УДК 004.75

*АМОНС О.А.,*
*ХМЕЛЮК В.С.,*
*ЧАЛИЙ О.І.,*
*КИРИЧЕК О.О.*

# АТРИБУТИВНИЙ ПОШУК ДОКУМЕНТІВ В РОЗПОДІЛЕНИХ СИСТЕМАХ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

Робота присвячена огляду та аналізу алгоритмів пошуку документів в розподілених системах. Порівнюються алгоритми пошуку. Запропоновано модифікувати вибрані алгоритми пошуку та порівняння запитів на пошук документів. Пропонується схема програмної реалізації процесу атрибутивного пошуку документів в розподіленій системі електронного документообігу.

The paper is devoted to the review and analysis of algorithms for document retrieval in distributed systems. Search algorithms are compared. Modification to the selected algorithms for queries to find documents search and comparison is proposed. Software implementation design of process of attributed document retrieval in a distributed electronic document workflow system is suggested.

## 1. Вступ

У міру зростання об’єму накопичувачів і, відповідно, архівів документів, проблема ефективності пошуку інформації набуває все більш гострий характер, а падіння продуктивності ієрархічних файлових систем як засобу впорядкування стає все більш помітним. На передній план виходять такі проблеми каталогізації, як необхідність дисциплінованого підходу до іменування і розміщення файлів, вдумливому складання назв, а також значні трудовитрати по розбору вже існуючого хаосу. Швидкий розвиток мережевих технологій потребує постійного покращення рівня організації комунікації. Для впорядкування комунікаційних структур використовують розподілені системи, які керують множиною об’єднаннях в мережу комп’ютерів представляючи їх єдиним цілим. Розподілені системи мають широкий спектр властивостей: від можливості маніпулювання інформаційними потоками між віддаленими точками на планеті, до контролю кластерів відносно недалеко розташованих машин. На даний момент доступні централізовані та децентралізовані розподілені системи. Вибір типу системи головним чином залежить від кількості машин в мережі та об’єму оброблюваної інформації. Головною перевагою децентралізованих розподілених систем перед системами, що мають явно виражений центральний вузол – відмовостійкість. При чому, з розвитком мережевої інфраструктури зростає важливість цієї властивості, цим самим стимулюючи активний розвиток даного класу розподілених систем. Система повинна створювати транзитивне замикання інформаційного простору, потрібно переходити від автоматизованого пошуку інформації і від автоматизованого виконання бізнес-процесів до автоматичного. Користувач системи повинен тільки поставити завдання і отримувати результат, не беручи участі у проміжних етапах пошуку, оцінки та аналізу інформації. Головним завданням такої системи є управлінням інформацією.

Процес пошуку інформації є невід’ємною частиною функціонуванні будь-якої системи. Не є винятком і клас розподілених систем. Саме завдяки ефективним алгоритмам пошуку знаходять потрібну інформацію, що може знаходиться на N-вузлах у вигляді k-частин. При використанні модифікованих алгоритмів пошуку можна досягти збільшення продуктивності процесу пошуку на 15-18%, зменшити навантаження мережі зайвими запитами на 15%. Також на 30% збільшується швидкість реагування на запит, що відіграє важливу роль у функціонуванні розподілених систем [1]. Однак для організації продуктивного пошуку інформації необхідно підбирати алгоритми пошуку в залежності від конкретної розподіленої системи.

В даній статті пропонується використання модифікованого методу пошуку документів в розподіленій системі електронного документообігу (СЕДО), який базується на основних засадах алгоритму пошуку в ширину з ітеративним заглибленням та алгоритму ISM.

Враховуючи специфічні особливості формування документів в СЕДО та розмежування прав на доступ будемо використовувати поряд з

модифікаціями вище зазначеного алгоритму ще й статистичну інформацію.

Сформований запит буде порівнюватися з запитами, що вже виконувалися на вузлі. Та в залежності від того до якого запиту він буде подібний, буде формуватися множина вузлів, до яких направлятиметься запит. Величина глибини слідування запиту задаватиметься користувачем.

Для зменшення часу пошуку документів та вдосконаленню роботи з документами необхідно адаптувати систему пошуку документів до СЕДО.

## 2. Актуальність проблеми

Новий рівень розвитку мережевого інформаційного простору обумовлює потребу створення і розвитку прогресивних моделей інформаційного простору, інформаційних потоків, мережевого пошуку. В зв'язку з цим виникає інтерес до підходів, що базуються на розумінні інформації, як деякої міри впорядкованості деякої системи і, відповідно, до статистичних методів її обробки. Таким чином, актуальність проблеми обумовлюється потребою створення алгоритму пошуку документів для системи електронного документообігу, який дозволятиме якомога ефективніше виконувати пошук документів в розподіленій системі електронного документообігу.

## 3. Огляд існуючих алгоритмів пошуку

На сьогоднішній день існує велика кількість різноманітних алгоритмів пошуку інформації в розподілених системах. Продуктивність системи залежить від правильної побудови її функціонування, тобто потрібно вибрати такий алгоритму, який буде максимально підходити до структури системи. Не є винятком і процеси пошуку. Розглянемо деякі алгоритми пошуку в розподілених системах.

*Алгоритм пошуку в ширину (BFS, Breadth-first search)* – метод обходу та розмітки вершин графу. Вся система представляється у вигляді графу, де вузол це окрема машина. Пошук в ширину виконується в такому порядку: вершині, з якої починається обхід приписується номер 1, сусіднім вершинам – 2 і т. д. Потім по черзі розглядуються всі суміжні вершини.

