УДК 004.421:004.5
DOI https://doi.org/10.32782/2663-5941/2025.6.2/50

Турчак В.Р.
Національний університет «Львівська політехніка»

МОДЕЛЬ ІНФОРМАЦІЙНОЇ СИСТЕМИ БАГАТОКРИТЕРІАЛЬНОГО СОРТУВАННЯ ДАНИХ З ЇХ ДИНАМІЧНО-НАСЛІДКОВИМ ОБМЕЖЕННЯМ ПЕРЕМІЩЕННЯ

Інформаційні системи сортування структурованих даних є ключовими для аналітики, логістики, освітнього моніторингу та технічної безпеки. Їх ефективність визначається не лише точністю алгоритмів, а й узгодженістю архітектури етапів обробки – від імпорту та нормалізації до контролю впливу критеріїв і пояснюваності результатів. Метою роботи є побудова моделі інформаційної системи багатокритеріального сортування, що зберігає частковий початковий порядок за рахунок керованого локального переміщення елементів і забезпечує відтворюваність кожного кроку зміни рангу.

Розроблена модель охоплює повний цикл опрацювання табличних даних: перехід від початкового стану T_1 до нормалізованого T_1', специфікацію множини критеріїв T_2, побудову компараторів із підтримкою нечіткого текстового збігу на основі нормалізованої відстані Левенштейна та формування впорядкованого результату T_3 із журналом стабільності. Архітектура реалізована як набір взаємодіючих модулів (імпорт/попередня обробка; менеджер критеріїв; модуль порівняння; рушій сортування; відображення й експорт; журнал і метрики), для яких чітко визначено входи/виходи та інваріанти поведінки.

Алгоритмічне ядро ґрунтується не на повному пересортуванні масиву записів, а на поетапному частковому "підтягуванні" елементів відповідно до пріоритету критерію. Для критерію з пріоритетом pr_j вводиться обмеження переміщення $R_j = N/pr_j$ (де N – кількість рядків), що лімітує максимальну дальність локального зсуву за одну ітерацію. На кожному кроці система обирає локально найкращий елемент у допустимому вікні та підтягує його вгору не далі ніж на R_j позицій, виконуючи лише послідовні суміжні обміни (adjacent-only), а у випадку рівності значень застосовується стабільне розв'язання рівностей (tie-break) за початковим порядком. Така інформаційна технологія забезпечує контрольовану зміну ранжування без руйнування глобального порядку і не дозволяє критеріям нижчого пріоритету повністю переважати критерії з вищим пріоритетом.

Система формує впорядкований список T_3, супроводжений детальним журналом кроків та узагальненими метриками стабільності (зокрема Kendall's τ та середній зсув позиції). Це підвищує пояснюваність і відтворюваність результатів у прикладних задачах, а також забезпечує можливість кількісного оцінювання масштабу втручання в початковий порядок і проводити аудит прийнятих рішень на рівні окремих операцій.

Наукова новизна полягає у створенні моделі інформаційної системи багатокритеріального сортування як послідовності взаємодіючих модулів із жорсткими інваріантами (обмеження на переміщення, режим покрокової заміни, стабільний початковий порядок, пороговий нечіткий збіг), що мають узгоджені відповідники в моделях, структурі даних і програмних інтерфейсах. На відміну від існуючих моделей, створена модель поєднує керовану локальність впливу критеріїв із формалізованим журналюванням процесу, що забезпечує як стабільність часткового порядку, так і прозорий аудит кроків сортування.

Практична значущість полягає в тому, що запропонована модель орієнтована на використання у вебзастосунках без серверної постопрацювання, підтримує змішані типи полів (числові й текстові), забезпечує прозорий контроль сили впливу кожного критерію, передбачає технологію експорту результатів і журналу для подальшого аналізу та може інтегруватися в робочі процеси електронної комерції, технічного моніторингу, освітньої аналітики й управління ресурсами.

Ключові слова: *інформаційна система, багатокритеріальне сортування, модульна архітектура, динамічно-наслідкове обмеження переміщення, нечіткий збіг, стабільність порядку, структуровані дані.*

Постановка проблеми. У прикладних системах роботи з таблицями багатокритеріальне впорядкування зазвичай реалізують як послідовні повні пересортування або як лексикографічне сортування без чітко заданої межі впливу кожного критерію. У такій постановці навіть незначні зміни ваг чи додавання нового критерію можуть повністю зруйнувати вже сформований частковий порядок, а кроки зміни рангу не фіксуються і не підлягають подальшому аудиту. Для гібридних таблиць з різнотипними полями (числовими, текстовими тощо) додатково потрібні уніфіковані правила порівняння з підтримкою порогового нечіткого збігу та стабільне розв'язання рівностей (tie-break) зі збереженням початкового відносного порядку рядків. В умовах сталого щорічного зростання обсягів даних такі проблеми призводять до відчутних втрат ефективності в освітній аналітиці, логістиці та суміжних сферах, де помилки й непрозорість сортування безпосередньо впливають на якість прийнятих рішень.

