



б)

Рисунок 3.5 – Графік залежності критерію функціональної ефективності від радіусів класів: а) X_1^o; б) X_2^o

Результати аналізу рис. 3.5 наведено в табл. 3.3.

Таблиця 3.3 – Результати навчання

Клас	КФЕ	Радіус	Міжцентрова відстань	D_1	β
X_1^o	1.51560	27	30	0,822	0,161
X_2^o	1.58903	29	30	0.795	0,121

Аналіз табл. 3.3 показує, що після паралельної оптимізації системи контрольних допусків точність вирішальних правил в середньому складає 83,375%. Для підвищення точності вирішальних правил застосуємо послідовний алгоритм оптимізації СКД (рис. 3.3 б). Результати такої оптимізації подано на рис. 3.6.

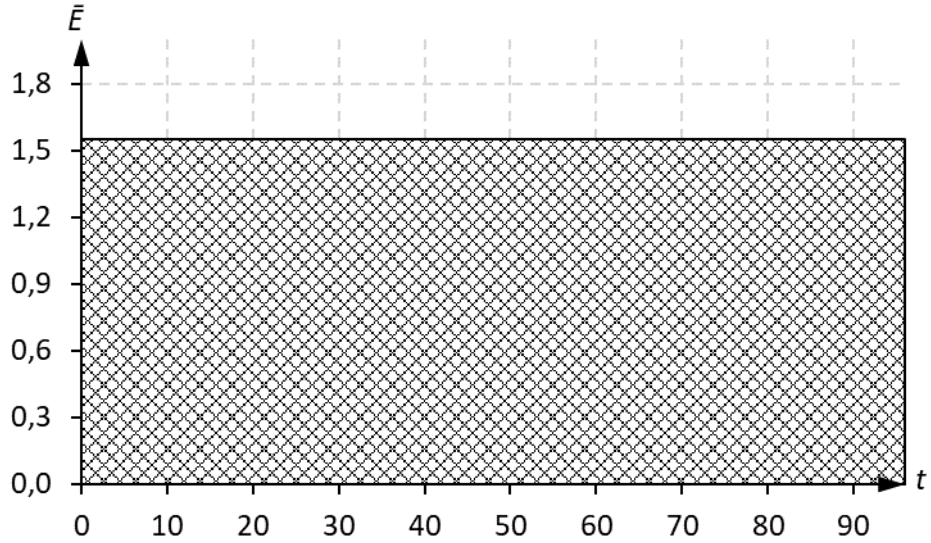
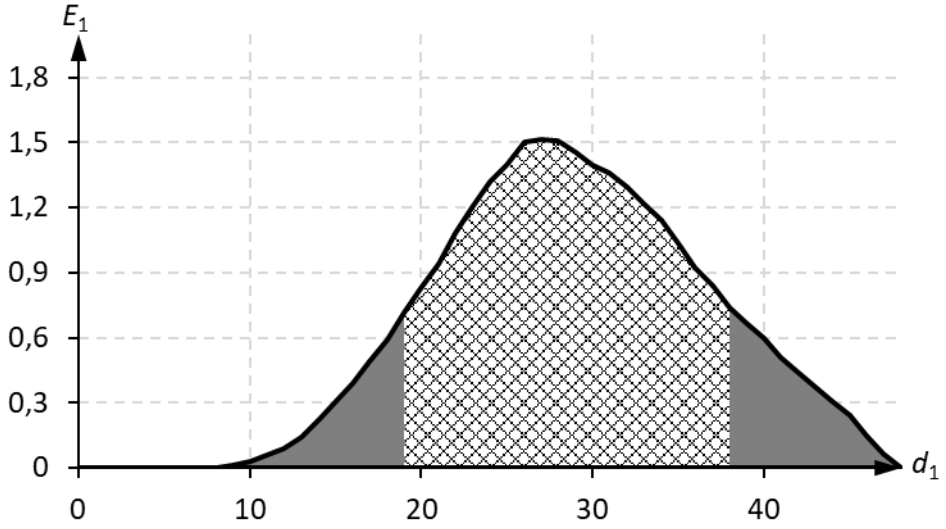
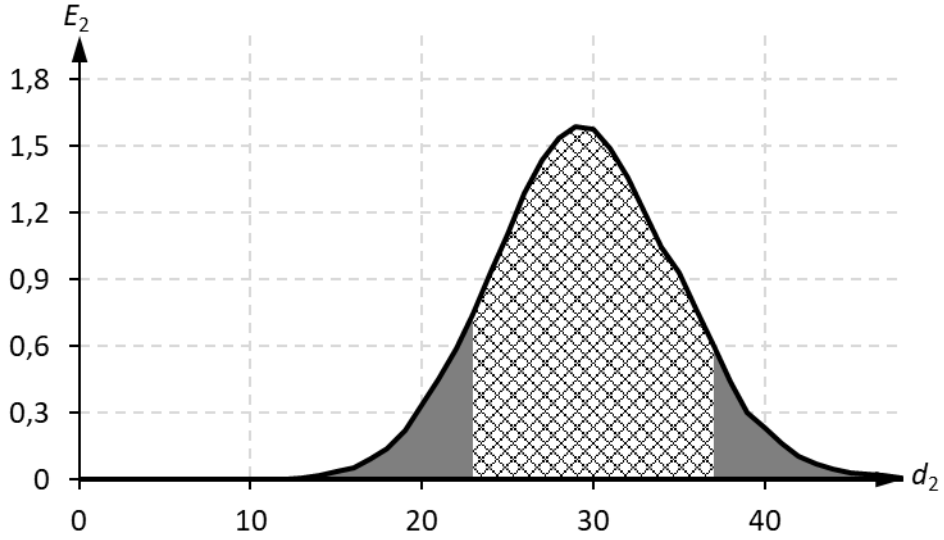