Вузол-ініціатор генерує запит який адресується всім ближчим сусідам. Коли вузол в мережі отримує запит на пошук то виконується пошук в його локальному індексу і повинен обов'язково генерує запит-відповідь щоб повернути результат. Якщо будь-який вузол, який оброблював запит отримує запита-відповіді більш ніж від одного вузла, то він може виконувати операцію завантаження інформацію з найбільш доступного ресурсу.

Недоліком використання даного алгоритму є надмірне перенавантаження мережі зайвими запитами і як результат вузол з найменшою пропускною здатністю може спричинити зменшення швидкодії мережі, а перевагою являється те, що збільшується ймовірність отримання позитивної відповіді при перегляді значної частини мережі. [2,3].

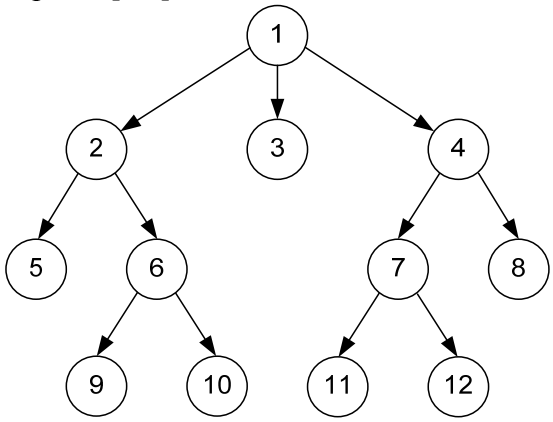


***Рис. 1. Порядок обходу дерева в ширину***

*Алгоритм пошуку в глибину (DFS, Depth-first search)* – основна ідея алгоритму полягає у тому, що для кожної не пройденої вершини необхідно знайти всі не пройдені суміжні вершини і повторити пошук для них [2,3].

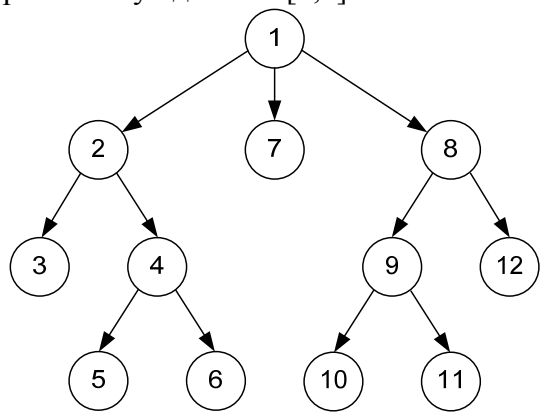


***Рис. 2. Порядок обходу дерева в глибину***

*Алгоритм випадкового пошуку в глибину (RBFS, Random Breadth First Search)* – модифікація алгоритму пошуку в ширину, суть якого полягає в тому, що вузол-ініціатор пересилає запит тільки випадково вибраним вузлам в мережі. Перевага даного алгоритму полягає в тому, що не потрібно знати глобальну інформа-

цію про стан контенту мережі, вузол може отримувати локальні рішення так швидко, як це необхідно. Недоліком являється той факт, що даний алгоритм є ймовірнісним, тому деякі великі сегменти мережі можуть стати недоступними [2, 3].

*Алгоритм пошуку A* («Астар»)* – це алгоритм пошуку по першому найкращому співпадінні на графі, який знаходить маршрут з найменшою метрикою від однієї вершини (початкової) до іншої (кінцевої). Порядок обходу вершин визначається евристичною функцією «відстань + метрика», f(x)=g(x) + h(x). Суть цього алгоритму заключається в тому, що A* крок за кроком розглядає всі шляхи від початкової вершини до кінцевої, доки не знайде мінімальний. Спочатку він розглядає ті маршрути, які «здаються» ведуть до кінцевої вершини, а при виборі вершини – враховує весь пройдений до неї шлях. A* розглядає вузли суміжні з початковим (ініціатором) таким чином, щоб метрика була найменшою, потім він розкривається. На кожному етапі алгоритм оперує множиною шляхів із початкової точки до всіх ще не розкритих вершин графу, яка знаходиться в пріоритетній черзі. Пріоритет визначається як f(x)=g(x) + h(x). Ця дія виконується доки значення f(x) кінцевої функції вершини не буде меншим, чим будь-яке значення в пріоритетній черзі або доки все дерево не буде переглянутим [4].

*Інтелектуальний пошуковий механізм (ISM, Intelligent Search Mechanism)* – даний алгоритм пошуку використовується в мережах де кожен вузол може зберігати деяку спеціалізовану інформацію. Суть даного метода полягає в тому, що вузол-ініціатор генерує повідомлення, що описує запит на пошук, знаходить множину найбільш підходящих вузлів, що можуть повернути результат, використовуючи профайлер та механізм ранжування, та надсилає запит на пошук лише цим вузлам. Профайлер – файл, який зберігає інформацію про останні запити в стек в форматі: Запит, Вузол. Коли стек заповнюється відбувається заміна останнього запита, який не використовуюся найдовше. Ранжування базується на понятті рангу релевантності. Враховуючи ранг релевантності вузол формує «представлення» про сусідні вузли, вибираються вузли до яких запит буде надходити першим. Ранг релевонтності визначає найбільший ранг вузла, який повертає найбільше результатів. Крім того в даному методі використовується параметр α, за допомогою якого можна збільшити вагу запитів найбільш подібних початковому. Таким чином можна сказати, що якість процесу пошуку за допомогою даного алгоритму залежить від сусідніх вузлів до вузла, який задає запит. Головна проблема даного алгоритму полягає тому, що запити можуть зациклюватися і не досягати деяких частин в мережі. На даний момент існують модифікації даного алгоритму, що дозволяють позбутися цієї проблеми [3].