За цих умов потрібна клієнтська модель, яка зберігає частковий початковий порядок не за рахунок повної перебудови масиву, а через кероване обмеження локального впливу кожного критерію залежно від його пріоритету, виконує переміщення елементів лише як послідовні суміжні обміни, застосовує стабільний механізм tie-break за початковим індексом, уніфікує порівняння для числових і текстових полів із підтримкою порогового нечіткого збігу та веде формальний журнал кроків сортування разом із метриками стабільності для подальшого аудиту. Така модель має бути реалізована повністю на боці клієнта (у браузері), спиратися на чітко визначені інваріанти й забезпечувати можливість відтворення кожного кроку зміни рангу без залучення серверної постобробки.

Об'єкт дослідження – процес багатокритеріального впорядкування табличних даних. Предмет дослідження – архітектурна модель клієнтської інформаційної системи з локально обмеженим впливом критеріїв, яка забезпечує відтворюваність кроків сортування та пояснюваність отриманих результатів у прикладних сценаріях використання.

Аналіз останніх досліджень і публікацій. Клієнтська обробка табличних структур. За останні роки зросла роль безсерверних вебрішень, де імпорт, нормалізація, фільтрація та сортування великих таблиць виконуються безпосередньо у браузері. Практичні інструменти на кшталт SheetJS/XLSX, PapaParse та Handsontable забезпечують стабільний і протестований набір технологій для читання XLSX/CSV/JSON, відображення грідів і базових операцій сортування/експорту [1–3]. Водночас ці засоби, як правило, реалізують або повне пересортування, або лексикографічне впорядкування без формалізованого контролю локальної сили впливу критерію та без покрокового журналу зміни порядку; стабільність відносного порядку і пояснюваність кроків розв'язуються частково або лишаються поза моделлю користувацького інтерфейсу. Дослідження з досвіду взаємодії користувача (UX) з табличними інтерфейсами підтверджують потребу в адаптивних перетвореннях і чітких поясненнях, але не дають формалізованого механізму керованих локальних перестановок у клієнті [4, с. 2–3; 5, с. 5–6].

MCDA: робастність ранжувань і (не)компенсаторність. Сучасні роботи з багатокритеріального аналізу рішень (TOPSIS, варіанти вагових схем) демонструють істотну чутливість ранжувань до нормалізації, вибору ваг та порогів прийнятності; дедалі більшої уваги набувають аналіз чутливості, сценарії невизначеності та обмеження компенсаторності між критеріями [6, с. 33–34; 7, с. 3–6; 8, с. 2–3]. Фактично мета зсувається від пошуку "єдиного правильного" порядку до контролю амплітуди допустимих змін позиції за варіювання параметрів. Проте більшість цих моделей залишаються на рівні глобальних перетворень і не задають механізму локального підняття елемента з формальним радіусом впливу у виконуваному коді веб-клієнта.

Outranking-процедури. Сімейства ELECTRE/PROMETHEE надають апарати порогів переваги/індиферентності та контроль компенсаторності, а також сильну пояснюваність "чому A краще за B" [9, Art. 103116; 10, с. 1–2]. Однак їх параметризація і складність перенесення у змішані (число-текстові) дані роблять інтеграцію в типові браузерні застосунки нетривіальною; відсутня проста операційна інтерпретація у вигляді покрокових суміжних перестановок із явним радіусом локального зсуву, придатним для аудиту в користувацькому інтерфейсі (UI).

Лексикографічні та алгоритмічні засади стабільності. Лексикографічні схеми природно пріоритезують критерії й сумісні зі стабільними алгоритмами сортування, що зберігають відносний порядок рівних елементів. Недавні роботи з лексикографічної оптимізації акцентують саме на стабільності та структурі обмежень, але не встановлюють кількісної межі локального зсуву запису в реалізації веб-клієнта [11, с. 1–4]. Класичні

алгоритмічні джерела забезпечують технічну базу для стабільних процедур, однак питання керованої локальності (радіуса підняття) і формального журналювання кроків лишається поза їхнім фокусом [12, с. 75–80].

У попередніх працях автора було запропоновано механізм контрольованого "підтягування" елемента в межах локального радіуса з виконанням лише послідовних суміжних обмінів та зі стабільним розв'язанням рівностей за початковим індексом, формалізовано структуру критеріїв, а також покрокову взаємодію користувача з імпортом, заданням критеріїв, впорядкуванням і експортом із прив'язкою до викликів у коді [13, с. 20–22; 14, с. 175–176; 15, с. 14–15]. Водночас у відкритих джерелах не було знайдено цілісної клієнтської специфікації, яка одночасно вводить кількісну межу локального впливу критерію, гарантує режим суміжних обмінів зі стабільним розв'язанням рівностей, уніфікує порівняння числових і текстових полів із пороговим нечітким збігом, а також фіксує формалізований журнал кроків і метрики стабільності в архітектурі, орієнтованій на користувацький інтерфейс. Саме заповнення цієї прогалини і становить предмет подальшого викладу.

Постановка завдання. Метою статті є побудова моделі клієнтської інформаційної системи багатокритеріального сортування структурованих даних з їх динамічно-наслідковим обмеженням переміщення елементів, яка виконує впорядкування лише послідовними суміжними обмінами, застосовує стабільне розв'язання рівностей, підтримує пороговий нечіткий збіг для текстових полів і веде формалізований журнал кроків.