Рисунок 3.6 – Графік залежності усередненого КФЕ від ширини поля допусків

Аналіз рис. 3.6 показує, що максимальне значення усередненого КФЕ не змінилося, що доводять і результати оптимізації геометричних параметрів за такої СКД (рис. 3.7).



а)



б)

Рисунок 3.7 – Графік залежності критерію функціональної ефективності від радіусів класів: а) X_1^o; б) X_2^o

Таблиця 3.4 – Результати навчання

Клас	КФЕ	Радіус	Міжцентрова відстань	D_1	β
X_1^o	1.51687	27	30	0.822	0.161
X_2^o	1.59035	29	30	0.794	0.120

Аналіз табл. 3.4 показує, що з використанням послідовної оптимізації СКД точність вирішальних правил усереднена точність не змінилася і складає 83,375%.

ВИСНОВКИ

В роботі в рамках інформаційно-екстремальної інтелектуальної технології машинного навчання та розпізнавання образів розроблено систему інтелектуального аналізу даних розумного спортивного тренування.

При цьому виконано такі основні завдання:

1) Сформовано вхідний математичний опис системи інтелектуального аналізу даних розумних спортивних тренувань з бігу;
2) Як математичну модель навчання системи інтелектуального аналізу даних розумних спортивних тренувань з бігу використано об’єднання двох базових моделей навчання за інформаційно-екстремальною інтелектуальною технологією, а саме моделі навчання з оптимізацією геометричних параметрів вирішальних правил і моделі з оптимізацією системи контрольних допусків на ознаки розпізнавання.
3) Як критерії функціональної ефективності параметрів системи використано інформаційну міру Кульбака.
4) Розроблено та програмно реалізовано з використанням мови C# алгоритми навчання системи інтелектуального аналізу даних розумних спортивних тренувань з бігу
5) Перевірка працездатності системи інтелектуального аналізу даних розумних спортивних тренувань з бігу виконано для задачі формування вирішальних правил для двох класів – спортсмен - професіонал та спортсмен-аматор, що виконують однакові вправи в рамках спортивного тренування.

СПИСОК ЛІТЕРАТУРИ

1. Engelbrecht A.P. Computational Intelligence / A.P. Engelbrecht // New York, 2007.
2. Storn R. A simple and efficient heuristic for global optimization over continuous spaces / R. Storn, K. Price // Glob. Optim, 1997. – № 11, p. 341–359.
3. Yang X.S. Bat algorithm / X.S. Yang // arXiv 2013.
4. Kennedy J. Particle swarm optimization / J. Kennedy, R. Eberhart // ICNN'95-International Conference on Neural Networks, 1995. – № 4. – p. 1942–1948.
5. Sugeno M. Fuzzy Systems Theory and Its Applications / M. Sugeno, K. Asai, T. Terano // Tokyo Institute of Technology, 1992.
6. Laarhoven P.J. Simulated annealing / P.J. Laarhoven, E.H. Aarts // Springer, 1987. – p. 7–15.
7. Agrawal R. Fast algorithms for mining association rules / R. Agrawal, R. Srikant // 20th International Conference on Very Large Data Bases, 1994.– № 1215.– p. 487–499.
8. Quinlan J. Induction of decision trees / J.R. Quinlan / Mach. Learn. 1986. – № 1. – p. 81–106.
9. Margineantu D.D. Pruning Adaptive Boosting / D.D. Margineantu, T.G. Dietterich // ICML, 1997. – № 97.– p. 211–218.
10. Breiman L. Random forests / L. Breiman // Mach. Learn., 2001. – № 45. – p. 5–32.
11. Friedman J.H. Stochastic gradient boosting / J.H. Friedman // Computer Statistical Data Analysis, 2002.– № 38.– p. 367–378.
12. Peterson L.E. K-nearest neighbor / L.E. Peterson // Scholarpedia, 2009. – №4. – p. 1883.