*Алгоритм «напрямлений пошук в ширину та евристика найбільшої кількості результатів у минулому» (>RES)* – суть даного методу полягає в тому, що кожний вузол пересилає запит підмножині своїх вузлі, які будуються на основі деякої загальної статистики. Запит в методі вважається задовільним якщо видається z результатів (z – деяка константа). Даний алгоритм дуже схожий до ISM, вузол-ініціатор пересилає запити до вузлів, що повернули найбільші результати для останніх запитів. Таким чином даний алгоритм в результаті експериментів змінювався від пошуку в ширину до первинного пошуку в глибину. Також варто зауважити, що алгоритм >RES використовує більш простішу інформацію про вузли. Недоліком являється те, що даний алгоритм не аналізує параметри вузла, зміст яких пов'язаний з запитом. Перевагою даного алгоритму являється те, що він направляє запити в великі сегменти мережі, а також захватує сусідні вузли, які менш навантажені [3].

*Алгоритм «довільних блукань» (RWA, Random Walkers, Algorithm)* – суть даного алгоритму полягає в тому, що кожний вузол випадковим чином пересилає повідомлення з запитом одному із сусідніх вузлів і готовий знову приймати інші повідомлення. Даний алгоритм схожий до *RBFS* , але в *RBFS* кожний вузол пересилає запит тільки частині сусідів. До того ж в *RBFS* відбувається експоненціальне збільшення повідомлень, що пересилаються в мережі, коли в RWA – лінійне. RBFS та RWA не використовують ніяких правил щодо вибору найбільш релевантного вузла для пересилки пошукового запиту. Ще однією методикою, подібної RWA, є «адаптивний імовірнісний пошук» (Adaptive Probabilistic Search, APS) [5]. У APS кожен вузол розгортає на своїх ресурсах локальний індекс, що містить значення умовних ймовірностей для кожного сусіда, який може бути вибраний для обробки наступного запиту. Головна відмінність від RWA в даному випадку – це те, що в APS вузол використовує в якості зворотнього зв'язку результати попередніх пошуків (у вигляді умовних ймовірностей) замість повністю випадкових переходів. Тому метод APS часто дає кращі результати, ніж RWA [3, 5].

***Табл. 1. Порівняння алгоритмів пошуку***

| Алгоритм | Переваги | Недоліки |
|---|---|---|
| Пошук в глибину | Швидко відбувається процес пошуку, якщо обрана правильна гілка пошукового графу, не велике навантаження на мережу, висока швидкість роботи | Потрібно вибрати оптимальну глибину зупинки пошуку, не висока релевантність результатів, висока ймовірність випадання запиту |
| Пошук в ширину | Збільшується ймовірність отримання позитивної відповіді при перегляді значної частини мережі, не високе споживання ресурсів | Надмірне перенавантаження мережі зайвими запитами. Пошук проходить в усіх напрямках. |
| ISM | Пошук проводиться на релевантних вузлах. Ведення статистики у вигляді профайлера. Можливість змінювати ранг релевантності вузла в залежності від запиту. Збільшення швидкості та ефективності процесу пошуку за рахунок зменшення кількості повідомлення між вузлами та кількості опитуваних вузлів | Існує ймовірність зациклювання запита в мережі, завжди існує велика частина мережі до якої не доходять запити. Якість пошуку залежить від сусідніх вузлів |
| RBFS | Не потрібно знати глобальну інформацію про стан контенту мережі, вузол може отримувати локальні рішення так швидко, як це необхідно. | Даний алгоритм є ймовірнісним, тому деякі великі сегменти мережі можуть стати недоступними. |
| Алгоритм пошуку A* | Завжди знайде оптимальний шлях до цілі. При правильному налаштуванні дає гарні результати в швидкодії. | Складність алгоритму і час пошуку залежить від евристичної моделі мережі. |
| >RES | Огляд великої кількості сегментів мережі. Використання простої статистичної інформації для вибору вузлів | Відсутність можливості аналізувати вузли, які зберігають інформацію, що пов'язана із запитом |
| RWA | Запит пересилається тільки частині сусідніх вузлів. Збільшення кількості запитів в мережі відбувається за лінійним законом | Вибір вузлів відбувається випадковим чином, без яких-небудь правил. Переглядається невелика кількість сегментів мережі. Перенавантаження мережі запитами. Невисока релевантність результатів |

Таким чином, проаналізувавши все вище сказане, можна сказати, що всі розглянуті алгоритми пошуку дозволяють вирішити лише деякі проблеми, тобто на даний момент не існує такого пошукового алгоритму, який був би чимось універсальним, широковживаним і дозволив би проводити процес пошуку максимально ефективно, незалежно від структури системи. Припускається, що саме структура системи в цілому повинна «підказувати» розробникам, який саме алгоритм варто використати.