Для досягнення поставленої мети необхідно розв'язати такі завдання:

1. Розробити інформаційну технологію керованого впорядкування, що реалізує *R*-обмежене локальне переміщення елементів у режимі лише послідовних суміжних обмінів (adjacent-only) зі стабільним розв'язанням рівностей за початковим індексом і підтримкою порогового нечіткого збігу для числових і текстових атрибутів без повного пересортування масиву.

2. Сформувати модульну архітектуру клієнтської системи, яка охоплює етапи імпорту даних, очищення, задання критеріїв, компарації, впорядкування, журналювання, експорту та пояснення результатів у браузерному середовищі без серверної постобробки.

3. Побудувати формалізований журнал виконання та систему метрик стабільності (коефіцієнт рангової кореляції Кендала, середній зсув позиції, частка значних зсувів), що забезпечують відтворюваність, пояснюваність і аудит результатів багатокритеріального сортування.

Виклад основного матеріалу. Модель клієнтської інформаційної системи багатокритеріального сортування структурованих даних з їх динамічно-наслідковим обмеженням переміщення розглядаємо як керований конвеєр перетворень показаний на рис. 1, де вхідна таблиця T_1 поступово доводиться до стану, придатного для надійного порівняння, а далі – впорядковується з контрольованою локальністю. Для цього ми спочатку уніфікуємо дані (отримуємо T_1'), потім специфікуємо критерії T_2, на їх основі будуємо компаратори $\mathrm{cmp}(T_2[j])$, і вже на завершальному етапі застосовуємо *R*-обмежене впорядкування з детальним журналюванням, формуючи T_3. Таким чином, кожен наступний модуль працює на чітко визначеному, стандартизованому вхідному потоці й повертає прозорий, відтворюваний результат.

Спочатку користувач через інтерфейс *B*0 (UI) запускає роботу системи, генеруючи події з множини Σ: *ev_import*, *ev_define_criteria*, *ev_sort_all* / *ev_next_step*, *ev_export*, *ev_error*. Тобто саме користувач своїми діями («імпортувати файл», «задати критерії», «відсортувати», «експортувати», «повідомити про помилку») послідовно переводить систему з одного кроку на інший. На цьому рівні ми свідомо розділяємо дві ролі. Інтерфейс *B*0 відповідає за прийом дій користувача, збір параметрів через форми, перевірку коректності введення (чи заповнені всі обов'язкові поля, чи коректно задані пріоритети, пороги тощо) і за показ повідомлень та результатів на екрані. Натомість обчислювальні модулі *B*1–*B*7 виконують «важку» роботу: імпорт із файлів, нормалізацію даних, побудову критеріїв, власне сортування, ведення журналу кроків, розрахунок метрик стабільності та експорт/збереження результатів. Інтерфейс сам нічого не сортує і не вирішує, кого поставити вище в таблиці: він лише передає дані у відповідні модулі й показує те, що вони обчислили. На екрані користувач бачить очищену таблицю T_1', відсортований результат T_3 та метрики стабільності, розраховані на основі журналу кроків. Такий поділ ролей не дозволяє «приховати» критичну логіку всередині UI і спрощує аудит: усі суттєві перетворення зосереджені в чітко визначених модулях, де їх можна окремо перевірити й відтворити.

Модуль *B*1 реалізує імпорт вхідних даних із файлів XLSX, CSV, JSON. Як показано на рисунку 1, *B*1 отримує команду від *B*0, читає файл, пере-