13. Drucker H. Support vector regression machines / H. Drucker, C.J. Burges, L. Kaufman, A.J. Smola, V. Vapnik // Advances In Neural Information Processing Systems, 1997. – p. 155–161.
14. Beale H.D. Neural Network Design / H.D. Beale, H.B. Demuth, M. Hagan //Pws, 1996.
15. Johnson S.C. Hierarchical clustering schemes / S.C. Johnson // Psychometrika, 1967. - №32.– p. 241–254.
16. Kanungo T. An efficient k-means clustering algorithm / T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, A.Y. Wu // Pattern Anal. Mach. Intell., 2002.– № 24.– p. 881–892.
17. LeCun Y. Deep learning / Y. LeCun, Y. Bengio, G. Hinton // Nature, 2015. – № 521.– p. 436–444.
18. Mikolov T. Recurrent neural network based language model / T. Mikolov, M. Karafiát, L. Burget, J. Cernock, S. Khudanpur // Eleventh Annual Conference of the International Speech Communication Association, 2010.
19. Hochreiter S. Long short-term memory / S. Hochreiter, J. Schmidhuber // Neural Comput., 1997. – №9.– p. 1735–1780.
20. Lawrence S. A convolutional neural-network approach / S. Lawrence, C.L. Giles, A.C. Tsoi, A.D. Back // Neural Netw., 1997. – № 8. – p. 98–113.
21. Kolodner J. Case-Based Reasoning / J. Kolodner // Morgan Kaufmann, 2014.
22. Berndt D.J. Using Dynamic Time Warping to Find Patterns in Time Series / D.J. Berndt, J. Clifford // KDD Workshop, 1994. – № 10. – p. 359–370.
23. Cheng J. Comparing Bayesian Network Classifiers / J. Cheng, R. Ell Greiner. – https://arxiv.org/ftp/arxiv/papers/1301/1301.6684.pdf
24. Geyer C.J. Practical markov chain monte carlo / C.J. Geyer // Stat. Sci. 1992. – № 7. – p. 473–483.

25. Hastie T.J. Generalized additive models / T.J. Hastie // Statistical Models, 2017. – p. 249–307.
26. Bonilla E.V. Multi-Task Gaussian Process Prediction / E.V. Bonilla, K.M. Chai, C. Williams // Advances in Neural Information Processing Systems. – https://homepages.inf.ed.ac.uk/ckiw/postscript/multitask GP_v22.pdf
27. Seber G.A. Linear Regression Analysis / G.A. Seber, A.J. Lee // John Wiley & Sons, 2012. – № 329.
28. Lee S.I. Regularized Logistic Regression / S.I. Lee, H. Lee, P. Abbeel, A.Y. Ng, L. Efficient // AAAI, 2006. – № 6.– p. 401–408.
29. Balakrishnama S. Linear discriminant analysis / S. Balakrishnama, A. Ganapathiraju // Inst. Signal Inf. Process, 1998.– № 18.– p. 1–8.
30. Schoenberg I.J. Cardinal Spline Interpolation / I.J. Schoenberg // Siam, 1973.– № 12.
31. Lin Z. The Design and Implementation of Shooting Training and Intelligent Evaluation System / Z. Lin, S. Wu // Emerging Computation and Information Technologies for Education, 2012. – p. 107–115.
32. Guangjun L. Knowledge Rule Discovery Based on Training Data of Rowing / L. Guangjun, P. Kejun // International Conference on Future Computer Science and Education, 2011. – p. 338–340.
33. Fister I. Post hoc analysis of sport performance with differential evolution / I. Fister, D. Fister, S. Deb, U. Mlakar, J. Brest // Neural Computing and Applications, 2018. – p. 1–10.
34. Fister I. Making up for the deficit in a marathon run / I. Fister, D. Fister, S. Deb, U. Mlakar, J. Brest // International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence, 2017. – p. 11–15.
35. Op De Beéck T. Fatigue Prediction in Outdoor Runners Via Machine Learning and Sensor Fusion / T. Op De Beéck, W. Meer, K. Schütte, B. Vanwanseele, J. Davis // ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018. – p. 606–615.