Наступним етапом виконано порівняння алгоритмів пошуку – це виділення основних параметрів, по яких порівнюємо та зведення все в одну таблицю (табл. 2).

Для всіх параметрів порівняння алгоритмів пошуку використовували наступну міру: <0; 0.6; 1>. Тобто, діапазон «0 – 0.6» – відповідає позначці «-», а діапазон «0.6 – 1» – відповідає позначці «+» в табл. 2.

Формули розрахунків параметрів наведені далі в описі математичної моделі.

В табл. 2 зібрано характеристики розглянутих алгоритмів. Ці алгоритми можна згрупувати наступним чином:

- ISM, >RES, A* – характеризуються як найбільш стійкі та продуктивні алгоритми серед усіх розглянутих, але при роботі споживають багато ресурсів системи;
- RWA та RBFS – менш продуктивні, існує велика ймовірність отримання не релевантних результатів, але швидкість роботи на досить високому рівні та не велике споживання системних ресурсів;
- Пошук в глибину, пошук в ширину – ці алгоритми дещо схожі між собою, але все-таки мають ряд відмінних властивостей, які показують ефективність того чи іншого алгоритму в різних умовах.

Керуючись цими факторами, властивостями алгоритмів із Таблиці 2 та обмеженнями СЕДО ми вирішили використовувати деякі властивості алгоритму пошуку в глибину з ітеративним заглибленням та алгоритму ISM.

***Табл. 2. Порівняння алгоритмів пошуку***

| Алгоритм | Синтаксичний аналіз запиту | Використання статистичної інформації | Релевантність результатів | Високе навантаження на мережу | Висока ймовірність випадіння запиту | Висока швидкість роботи | Велике споживання ресурсів |
|---|---|---|---|---|---|---|---|
| Пошук в глибину | - | - | - | - | - | + | + |
| Пошук в ширину | - | - | + | + | + | - | - |
| ISM | + | + | + | - | - | + | + |
| RWA | - | - | - | - | - | + | - |
| >RES | + | + | - | - | - | + | + |
| A* | + | - | + | + | - | + | + |
| RBFS | - | - | - | - | + | - | - |

## 4. Модифікований алгоритм пошуку документів

В розроблюваній підсистемі атрибутивного пошуку документів, використовується алгоритм пошуку в ширину із ітеративним заглибленням. Величина заглиблення встановлюється на вибір користувача. Суть даного методу полягає у тому, що вузол-ініціатор запиту виконує пошук у себе і потім передає запит одному із сусідів. Вузол, який отримав запит виконує пошук у себе на вузлі і передає запит своїм сусідам на величину d, вказану вузлом-інціатором і повертає потік знайдених результатів. Далі вузол-ініціатор передає запит іншому сусідові який виконує аналогічні дії як і всі інші вузли-сусіди вузла-ініціатора.

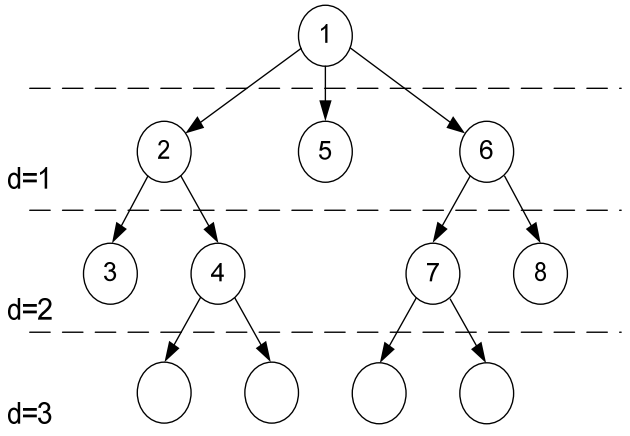


***Рис. 3. Порядок обходу дерева в ширину з ітеративним заглибленням (d=2)***

Пропонується модифікувати цей алгоритм шляхом використання статистичної інформації. Суть модифікації полягає в тому, що при кожному виконанні запиту на будь-якому вузлі зберігається інформація про структуру самого запиту та про відповідь, яка була відправлена. При надходженні нового запиту на вузол, спо чатку аналізується статистичні дані вузла, а сам запит «заморожується». Якщо новий запит буде схожим на будь-який запит, що вже виконувався на цьому вузлі, то буде сформовано «корисну множину» вузлів. До такої множини включатимуться ті вузли, які повертали відповідь на найбільш схожі запити. Пошук буде виконуватися вже не по всіх сусідах вузла, а тільки по тим, які «можуть» містити потрібну інформацію. Таким чином кожний вузол системи формує представлення про інформацію, яка зберігається на ньому. Схожість оцінюється алгоритмами наведеними нижче.

Розглянемо процес порівняння запитів. Розробимо алгоритм визначення подібності двох запитів, для цього весь процес порівняння розіб'ємо на два етапи: на першому етапі визначимо чи є один запит підмножиною іншого, а на другому на скільки обидва запити подібні.

## 5. Алгоритм визначення, чи є один запит підмножиною іншого

Якщо кожний елемент множини A належить і множині B то можна говорити, що множина A є підмножиною B, тобто:

$$A \subseteq B \qquad (5.1)$$

Виходячи з цього поняття, сформуємо словник всіх атрибутів D=<індекс, атрибут>.

Використовуючи цей еталонний словник D формуємо дві множини A та B таким чином, щоб множина A включала всі індекси які відпо-

відають використаним атрибутам у першому запиті, а множина B – у другому.