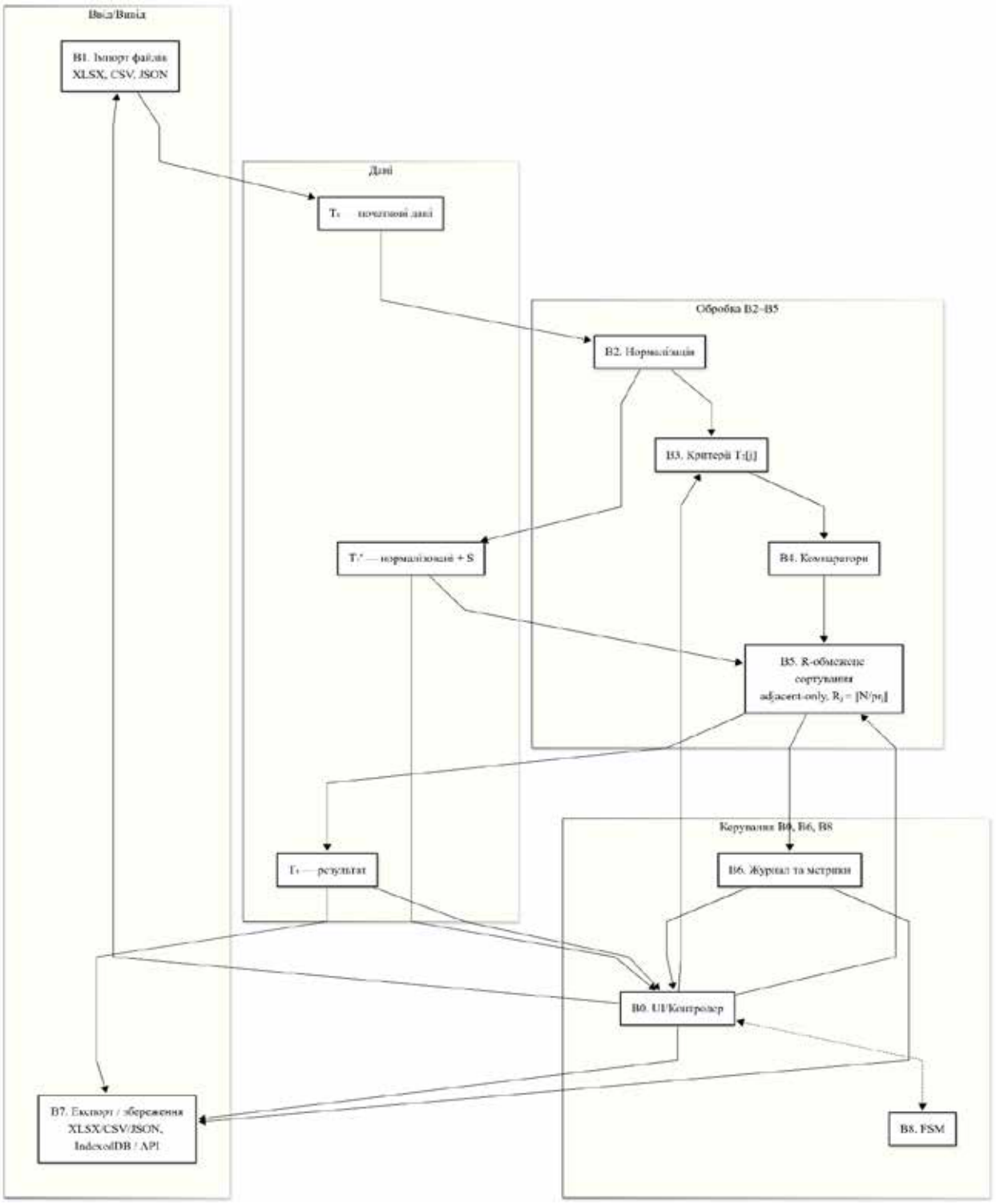

Рис. 1. Модель інформаційної системи багатокритеріального сортування даних з їх динамічно-наслідковим обмеженням переміщення

творює його до внутрішнього табличного представлення і формує початкову таблицю T_1, яка далі передається в модуль нормалізації $B2$. Отже, $B1$ відповідає за коректне та відтворюване завантаження сирих структурованих даних у систему.

Далі, оскільки будь-яке порівняння дуже чутливе до «сміття» у вхідних даних (зайві пробіли, різний регістр, неоднорідні типи, випадкові пропуски, дивні символи), першим кроком ми нормалізуємо вхідну таблицю в модулі $B2$. Для цього застосовується функція $f_1(T_1 \rightarrow T_1')$, яка по суті є фільтром очищення. Вона послідовно прибирає

зайві пробіли на початку та в кінці рядків (*trim*), щоб « Sensor » і «Sensor» не вважалися різними значеннями, зводить текст до єдиного регістру (нижній), щоб «Detector» і «detector» порівнювалися однаково, і за потреби усуває діакритики (наприклад, перетворює «é» на «e»), щоб незначні відмінності написання не ламали збіг. Далі f_1 приводить кожну колонку до одного з узгоджених типів із фіксованого набору {*number*, *text*, *date*, *bool*}, тобто перетворює те, що було рядком «10», у число 10, дати приводить до єдиного формату, логічні значення типу «так/ні» – до *bool*. Окремо фіксується політика обробки пропусків і *NaN*: за замовчуванням усі такі значення при порівнянні стабільно відносяться в кінець, щоб пропуски автоматично не «випереджали» заповнені рядки.

Паралельно з очищенням ми формуємо схему *S*, яку можна розуміти як «паспорт» таблиці. Для кожної колонки схема *S* зберігає її тип (число, текст, дата, *bool*), межі значень (*min*/*max*), а також параметри для нормування (масштаби, коефіцієнти, що дозволяють перевести значення у відносний інтервал, наприклад, [0;1]). Наприклад, для числової колонки «Ціна» ми запам'ятовуємо мінімальну та максимальну ціну, а для дати – найранішу і найпізнішу дату в цій колонці. Схема *S* формується автоматично, поки ми проходимо по рядках і очищуємо значення. Завдяки цьому T_1' стає канонічним, узгодженим за типами та форматами входом для наступних етапів, тобто всі подальші модулі працюють вже не з «як було в Excel», а з очищеною та однорідною таблицею. Схема *S* дає правдиві межі й типи колонок, на які ми далі спираємося при налаштуванні критеріїв та реалізації компараторів, не вгадуючи ці параметри щоразу заново.

Коли дані уніфіковано, у модулі *B*3 ми переходимо до задання критеріїв сортування. На цьому етапі формуємо множину $T_2 = \{T_2[j]\}, j = 1..m$, де кожен елемент $T_2[j]$ має структуру $T_2[j] = (c_j, w_j, d_j, f_j, pr_j, \tau_j)$.