36. Vales-Alonso J. Ambient Intelligence Systems for Personalized Sport Training / J. Vales-Alonso, P. López-Matencio, F.J. Gonzalez-Castaño, H. Navarro-Hellín, P.J. Baños-Guirao, F.J. Pérez-Martínez, R.P. Martínez-Álvarez, D. González-Jiménez, F. Gil-Castiñeira, R. Duro-Fernández // Sensors, 2010. – № 10. – p. 2359–2385.
37. Buttussi F. MOPET: A context-aware and user-adaptive wearable system for fitness training / F. Buttussi, L. Chittaro // Artif. Intell. Med., 2008. – № 42. –p. 153–163.
38. Brzostowski K. Adaptive decision support system for automatic physical effort plan generation-data-driven approach / K. Brzostowski, J. Drapała, A. Grzech, P. Swiatek // Cybern. Syst., 2013. – № 44. –p. 204–221.
39. Lopez-Matenci P. Ambient intelligence assistant for running sports based on k-NN classifiers / P. Lopez-Matenci, J.V. Alonso, F.J. Gonzalez-Castano, J.L. Sieiro, J.J. Alcaraz // International Conference on Human System Interaction, 2010.– p. 605–611.
40. Pantazopoulos A. Sports & Nutrition Data Science using Gradient Boosting Machines / A. Pantazopoulos, M. Maragoudakis // Hellenic Conference on Artificial Intelligence, 2018. – p. 1–7.
41. Suh M. Machine Learning-Based Adaptive Wireless Interval Training Guidance System / M. Suh, A. Nahapetian, J. Woodbridge, M. Rofouei, M. Sarrafzadeh // Mob. Netw. Appl.,2012. – № 17. – p. 163–177.