Виділимо такі етапи:

1) Формуємо дві множини, множина A – всі атрибути першого запиту, множина B – всі атрибути другого запиту.
2) Визначаємо потужності сформованих множин m(A) та m(B).
3) Порівнюємо знайдені потужності, таким чином визначаючи яка множина містить більше елементів. Якщо m(A)=m(B) то переходимо до пункту (6).
4) Виконуємо операцію перетину множин A і B. Як результат отримуємо множину спільних елементів.
$$S = A \cap B \qquad (5.2)$$
5) Визначаємо підмножини a=S на множині A, b=S на множині B.
6) Підраховуємо коефіцієнт подібності множин (запитів) $K_s$. Якщо хоч одна із множин A\S та B\S не є пустою, то припиняється процес порівняння запитів, оскільки робиться припущення, що два порівнюваних запити не є схожими. А якщо A\S та B\S пусті, то припускається, що два порівнювані запити максимально подібні по структурі. Якщо одна з множин є пустою, а інша – ні, то для продовження процесу порівняння потрібно щоб коефіцієнт подібності Ks був більше 0,6.
$$K_S = \frac{m(S)}{m(L)} \qquad (5.3)$$
m(S) – потужність множини S
m(L) – потужність не пустої множини серед двох порівнюваних множин A та B.
7) Виконуємо операцію сортування по індексам в кожній множині. Встановляємо взаємно однозначну відповідність між підмножинами a та b, або між множинами A та B, якщо m(A)=m(B).
8) Порівнюємо попарно всі елементи.

## 6. Алгоритм визначення чи є один запит подібний іншому (попарного порівняння елементів)

Атрибути документу можна умовно розділити на ті, які користувач заповнює сам (наприклад, номер документу, короткий зміст і т.д.) та на атрибути які користувач вибирає із списку (наприклад, журнал, тип документу, тематика та ін.). В залежності від того, до якого типу належить вибраний атрибут будемо формувати дві множини атрибутів з різними ваговими коефіцієнтами. Керуючись тим, що співпадіння слів більш вагоме чим співпадіння літер виконаємо наступні дії. Атрибути документу розділимо умовно на дві множини: множина $A_1$, множина $B_1$. До множини $B_1$ належать атрибути, що мають ваговий коефіцієнт 2, тобто ті атрибути, які користувач вибирає із списку. А до множини $A_1$ належать всі інші атрибути, їх ваговий коефіцієнт – 1, ті, що користувач вводить самостійно. Таким чином процес аналізу подібності елементів можна розділити на два етапи. На першому етапі відбувається формування множин із задіяних атрибутів, визначення максимально можливого коефіцієнту N.
$$N = r * 100\% \qquad (6.1)$$
r – кількість атрибутів

На другому етапі відбувається безпосередньо порівняння елементів, по сформованим множинам. Оскільки в множині $A_1$ знаходяться атрибути, значення яких користувач вводить самостійно, то спочатку потрібно привести введений текст до канонічного виду і потім схожі атрибути порівнюються з використанням «алгоритму шинглів»[6] (або просто визначається перетин цих двох підмножин). Подібність введеної інформації визначається коефіцієнтом подібності, який дорівнює ваговому коефіцієнту (при використанні «алгоритму шинглів») або визначається як:
$$s = \frac{k}{n} * 100\% \qquad (6.2)$$
k – кількість однакових символів\слів
n – загальна кількість символів\слів

Значення коефіцієнту s перемножується на ваговий коефіцієнт множини $A_1$ і записується в загальний коефіцієнт K.

Множина $B_1$ складається із атрибутів, значення яких не вводяться користувачем, а вибираються із запропонованого списку. Таким чином, аналогічно і з визначенням коефіцієнту подібності на множині $A_1$ на множині $B_1$ виконуються ті ж самі операції:
$$s_1 = \frac{k_1}{n_1} * 100\% \qquad (6.3)$$
$k_1$ – кількість однакових слів
$n_1$ – загальна кількість слів

Коефіцієнт $s_1$ перемножується із ваговим коефіцієнтом множини $B_1$ і потім додається до загального коефіцієнту K. Аналогічно цей про-

цес повторюється для всіх атрибутів множини $B_1$. Таким чином при співпадінні елементів на множині $B_1(s_1>40\%)$ робимо висновок, що два порівнювані атрибути є елементами одного дерева.

Результуючий коефіцієнт подібності елементів визнається як:

$$M = \frac{K}{N} * 100\% \qquad (6.4)$$

K – глобальний ваговий коефіцієнт

N – максимально можливий ваговий коефіцієнт

M – коефіцієнт подібності

На основі проведених випробувань в пошукові системі СЕДО, рекомендується встановлювати граничне значення коефіцієнту подібності не менше чим 75%.

Але все-таки даний алгоритм має ряд як переваг, так і недоліків.

Недоліки:

- Складність реалізації механізму аналізу запитів;
- Пошук виконується по всім таблицям в базах даних вузла;
- При великому завантаженні одного із вузлів, значно підвищується час пошуку.
- Організація додаткових заходів щодо очищення статистичних даних на вузлах, які не використовувалися або використовувалися в незначній мірі.

Переваги:

- Зменшується навантаження на мережу при подібності запитів;
- Зменшується час пошуку в цілому;
- Дає можливість вибору вузлів пошуку.
- Потужна система аналізу запитів.