Тут c_j – це цільовий атрибут, тобто колонка, на яку посилається критерій (індекс або ім'я колонки згідно зі схемою *S*). Параметр $w_j \geq 0$ – вага критерію (якщо використовується вагова схема), яка показує відносну важливість цього критерію у групових або агрегованих оцінках. Параметр $d_j \in$ {*min*, *max*, *asc*, *desc*, *match*} задає режим порівняння: *asc*/*desc* означають стандартне зростання/спадання, а *min*/*max* розглядаються як синоніми *asc*/*desc* після нормування (*min* – це «краще менше», *max* – «краще більше»). Режим *match* означає, що критерій працює не як звичайне «більше-менше», а як перевірка на збіг із деяким опорним значенням. Це опорне значення задається параметром f_j – шаблоном або еталоном, за яким ми шукаємо «схожі» рядки. Пріоритет pr_j визначає порядок важливості критеріїв: 1 – найвищий пріоритет; значення пріоритетів не повторюються й утворюють безперервну послідовність (тобто без пропусків). Нарешті, $\tau_j \in [0;1]$ – поріг нечіткого збігу, який використовується для текстових порівнянь: він визначає, яку мінімальну «схожість» треба досягти, щоб вважати рядок таким, що задовольняє критерію *match*.

На вході до *B*3 ми не просто зберігаємо ці параметри, а суворо перевіряємо їх на валідність. По-перше, цільові атрибути c_j мають бути унікальними, тобто кожен критерій працює з конкретною колонкою, і ми не дублюємо критерії для однієї й тієї ж колонки без потреби. По-друге, пріоритети pr_j мають бути без пропусків і дублювання, щоб утворювати чіткий порядок: 1, 2, 3, …, *m* без повторів і «дір». По-третє, режим d_j повинен бути узгоджений із типом колонки в схемі *S*: наприклад, режим *match* не застосовується до чисто числових полів, якщо для них не визначено текстове представлення, а режими *min*/*max* використовуються там, де є сенс говорити про «більше/менше». По-четверте, поріг τ_j має лежати в межах [0;1], тобто бути коректним коефіцієнтом, а не довільним числом. У результаті специфікація критеріїв T_2 перетворюється на формальний і перевірений вхідний артефакт для наступних етапів: саме на його основі будуть побудовані компаратори (реальні функції порівняння) і організований журнал кроків сортування.

У модулі *B*4 ми будуємо компаратори, тобто функції порівняння, які вже безпосередньо застосовуються до рядків у таблиці під час сортування. На цьому кроці абстрактні записи $T_2[j]$ перетворюються на операційні процедури $cmp(T_2[j])$. Для числових колонок ми спочатку нормуємо значення на основі параметрів із *S* (наприклад, приводимо їх до інтервалу [0;1], використовуючи *min*/*max* цієї колонки, щоб різні шкали – гривні, відсотки, метри – стали порівнюваними), а потім застосовуємо режими *asc*/*desc* (або їхні синоніми min/max). При цьому ми забезпечуємо, щоб *NaN* і порожні значення стабільно опинялися наприкінці, не змінюючи відносний порядок валідних елементів за початковим індексом *origIndex*. Це означає, що «порожні» рядки не будуть випадково «підстрибувати» вгору і не зламають логіку сортування.

Для текстових полів компаратор працює інакше. Ми обчислюємо нормалізовану відстань Левенш-

тейна *d*_norm ∈ [0;1] між поточним значенням у рядку та опорним шаблоном f_j. Відстань Левенштейна показує, скільки редагувань (вставок, видалень, замін) потрібно, щоб одне слово перетворити на інше; нормалізована версія приводить ці значення до інтервалу [0;1]. Потім ми визначаємо міру збігу *matchScore* = 1 − *d_norm*: чим ближчі рядки, тим більший matchScore. Далі застосовуємо порогове правило $matchScore \geq \tau_j$: тобто ті рядки, де схожість із шаблоном перевищує або дорівнює порогу τ_j, вважаються такими, що «добре збігаються», і формують «клас переваги» за цим критерієм. Так ми отримуємо нечіткий, але контрольований збіг тексту: невеликі помилки чи варіанти написання не вибивають записи з вибірки.

Щоб не втрачати стабільність сортування, у випадку рівності значень за всіма параметрами компаратор щоразу застосовує *tie-break* за первісним індексом *origIndex*. Це означає, що якщо два рядки для критерію виявилися однаково «хорошими», їхній взаємний порядок залишається таким самим, як у вихідній таблиці. Така стабільність важлива, тому що на наступному етапі ми працюємо з локальними підняттями в межах «вікна» R_j, і будь-яка неточність або випадкова перестановка рівних елементів може породити зайві, погано пояснювані переміщення. З огляду на обчислювальні витрати, ми додатково кешуємо результати нормалізації рядка *normStr*(*x*) (наприклад, уже обрізаний і приведений до нижнього регістру текст) та значення відстані Левенштейна *Lev*(*x*, f_j) для фіксованого f_j. Якщо однаковий текст порівнюється з одним і тим самим шаблоном багато разів, ми не перерахуємо цю відстань щоразу, а візьмемо уже збережене значення. Це суттєво зменшує навантаження без зміни логіки порівняння.