ДОДАТОК

```
<?xml version="1.0" encoding="utf-8" ?>
<Project ToolsVersion="3.5" DefaultTargets="Build" xmlns=
"http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
      <Configuration Condition=" '$(Configuration)' == '' ">Debug
      </Configuration>
      <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform
      >
      <ProductVersion>9.0.21022</ProductVersion>
      <SchemaVersion>2.0</SchemaVersion>
      <ProjectGuid>{BDB57DE9-16DB-4776-A42A-8E2ECA8F03E9}</
      ProjectGuid>
      <OutputType>WinExe</OutputType>
      <StartupObject>Teach.TeachUnit</StartupObject>
      <AppDesignerFolder>Properties</AppDesignerFolder>
      <RootNamespace>Teach</RootNamespace>
      <AssemblyName>Teach</AssemblyName>
      <TargetFrameworkVersion>v3.5</TargetFrameworkVersion>
      <FileAlignment>512</FileAlignment>
</PropertyGroup>
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' ==
'Debug|AnyCPU' ">
      <DebugSymbols>true</DebugSymbols>
      <DebugType>full</DebugType>
      <Optimize>false</Optimize>
      <OutputPath>bin\Debug\</OutputPath>
      <DefineConstants>DEBUG;TRACE</DefineConstants>
      <ErrorReport>prompt</ErrorReport>
      <WarningLevel>4</WarningLevel>
</PropertyGroup>
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' ==
'Release|AnyCPU' ">
      <DebugType>pdbonly</DebugType>
      <Optimize>true</Optimize>
      <OutputPath>bin\Release\</OutputPath>
      <DefineConstants>TRACE</DefineConstants>
      <ErrorReport>prompt</ErrorReport>
      <WarningLevel>4</WarningLevel>
</PropertyGroup>
<ItemGroup>
      <Reference Include="System" />
      <Reference Include="System.Core">
          <RequiredTargetFramework>3.5</RequiredTargetFramework>
      </Reference>
      <Reference Include="System.Xml" />
      <Reference Include="System.Windows.Forms" />
</ItemGroup>
<ItemGroup>
      <Compile Include= "Teach.cs">
          <SubType>Code</SubType>
      </Compile>
</ItemGroup>
<Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
</Project>
```

```
using System;
using System.IO;
using System.Windows.Forms;
namespace Teach
{
    public class TeachUnit
    {
        public static int n = 0;
        public static int[, , ] Y = new int[m + 1, n_max + 1, nr +
         1];
        public static int[, , ] BM = new int[m + 1, n_max + 1, nr
        + 1];
        public static double[] VD = new double[n_max + 1];
        public static double[] ND = new double[n_max + 1];
        public static int[, ] EV = new int[m + 1, n_max + 1];
        public static int[] SK = new int[nr + 1];
        public static int[] SKS = new int[Convert.ToInt32(m - 1 *
        nr) + 1];
        public static double[] A = new double[m + 1];
        public static double[] B = new double[m + 1];
        public static double[] D1 = new double[m + 1];
        public static double[] D2 = new double[m + 1];
        public static double[] EM = new double[m + 1];
        public static double[] A_NWS = new double[m + 1];
        public static double[] B_NWS = new double[m + 1];
        public static double[] D1_NWS = new double[m + 1];
        public static double[] D2_NWS = new double[m + 1];
        public static double[] EM_NWS = new double[m + 1];
        public static int[] D_NWS = new int[m + 1];
        public static int[] D = new int[m + 1];
        public static int[] Dt = new int[m + 1];
        public static int i = 0;
        public static int j = 0;
        public static int k = 0;
        public static int an = 0;
        public static int av = 0;
        public static int ii = 0;
        public static System.IO.Stream FT = null;
        public static System.IO.Stream FT_EXM = null;
        public static System.IO.Stream FT2 = null;
        public static string FN = String.Empty;
        public static int tdelta = 0;
        public static bool ifsaveclasses = false;
        public static bool ifshowclasses = false;
        public static double eavg = 0;
        public static double edmax = 0;
        public static bool __bool = false;
        public static int delta = 0;
```

```
public static int delta_opt = 0;
public static int ie = 0;
public static int[] ie_c = new int[n_max + 1];
public const int m = 2;
public const int n_max = 48;
public const int nr = 4500;
public static void LoadClass(int num)
{
    System.IO.Stream fClass;
    Console.Out.Write("FileName for ");
    Console.Out.Write(num);
    Console.Out.Write(" class:(ENTER=");
    Console.Out.Write(num + 1);
    Console.Out.Write(".txt) ");
    FN = Console.In.ReadLine();
    if (FN == "")
    {
        FN = (num + 1).ToString() + ".txt";
    }
    fClass = new FileInfo(FN);
    StreamReader _R_0 = fClass.OpenText();
    for (j = 1; j <= nr; j ++ )
    {
        for (i = 1; i <= n; i ++ )
        {