Використання цього модифікованого алгоритму породжує такі проблеми, як:

- Кожен вузол повинен вести свою статистику;
- Затрачується додатковий час на аналіз статистичних даних;
- Якщо при аналізі статистичних даних не знайдено схожих запитів з оброблюваним, то виконується новий пошук по всім таблицям і як результат час пошуку збільшується;
- Відбувається накопичення статистичної інформації.

## 7. Математична модель модифікованого алгоритму пошуку

За допомогою математичної моделі можна промоделювати як сам процес пошуку документів, так і роботу самої системи пошуку. Розгляд математичної моделі розпочнемо із визначення основних понять.

Отже, в загальному вигляді процес пошуку документів можна представити у вигляді впорядкованої та скінченної сукупності елементів (кортежу) IR=<D, Q, R(d,q)>, де D – множина документів на вузлі, а Q – множина представлення інформаційної потреби (запиту);

R(d,q) – функція ранжування:

- зв'язує документ d із множини D та запит q із множини Q;
- визначає порядок документів відповідно до запиту q.

Релевантний документ – документ, зміст якого максимально правильно задовольняє інформаційний запит, тобто семантична відповідність пошукового запиту та пошукового образу документа. Для визначення релевантності найчастіше використовують TF-IDF метод [7].

Пертинентність – відповідність отриманих в результаті пошуку документів інформаційним потребам користувача [7].

Для оцінювання релевантності результатів пошуку основуємося на загальній моделі [3], та доповнимо її оцінюванням подібності документів запиту. Отже, введемо наступні позначки

***Табл. 3. Типи документів***

| Документи | Видані | Не видані |
|---|---|---|
| Релевантні | $D_{gr}$ | $D_{nr}$ |
| Не релевантні | $D_{gnr}$ | $D_{nn}$ |

За допомогою табл. 3 розраховувати показники інформаційного пошуку варто наступним чином.

Коефіцієнт повноти (Recall):

$$R = \frac{D_{gr}}{D_{gr}+D_{nr}} \qquad (7.1)$$

Коефіцієнт точності (Precision):

$$P = \frac{D_{gr}}{D_{gr}+D_{gnr}} \qquad (7.2)$$

Коефіцієнт правильності (Accuracy):

$$A = \frac{D_{gr}+D_{nn}}{D_{gr}+D_{gnr}+D_{nr}+D_{nn}} \qquad (7.3)$$

Коефіцієнт помилки (Error) :

$$E = \frac{D_{gnr}+D_{nr}}{D_{gr}+D_{gnr}+D_{nr}+D_{nn}} \qquad (7.4)$$

Значення F-міри (F-measure):

$$F = \frac{2}{\frac{1}{P}+\frac{1}{R}} \quad (7.5)$$

Коефіцієнт середньої точності (Average precision):

$$ArgPrec = \frac{1}{k}\sum_{i=1}^{k} P(i) \quad (7.6)$$

k – кількість документів, релевантних деякому запиту.

i – номер релевантного запиту документа.

P(i) – точність i-ого релевантного документу.

Подібність документів запиту q на вузлі $P_i$

$$P_{sim}(P_i, q) = \sum Q_{sim}(q_j, q)^{\alpha} \quad (7.7)$$

$P_i$ – і-тий вузол;

q – пошуковий запит;

$q_j$ – відповідь від вузла $P_i$;

$Q_{sim}$ – функція визначення подібності запитів;

α – коефіцієнт «підсилення» більш подібних запитів:

α > 0 – розглядається один сусід

α = 0 – розглядається К-сусідів

α = 1 – подібність по всім відповідям однакова.

Ваговий коефіцієнт в статистиці

$$R = \frac{R_d}{N_d} \quad (7.8)$$

Час потрібний на формування запиту (обробка вузлом візуальних елементів)

$$t_e = t_{eo} * n_e \quad (7.9)$$

$t_{eo}$ –час, затрачуваний на обробку одного елементу

$n_e$ – кількість заповнених елементів

## 8. Реалізація модифікованого алгоритму пошуку

Спочатку користувач заповнює поля пошуку, вибирає атрибути і при натисненні на кнопку «Почати пошук» відбувається відправлення даних на сервер застосувань, де відбувається формування словників вибраних елементів із надісланих даних. На основі цієї інформації відбувається безпосередньо формування запиту, його аналіз та розбиття на підзапити. На кожному вузлі відбувається порівняння виконуваного запиту із статистичною інформацією по вище описаному алгоритму.

Отже, якщо запит подібний будь-якому запиту, що вже виконувався на вузлі, формується список вузлів до яких буде перенаправлятися запит на пошук документів. Після виконання пошуку на вузлах через Менеджер БД виконується повернення набору даних до Серверу застосувань, де вже виконується обмеження прав доступу та об'єднання результатів Серверу застосувань надсилає агреговані результати до Форми пошуку, де користувач може у зручному вигляді переглядати знайдені результати та виконувати подальші маніпуляції із знайденими документами.