Озброївшись компараторами, у модулі *B*5 ми запускаємо рушій сортування. Його ключова властивість – контрольована локальність впливу кожного критерію. Для критерію з пріоритетом pr_j ми задаємо радіус впливу $R_j = N / pr_j$, де $N = |T_1'|$ – кількість рядків у нормалізованій таблиці. Це означає, що для старших критеріїв (малий pr_j, наприклад 1) радіус R_j більший, і такі критерії можуть істотно змінювати порядок, «бачачи» майже весь список. Для молодших критеріїв (великий pr_j, наприклад 3, 4, 5) R_j менший, і їхній вплив обмежено локальними перестановками, які не руйнують структуру, сформовану старшими. Таким чином, пріоритети прямо пов'язані з «силою» втручання в початковий порядок.

Алгоритм роботи можна описати так. Для поточної позиції і ми розглядаємо підмасив [*i*, $min(i + R_j, N - 1)$] – це «вікно», в межах якого даний критерій має право шукати кращий кандидат для позиції і. У цьому вікні ми шукаємо «найкращий» елемент за компаратором $cmp(T_2[j])$ і визначаємо його індекс *bestIndex*. Далі обчислюємо *steps* = *bestIndex* − *i*, тобто відстань до цього елемента у вікні, *i move* = *min*(*steps*, R_j), тобто фактичний дозволений зсув угору. Після цього зсуваємо обраний елемент вгору на *move* позицій за допомогою послідовних суміжних обмінів (adjacent-only), тобто міняємо його місцями з сусідніми рядками по одному кроку, поки він не досягне нової позиції. Якщо *steps* > R_j, ми не дозволяємо елементу стрибнути на всю відстань, а виконуємо лише часткове підтягування (на відстань R_j), завдяки чому й досягається бажана локальність: критерій не може «витягнути» елемент надто високо за один прохід.

Проходи виконуються в порядку зростання пріоритету: спочатку критерій із pr_j = 1, потім із pr_j = 2 і т. д. Таким чином, спочатку формується «каркас» порядку за найважливішими критеріями, які мають більший радіус впливу, а молодші критерії з меншими R_j можуть лише локально уточнювати цей порядок, не перекриваючи і не ламаючи результат старших. У поєднанні зі стабільним tie-break це забезпечує керовані, логічно пояснювані перестановки, де кожен крок має обмежений радіус дії й зрозуміле пояснення.

Щоб забезпечити відтворюваність усіх кроків, модуль *B*6 протоколює кожну операцію сортування. Для цього ми записуємо структуру (*id*, *P* = pr_j, *i*, R_j, *steps*, *move*, *swaps*), де *id* (*origIndex*) – незмінний ідентифікатор рядка, який не змінюється протягом усього процесу, *P* – активний пріоритет, за яким зараз виконується сортування, *i* – позиція опори, для якої ми шукаємо кращий елемент, R_j – поточний радіус впливу, *steps* – відстань до знайденого «найкращого» елемента у вікні, *move* – фактичний зсув угору, який ми дозволили, *swaps* – кількість виконаних суміжних обмінів (тобто скільки разів елемент поміняся місцями з сусідом). Після завершення проходів ми маємо повний трасувальний журнал: для кожного кроку чітко видно, хто, куди й чому змістився.

На основі цього журналу ми обчислюємо метрики стабільності. Коефіцієнт *Kendall's τ* показує, наскільки кінцевий порядок узгоджений із початковим: значення, близькі до 1, означають, що порядок мало змінився; малі або від'ємні значення означають суттєве перетасування. Середній модуль зсуву позиції *avgShift* обчислюється як середня абсолютна різниця між початковою

та фінальною позиціями рядків: він показує, на скільки позицій у середньому «переїхали» елементи. Частка великих зсувів *share*($\Delta > k$) відображає долю рядків, які змістилися більш ніж на заданий поріг k, тобто дає уявлення, чи були «радикальні» перестановки для окремих елементів. Таким чином, ми отримуємо не тільки впорядкований результат T_3, але й кількісну оцінку того, наскільки сильно система втручалася в початковий порядок, і можемо в будь-який момент відтворити траєкторію будь-якого запису, подивившись на його *id*/*origIndex* у журналі.

Коли результат готовий, модуль *B*7 забезпечує експорт і збереження даних. Як видно з рисунка 1, *B*7 приймає результат T_3 та журнал кроків із модулів *B*5 і *B*6, а також керувальні команди від *B*0. На основі цього він дозволяє вивантажити результат у файлові формати XLSX, CSV або JSON (наприклад, для передачі в іншу систему чи формування звіту) і зберегти T_3 разом із журналом у локальному сховищі браузера (IndexedDB, браузерна SQLite через sql.js) або на зовнішньому сервісі через REST API. Такий підхід, з одного боку, полегшує інтеграцію системи в реальні бізнес-процеси (дані можна забрати у звичному форматі або тримати локально), а з іншого – гарантує тривалу відтворюваність експериментів і можливість незалежної перевірки результатів: маючи T_3 і журнал, можна повторно проаналізувати, як саме був отриманий цей порядок.