            Y[num, i, j] = _R_0.Read();
        }
    }
    _R_0.Close();
}

public static void MakeSD2()
{
    for (i = 1; i <= n; i ++ )
    {
        ND[i] = delta;
        VD[i] = 255;
    }
}

public static void MakeSD()
{
    System.IO.Stream fSKD;
    Console.Out.Write("FileName for system of
    tolerances:(ENTER= SKD.txt) ");
    FN = Console.In.ReadLine();
    if (FN == "")
    {
```

```
            FN = "SKD.txt";
        }
        fSKD = new FileInfo(FN);
        StreamReader _R_1 = fSKD.OpenText();
        for (i = 1; i <= n; i ++ )
        {
            ND[i] = _R_1.Read();
        }
        for (i = 1; i <= n; i ++ )
        {
            VD[i] = _R_1.Read();
        }
        _R_1.Close();
    }

    public static void MakeBM()
    {
        for (k = 1; k <= m; k ++ )
        {
            for (i = 1; i <= n; i ++ )
            {
                for (j = 1; j <= nr; j ++ )
                {
                    if ((Y[k, i, j] >= ND[i]) && (Y[k, i, j]
                    <= VD[i]))
                    {
                        BM[k, i, j] = 1;
                    }
                    else
                    {
                        BM[k, i, j] = 0;
                    }
                }
            }
        }
    }

    public static void MakeEV()
    {
        for (k = 1; k <= m; k ++ )
        {
            for (i = 1; i <= n; i ++ )
            {
                EV[k, i] = 1;
            }
        }
    }
```

```
public static void MakeSK(int num)
{
    int kdop;
    int t_sk;
    for (j = 1; j <= nr; j ++ )
    {
        for (kdop = 1; kdop <= m; kdop ++ )
        {
            t_sk = 0;
            for (i = 1; i <= n; i ++ )
            {
                if (ie != i)
                {
                    t_sk = t_sk + Math.Abs(EV[num, i] - BM
                    [kdop, i, j]);
                    if (kdop > num)
                    {
                        SKS[j + nr * (kdop - 2)] = t_sk;
                    }
                    else if (kdop == num)
                    {
                        SK[j] = t_sk;
                    }
                    else
                    {
                        SKS[j + nr * (kdop - 1)] = t_sk;
                    }
                }
            }
        }
    }
}

public double INFK_log_(double k1_log, double k2_log, int
t1_, int t_, ref double t_D1, ref double t_betta)
{
    double result;
    if ((k1_log == 0))
    {
        result = 0;
    }
    else
    {
        result = (k1_log / (k1_log + k2_log)) * (Math.Log(
        k1_log / (k1_log + k2_log)) / Math.Log(2.0));
    }
    return result;
}
```

```
public static double INFK(int t1_, int t_, ref double t_D1
, ref double t_betta)
{
    double result;
    int k1;
    int k2;
    int k3;
    int k4;
    int kdop;
    double t_alfa;
    double t_d2;
    k1 = 0;
    k2 = 0;
    k3 = 0;
    k4 = 0;
    for (j = 1; j <= nr; j ++ )
    {
        if ((SK[j] <= t_) && (SK[j] > t1_))
        {
            k1 ++;
        }
        for (kdop = 1; kdop <= m - 1; kdop ++ )
        {
            if ((SKS[j + nr * (kdop - 1)] <= t_) && (SKS[j
             + nr * (kdop - 1)] > t1_))
            {
                k3 ++;
            }
        }
    }
    k4 = (m - 1) * nr - k3;
    k2 = nr - k1;
    t_D1 = k1 / nr;
    t_betta = k3 / nr;
    if (t_betta > 1)
    {
        t_betta = 1;
    }
    t_alfa = 1 - t_D1;
    t_d2 = 1 - t_betta;
    result = 0.5 * ((t_D1 + t_d2 + 0.01) - (t_alfa +
    t_betta + 0.01)) * Math.Log((t_D1 + t_d2 + 0.01) / (
    t_alfa + t_betta + 0.01)) / Math.Log(2.0);
    return result;
}

public static void SaveClass(int num)
{
```

```
        FT.WriteLine("Class ");
        FT.WriteLine(num);
        FT.WriteLine("BM[");
        FT.WriteLine(num);
        FT.WriteLine(']');
        for (i = 1; i <= n; i ++ )
        {
            for (j = 1; j <= nr; j ++ )
            {
                FT.Write(BM[num, i, j]);
                FT.Write((char)(9));
            }
            ;
        }
        FT.WriteLine("EV[");
        FT.WriteLine(num);
        FT.WriteLine(']');
        for (i = 1; i <= n; i ++ )
        {
            FT.Write(EV[num, i]);
            FT.Write((char)(9));
        }
        ;
        FT.WriteLine("dmin");
        FT.WriteLine((char)(9));
        FT.WriteLine("dmax");
        FT.WriteLine((char)(9));
        FT.WriteLine('E');
        FT.WriteLine((char)(9));
        FT.WriteLine("D1");
        FT.WriteLine((char)(9));
        FT.WriteLine("Betta");
        FT.WriteLine((char)(9));
        FT.WriteLine("Alfa");
        FT.WriteLine((char)(9));
        FT.WriteLine("D2");
    }

    public static void MakeDO()
    {
        double te;
        double tD1;
        double tbetta;
        int t;
        int t1;
        int ti;