Але існує й інший сторона медалі. При обробці вузлом запита він «заморожується», тобто запит не розповсюджуються далі, залишаються в середині деякої множини вузлів. Заморожені запити отримують відповіді із потоку відповідей, що проходить через ці вузли, ініційованого «діючими» запитами. При ретельному виборі множини «заморожених» запитів, отримавши відповіді із потоку даних, збільшується достовірність формування правильних відповідей, та зменшується час протікання цього процесу, навіть при перенавантаженій системі. Робимо припущення, що якщо запити схожі, то і відповіді мають бути схожими в межах деякої метрики. Таким чином, коли в системі виникають два, а то й більше схожих запитів в межах деякої метрики, використовуючи модифікований метод пошуку значною мірою скорочується час виконання запитів у всій системи, зникають перенавантаження на вузлах системи, зменшується кількість зайвих повідомлень в мережі чим покращується ефективність роботи всієї системи.

Вузли підключаються до системи та покидають її динамічно, в будь-який час. Для цього в системі потрібно встановлення додаткового серверу S-i, який буде підтримувати цю динаміку. Його функції полягають у тому, що коли вузол підключається до системи, сервер визначає з якими іншими вузлами він буде обмінюватися даними, потім стартує протокол з'єднання. Сервер такого типу бере участь тільки у формуванні і зміні системи, а не для передачі і обробки запитів.

На рис. 4 представлено неповну діаграму класів розподіленої СЕДО. Основний клас, що описує документ – це DocumentReestration. Він містить основні атрибути та посилання на інші таблиці.

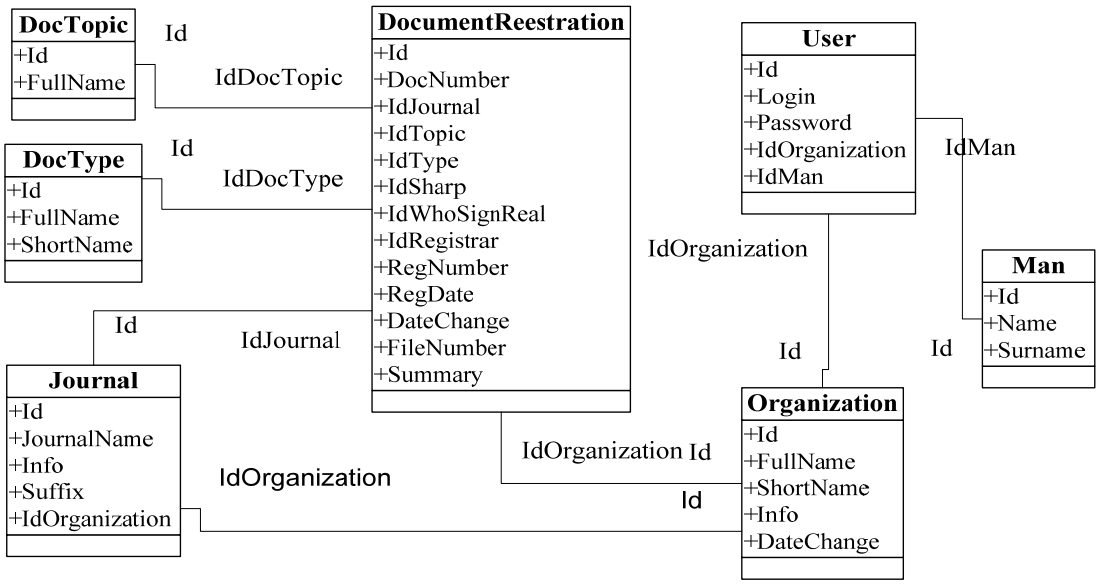


***Рис. 4. Діаграма класів***

## 9. Експериментальне дослідження

В процесі розробки системи, до отримання кінцевих результатів ефективності, ми отримали проміжні результати, які є не менш важливими для оцінки процесу пошуку та для майбутніх модифікацій. Таким чином, велику увагу було приділено процесу порівняння запитів: створеного запиту та запитів із статичної інформації кожного із вузлів. Тестування проводилося на 20 вузлах.

Також на рис. 5 графічно показано як змінювався час пошуку документів в залежності від кількості вузлів в мережі. А на рис. 6 показано взаємозв'язок між кількістю запитів, що генеруються у мережі від кількості вузлів даної мережі.