Увесь рух між етапами координується скінченним автоматом у модулі *B*8. Ми фіксуємо дозволені стани (наприклад, *Idle* → *Loaded* → *CriteriaReady* → *Computing* → *Viewing* → *Viewing*/*Export*) і інваріанти, які забороняють нелегальні переходи. Це означає, що система не може «перескочити» через крок: наприклад, перехід із *Computing* назад у *CriteriaReady* неможливий без події *ev_finishSorting*, а запуск сортування (*ev_sort_all* або ш) блокується, поки не буде сформовано коректну множину критеріїв T_2 і система не перейде в стан *CriteriaReady*. Як відображено на рисунку 1, модуль ш8 отримує події Σ від *B*0 (стрілка *B*0 → *B*8) і повертає дозволені стани або повідомлення про помилки назад у *B*0 (стрілка *B*8 → *B*0). Якщо під час роботи стається збій, генерується подія *ev_error*, система переходить у стан *Error*, формує опис помилки в ш і повертає його в *B*0 у вигляді повідомлення (*toast* або *modal*), щоб користувач одразу бачив, що саме пішло не так і на якому етапі.

Оцінюючи властивості моделі в цілому, бачимо, що стабільність порядку забезпечується *tie-break* за *origIndex*, тобто рівні елементи зберігають свій початковий взаємний порядок. Обмеженість локального впливу критеріїв реалізується формулою $R_j = N / pr_j$, яка зв'язує пріоритет із максимальною «дальністю» переміщення. Пояснюваність гарантується наявністю детального журналу кроків і метрик стабільності, що дозволяють кількісно описати втручання в початковий порядок. За обчислювальною вартістю кожен прохід із вікном R_j має складність порядку $O(N \cdot R_j)$, а сумарна вартість є адитивною за пріоритетами, тобто загальна складність визначається сумою по всіх критеріях. При цьому зі збільшенням pr_j значення R_j зменшується, що природним чином обмежує витрати на критерії з нижчим пріоритетом: молодші критерії працюють із меншими вікнами й не перевантажують систему зайвими перестановками. Отже, досягається баланс між керованістю, прозорістю та продуктивністю.

Висновки. У роботі розроблено клієнтську модель системи багатокритеріального сортування табличних даних, яка поєднує локальне R-обмежене переміщення елементів ($R_j = N/pr_j$), режим adjacent-only зі стабільним tie-break за початковим індексом, уніфіковане порівняння числових і текстових атрибутів з пороговим нечітким збігом та формалізовану систему журналювання й метрик стабільності.

1. Розроблено керовану інформаційну технологію впорядкування структурованих даних із підтримкою порогового нечіткого порівняння, що забезпечує збереження часткового початкового порядку без повного пересортування масиву.

2. Побудовано модульну архітектуру *B*0–*B*8 із подієвим керуванням (FSM), що охоплює повний цикл: імпорт, очищення, специфікацію критеріїв, впорядкування, пояснення та експорт, зберігаючи контроль над локальним впливом критеріїв.

3. Запроваджено журнал кроків та систему метрик (*Kendall's τ,* середній зсув позиції, частка великих зсувів), які забезпечують відтворюваність, пояснюваність та аудит змін порядку на рівні окремих операцій.

Створена модель системи є інтегрованою у браузерні клієнтські середовища без потреби у серверній постопрацюванні, зберігає контроль над локальним впливом критеріїв і гарантує пояснюваність кожного кроку сортування.

Перспективи подальших досліджень полягають в оптимізації обчислювальної складності моделі для великих значень N, аналізі чутливості результатів до вибору параметрів pr_j та τ_j, а також у порівнянні R-обмеженого підходу з методами TOPSIS та outranking-процедурами в клієнтському (браузерному) середовищі.

Список літератури:

1. SheetJS (XLSX). Docs (Community Edition). 2025. URL: https://docs.sheetjs.com (дата звернення: 20.10.2025).
2. Papa Parse. Documentation. 2025. URL: https://www.papaparse.com/docs (дата звернення: 20.10.2025).
3. Handsontable. Multi-column sorting API / Row sorting guide. 2023. URL: https://handsontable.com/docs/javascript-data-grid/api/multi-column-sorting/ (дата звернення: 20.10.2025).
4. Díaz Ceñera M., Grigera J., Pascual Espada J. Enhancing data tables user experience via automated adaptations. *Expert Systems with Applications*. 2024. Vol. 255. Art. 124491. DOI: 10.1016/j.eswa.2024.124491.
5. Huang Y., Miao Y., Weng D., Perer A., Wu Y. StructVizor: Interactive profiling of semi-structured textual data. In: CHI '25: Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems. 2025. Pp. 393:1–393:18. DOI: 10.1145/3706598.3713484.
6. Bánhidi Z., Dobos I. Sensitivity of TOPSIS ranks to data normalization and objective weights on the example of digital development. *Central European Journal of Operations Research.* 2024. Vol. 32, No. 1. Pp. 29–44. DOI: 10.1007/s10100-023-00876-y.
7. Więckowski J., Sałabun W. Sensitivity analysis approaches in multi-criteria decision-making: A systematic review. *Applied Soft Computing*. 2023. Vol. 148. Art. 110915. DOI: 10.1016/j.asoc.2023.110915.
8. Gaona À., Guisasola A., Baeza J. A. An integrated TOPSIS framework with full-range weight sensitivity analysis for robust decision analysis. *Decision Analytics Journal*. 2025. Vol. 17. Art. 100642. DOI: 10.1016/j.dajour.2025.100642.
9. Liu X., Liu Y. Sensitivity analysis of the parameters for preference functions and rank reversal analysis in the PROMETHEE II method. Omega. 2024. Vol. 128. Art. 103116. DOI: 10.1016/j.omega.2024.103116.
10. Jokar F., Jalali Varnamkhasti M., Hadi-Vencheh A. A new ELECTRE method based on left and right score for multicriteria decision-making. *Computational Intelligence and Neuroscience*. 2023. Article ID 6414686. DOI: 10.1155/2023/6414686.
11. Abernethy J., Schapire R. E., Syed U. Lexicographic optimization: Algorithms and stability. *arXiv preprint*. 2024. URL: https://arxiv.org/abs/2405.01387 (дата звернення: 20.10.2025).
12. Knuth D. E. The Art of Computer Programming. Vol. 3: Sorting and Searching. 2nd ed. Reading (MA): Addison-Wesley, 1998. 780 p.
13. Овсяк В. К., Турчак В. Р., Овсяк О. В. Модель вибору сповіщувачів системи безпеки. *Український журнал інформаційних технологій*. 2023. Т. 5, № 2. С. 17–24. DOI: 10.23939/ujit2023.02.017.
14. Турчак В. Р., Овсяк О. В. Модель алгоритму багатокритеріального сортування даних з їх динамічно-наслідковим обмеженням переміщень. *Науковий вісник НЛТУ України*. 2025. Т. 35, № 3. С. 175–182. DOI: 10.36930/40350319.
15. Овсяк В. К., Турчак В. Р. Модель користувацького інтерфейсу для сортування структурованих даних. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво.* 2025. Вип. 60. С. 13–30. DOI: 10.36910/6775-2524-0560-2025-60-02.

Turchak V.R. MODEL OF AN INFORMATION SYSTEM FOR MULTI-CRITERIA DATA SORTING WITH DYNAMIC CONSEQUENCE-BASED MOVEMENT CONSTRAINT

Information systems for sorting structured data are crucial for analytics, logistics, educational monitoring and technical security. Their effectiveness is determined not only by algorithmic accuracy, but also by the consistency of the processing architecture – from import and normalization to the control of criteria influence and the explainability of results. The purpose of this work is to develop a model of an information system for multi-criteria sorting that preserves a partial initial order through controlled local movements of elements and ensures reproducibility of each ranking change step.

The proposed model covers the full cycle of processing tabular data: transition from the initial state T_1 to the normalized T_1', specification of the set of criteria T_2, construction of comparators with fuzzy text matching based on the normalized Levenshtein distance, and formation of the ordered result T_3 accompanied by a stability log. The architecture is implemented as a set of interacting modules (import/preprocessing; criteria manager; comparison module; sorting engine; visualization and export; logging and metrics) with clearly defined inputs/outputs and behavioral invariants.

The algorithmic core is based not on complete re-sorting of the record array but on stepwise partial "pulling up" of elements according to the priority of each criterion. For a criterion with priority pr_j a movement constraint $R_j = N / pr_j$ is introduced (where N is the number of rows), which limits the maximum distance of local displacement in a single iteration. At each step, the system selects the locally best element within the admissible window and pulls it upwards by no more than R_j positions using only consecutive adjacent exchanges (adjacent-only). In the case of equal values, a stable tie-breaking strategy is applied based on the

initial order. This information technology provides controlled changes in ranking without destroying the global order and prevents lower-priority criteria from completely dominating higher-priority ones.

The system produces an ordered list T_3 accompanied by a detailed step log and aggregated stability metrics (in particular Kendall's τ and the mean position shift). This increases the explainability and reproducibility of results in applied tasks and enables quantitative assessment of the extent of intervention in the initial order and auditing of decisions at the level of individual operations. The scientific novelty lies in the development of a model of an information system for multi-criteria sorting represented as a sequence of interacting modules with strict invariants (movement constraints, stepwise replacement mode, stable initial order, threshold-based fuzzy matching) that have consistent counterparts in the models, data structures and program interfaces. Unlike existing models, the proposed model combines controlled locality of criteria influence with formalized process logging, which ensures both stability of the partial order and transparent auditing of sorting steps. The practical significance is that the model is oriented toward use in web applications without server-side post-processing, supports mixed field types (numeric and textual), provides transparent control over the strength of each criterion's influence, includes a technology for exporting results and logs for further analysis, and can be integrated into workflows of e-commerce, technical monitoring, educational analytics and resource management.

Key words: *information system, multi-criteria sorting, modular architecture, dynamic consequence-based movement constraint, fuzzy matching, order stability, structured data.*

Дата надходження статті: 17.11.2025
Дата прийняття статті: 04.12.2025
Опубліковано: 30.12.2025