        FT_EXM = new FileInfo("REZ_" + delta.ToString() +
        ".TXT");
```

```
StreamWriter _W_0 = FT_EXM.CreateText();
if (ifsaveclasses)
{
    for (ti = 1; ti <= n; ti ++ )
    {
        _W_0.Write(VD[ti]);
        _W_0.Write(' ');
        _W_0.Write(ND[ti]);
        _W_0.Write((char)(9));
    }
    ;
}
for (k = 1; k <= m; k ++ )
{
    if (ifsaveclasses)
    {
        FT = new FileInfo("REZ" + k.ToString() + delta
        .ToString() + ".TXT");
        StreamWriter _W_1 = FT.CreateText();
        SaveClass(k);
    }
    EM[k] = 0;
    EM_NWS[k] = 0;
    D[k] = 0;
    MakeSK(k);
    for (t1 = 0; t1 <= n; t1 ++ )
    {
        for (t = 0; t <= n; t ++ )
        {
            te = INFK(t1, t, ref tD1, ref tbetta);
            if (ifsaveclasses)
            {
                _W_1.WriteLine(t1);
                _W_1.WriteLine((char)(9));
                _W_1.WriteLine(t);
                _W_1.WriteLine((char)(9));
                _W_1.WriteLine(te);
                _W_1.WriteLine((char)(9));
                _W_1.WriteLine(tD1);
                _W_1.WriteLine((char)(9));
                _W_1.WriteLine(tbetta);
                _W_1.WriteLine((char)(9));
                _W_1.WriteLine(1 - tD1);
                _W_1.WriteLine((char)(9));
                _W_1.WriteLine(1 - tbetta);
            }
            if (te > EM_NWS[k])
            {
```

```
                    EM_NWS[k] = te;
                    D_NWS[k] = t;
                    D1_NWS[k] = tD1;
                    B_NWS[k] = tbetta;
                }
                // Додати додаткові умови щодо робочої
                області
                if ((tD1 >= 0.5) && (tbetta < 0.5))
                {
                    if (te > EM[k])
                    {
                        EM[k] = te;
                        Dt[k] = t1;
                        D[k] = t;
                        D1[k] = tD1;
                        B[k] = tbetta;
                    }
                }
            }
        }
        if (ifshowclasses)
        {
            Console.Out.WriteLine("Class ");
            Console.Out.WriteLine(k);
            Console.Out.WriteLine((char)(9));
            Console.Out.WriteLine("Em= ");
            Console.Out.WriteLine(EM[k]);
            Console.Out.WriteLine((char)(9));
            Console.Out.WriteLine("do1= ");
            Console.Out.WriteLine(Dt[k]);
            Console.Out.WriteLine((char)(9));
            Console.Out.WriteLine("do= ");
            Console.Out.WriteLine(D[k]);
            Console.Out.WriteLine((char)(9));
            Console.Out.WriteLine("D1= ");
            Console.Out.WriteLine(D1[k]);
            Console.Out.WriteLine((char)(9));
            Console.Out.WriteLine("Betta= ");
            Console.Out.WriteLine(B[k]);
        }
        if (ifsaveclasses)
        {
            _W_1.WriteLine("Class ");
            _W_1.WriteLine(k);
            _W_1.WriteLine((char)(9));
            _W_1.WriteLine("Em= ");
            _W_1.WriteLine(EM[k]);
             W 1.WriteLine((char)(9));
```

```
                _W_1.WriteLine("do1= ");
                _W_1.WriteLine(Dt[k]);
                _W_1.WriteLine((char)(9));
                _W_1.WriteLine("do= ");
                _W_1.WriteLine(D[k]);
                _W_1.WriteLine((char)(9));
                _W_1.WriteLine("D1= ");
                _W_1.WriteLine(D1[k]);
                _W_1.WriteLine((char)(9));
                _W_1.WriteLine("Betta= ");
                _W_1.WriteLine(B[k]);
                _W_0.WriteLine(k);
                for (ti = 1; ti <= n; ti ++ )
                {
                    _W_0.Write(EV[k, ti]);
                    _W_0.Write(' ');
                }
                ;
                _W_0.WriteLine(D[k]);
                _W_1.Close();
            }
        }
        _W_0.Close();
    }

    public static void Opt_delta()
    {
        ifshowclasses = false;
        ifsaveclasses = true;
        delta_opt = 0;
        edmax = 0;
        for (delta = 0; delta <= 150; delta ++ )
        {
            MakeSD2();
            MakeBM();
            MakeEV();
            MakeDO();
            eavg = 0;
            __bool = true;
            for (k = 1; k <= m; k ++ )
            {
                __bool = __bool && (D1[k] > 0.5) && (B[k] <
                0.5);
                eavg = eavg + EM[k];
            }
            eavg = eavg / m;
            if ((edmax < eavg) && __bool)
            {
```

```
                edmax = eavg;
                delta_opt = delta;
            }
        }
        delta = delta_opt;
        ifshowclasses = true;
        MakeSD2();
        MakeBM();
        MakeEV();
        MakeDO();
    }

    public static int Find_I()
    {
        int result;
        int i_out;
        int i1_out;