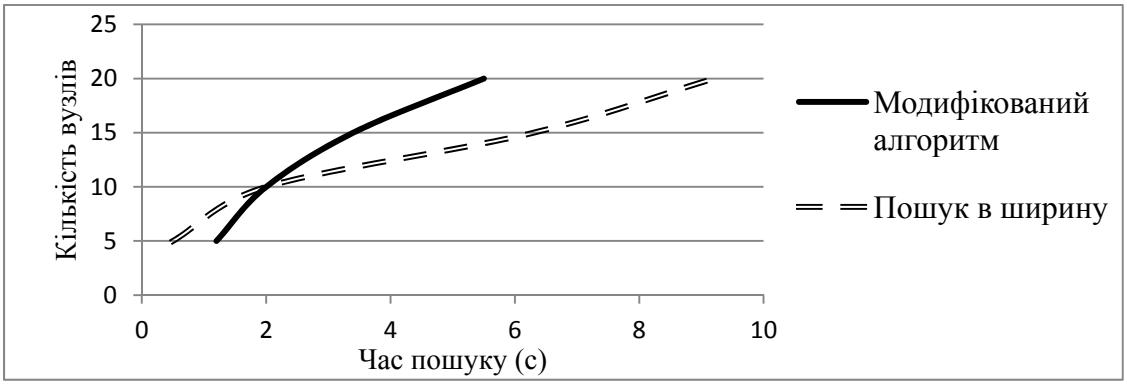


***Рис.5. Залежність часу пошуку документів від кількості вузлів***

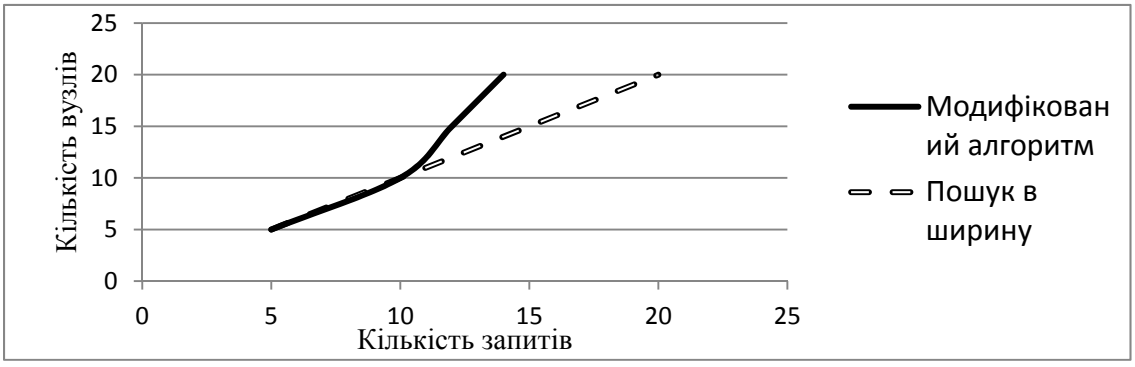


***Рис.6. Залежність кількості запитів від кількості вузлів***

Кількість запитів в мережі при використанні пошуку в ширину лінійно залежить від кількості сусідніх вузлів, а при використанні модифікованого методу пошуку кількість запитів зменшується завдяки розповсюдженню запитів тільки до тих вузлів, що можуть містити потрібну інформацію з високим рівнем ймовірності.

## 10. Висновок

Проаналізувавши найбільш популярні алгоритми пошуку документів в розподілених мережах, та враховуючи структуру СЕДО-КПІ [8], існуючу пошукову систему [9] та властивості платформи швидкої розробки застосувань SmartBase [8, 10] було запропоновано модифікувати існуючу систему шляхом адаптування та комбінування алгоритму ISM та алгоритму пошуку в ширину з ітеративним заглибленням.

Таким чином, запропонований модифікований метод пошуку документів в розподілених системах електронного документообігу дозволяє значно покращити процес пошуку документів. Покращилася продуктивність системи при збільшені навантаження як на всю систему, так і на окремі вузли. Такого результату вдалося досягти завдяки використанню «заморожування» запитів. Запит залишається в межах деякої множини вузлів. В цей час він виконується на самому вузлі та порівнюється з тими, що вже виконувалися для визначення корисної множини вузлів із всіх сусідів для подальшого перенаправлення. Також завдяки аналізу подібності запитів, який проводиться кожним вузлом окремо, скоротився час пошуку та множина сусідніх вузлів, яким направляється запит.

## Список літератури


1. Телятніков О. О. Моделі та алгоритми оптимізації розподілених баз даних комп'ютерних інформаційних систем : автореф. дис. на здобуття нак. ступеня д-ра тех. наук : спец. 05.13.06 "Технічні науки"/ Телятніков Олександр Олегович; Донецький національний університет – Д., 2005.
2. Кормен Томас. Алгоритмы. Построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М. : Вильямс, 2005. – 1296с.
3. Ландэ Дмитрий Владимирович. Интернетика: Навигация в сложных сетях: модели и алгоритмы / Д. В. Ландэ, А. А. Снарский, И. В. Безсуднов. – М.: ЛИБРИКОМ , 2009. – 264с
4. Рассел С. Дж., Искусственный интеллект: современный подход / С. Дж. Рассел, П. Норвиг; пер. с англ. К.А. Птицына. – М: Вильямс, 2006. – 1408с.
5. Zeinalipour-Yazti D. Information Retrieval in Peer-to-Peer Networks / D. Zeinalipour-Yazti, V. Kalogeraki, D. Gunopulos //IEEE CiSE Magazine, Special Issue on Web Engineering. – 2004. – P. 1-13.
6. Jun Fan A Fusion of Algorithms in Near Duplicate Document Detection / Jun Fan, Tiejun Huang // Lecture Notes in Computer Science. New Frontiers in Applied Data Mining. – 2012. – Vol.7104/2012. – P. 234-242.
7. Wen Zhang Comparative study of TF*IDF, LSI and multi-words for text classification / Wen Zhang, Taketoshi Yoshida, Xijin Tang A. // Expert Systems with Applications. – 2011. – Vol.38. – P. 2758–2765.
8. Теленик С.Ф. Алгебри для автоматичного проектування схем генерації і оброблення електронних документів / С.Ф. Теленик, В.С. Хмелюк, І.О. Безпалий, І.В. Клепач // Інформатика та системні науки (ІСН-2010). – 2010. – С. 185-188.
9. Теленик С.Ф., Пошук і реферування в системі електронного документообігу / С.Ф. Теленик, О.Ю. Шкабура, Н.О. Подригайло// Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка. – 2009. – №51.– С. 177-184.
10. Теленик С.Ф. Подход к построению бизнес-процессов в адаптивной технологии SmartBase / С.Ф. Теленик, В.С. Хмелюк, К.А. Крижова // Вісник національного технічного університету «ХПІ». –2007.– №7.– С. 86-100.