        double e_old;
        e_old = 0;
        for (i_out = 1; i_out <= n; i_out ++ )
        {
            Console.Out.Write(Math.Truncate(i_out / 10));
            ie = i_out;
            Opt_delta();
            if (edmax >= e_old)
            {
                e_old = edmax;
                i1_out = i_out;
            }
        }
        Console.Out.WriteLine(' ');
        result = i1_out;
        return result;
    }

    public static void change(int i_ch, int j_ch)
    {
        int k_ch;
        int i1_ch;
        int i3_ch;
        int y_ch;
        int e_ch;
        double sd_ch;
        if (i_ch != j_ch)
        {
            for (k_ch = 0; k_ch <= m - 1; k_ch ++ )
            {
                for (i1_ch = 0; i1_ch <= n - 1; i1_ch ++ )
```

```
                {
                    y_ch = Y[k_ch, i_ch, i1_ch];
                    Y[k_ch, i_ch, i1_ch] = Y[k_ch, j_ch, i1_ch
                    ];
                    Y[k_ch, j_ch, i1_ch] = y_ch;
                }
            }
            sd_ch = ND[i_ch];
            ND[i_ch] = ND[j_ch];
            ND[j_ch] = sd_ch;
            i3_ch = ie_c[i_ch];
            ie_c[i_ch] = ie_c[j_ch];
            ie_c[j_ch] = i3_ch;
        }
    }

    public static void Opt_SP()
    {
        int i_sp;
        int i1_sp;
        int i_find;
        double e_max_sp;
        System.IO.Stream file_Opt_SP;
        file_Opt_SP = new FileInfo("Rez_Opt_SP.txt");
        StreamWriter _W_2 = file_Opt_SP.CreateText();
        _W_2.WriteLine("N PO Eavg delta");
        n = n_max;
        for (i_sp = 1; i_sp <= n_max; i_sp ++ )
        {
            ie_c[i_sp] = i_sp;
        }
        ie = 0;
        Opt_delta();
        __bool = true;
        for (k = 1; k <= m; k ++ )
        {
            __bool = __bool && (D1[k] > 0.5) && (B[k] < 0.5);
        }
        _W_2.WriteLine(n);
        _W_2.WriteLine(__bool);
        _W_2.WriteLine(edmax);
        _W_2.WriteLine(delta_opt);
        for (i_sp = n_max; i_sp >= 1; i_sp-- )
        {
            n = i_sp;
            i_find = Find_I();
            change(i_find, i_sp);
            ie = 0;
```

```
            __bool = true;
            for (k = 1; k <= m; k ++ )
            {
                __bool = __bool && (D1[k] > 0.5) && (B[k] <
                0.5);
            }
            Console.Out.Write(i_find);
            Console.Out.Write(" - ");
            Console.Out.Write(i_sp);
            Console.Out.WriteLine(' ');
            Console.Out.WriteLine(n);
            Console.Out.WriteLine(' ');
            Console.Out.WriteLine(__bool);
            Console.Out.WriteLine(' ');
            Console.Out.WriteLine(edmax);
            Console.Out.WriteLine(' ');
            Console.Out.WriteLine(delta_opt);
            Console.Out.WriteLine(' ');
            _W_2.Write(n);
            _W_2.Write(' ');
            _W_2.Write(__bool);
            _W_2.Write(' ');
            _W_2.Write(edmax);
            _W_2.Write(' ');
            _W_2.Write(delta_opt);
            _W_2.Write(' ');
            for (i1_sp = 1; i1_sp <= n; i1_sp ++ )
            {
                _W_2.Write(ie_c[i1_sp]);
                _W_2.Write(' ');
            }
            ;
        }
        _W_2.Close();
    }

    [STAThread]
    public static void Main(string[] args)
    {
        FT2 = new FileInfo("REZ_all.TXT");
        StreamWriter _W_3 = FT2.CreateText();
        ifsaveclasses = true;
        ifshowclasses = false;
        Console.Out.WriteLine("LOAD");
        n = n_max;
        for (k = 1; k <= m; k ++ )
        {
            LoadClass(k);
```

```
}
Console.Out.WriteLine("LOAD..Done");
Console.Out.WriteLine("TEACH");
delta_opt = 0;
edmax = 0;
for (delta = 0; delta <= 150; delta ++ )
{
    MakeSD2();
    MakeBM();
    MakeEV();
    MakeDO();
    eavg = 0;
    __bool = true;
    for (k = 1; k <= m; k ++ )
    {
        __bool = __bool && (D1[k] > 0.5) && (B[k] <
        0.5);
        eavg = eavg + EM[k];
    }
    eavg = eavg / m;
    if ((edmax < eavg) && __bool)
    {
        edmax = eavg;
        delta_opt = delta;
    }
    _W_3.WriteLine("delta= ");
    _W_3.WriteLine(delta);
    _W_3.WriteLine(' ');
    _W_3.WriteLine(eavg);
    _W_3.WriteLine(' ');
    _W_3.WriteLine(__bool);
}
Console.Out.WriteLine("TEACH..Done");
ifshowclasses = true;
delta = delta_opt;
Console.Out.WriteLine("delta opt =");
Console.Out.WriteLine(delta);
MakeSD2();
MakeBM();
Console.Out.WriteLine("BM..Done");
MakeEV();
Console.Out.WriteLine("EV..Done");
MakeDO();
Console.Out.WriteLine("EV..Done");
ifshowclasses = true;
ifsaveclasses = true;
MakeSD();
MakeBM();
```

```
            MakeEV();
            MakeDO();
            Console.In.ReadLine();
            _W_3.Close();
            Application.Run();
        }

    } // end Teach

}
```