УДК 621.396.6.001

Магістрант Волчанова Ю.В., к.т.н., доцент Зорін Ю.М.

**Національний технічний університету України
«Київський політехнічний інститут»**

ГЕНЕТИЧНИЙ АЛГОРИТМ ОПТИМІЗАЦІЇ ЗАПИТІВ ДО РЕЛЯЦІЙНИХ БАЗ ДАНИХ

Вступ

Оптимізація запитів до баз даних – це процес, що здійснює пошук оптимального плану виконання запитів зі всіх можливих для заданого запиту. Один і той же результат може бути отриманий різними способами (планами виконання запитів), які можуть істотно відрізнятися як за витратами ресурсів, так і за часом виконання.

Найпопулярнішими в даний час є реляційні бази даних. Оператор реляційної алгебри – об'єднання (англ. - *Join*) в реляційних базах даних має найбільшу вартість витрати ресурсів під час виконання. Сутність оптимізації полягає в пошуку мінімуму функції вартості від перестановки порядку виконання об'єднань таблиць. Із зростанням числа таблиць k в запиті, кількість можливих перестановок зростає як $k!$, отже, пропорційно $k!$ зростає і їх час оцінки.

В [1] сформульовано принцип застосування генетичного алгоритму (ГА) для оптимізації запитів, згідно з яким генерувалась визначена кількість можливих перестановок операцій об'єднання, кожна з яких представляла окремий план виконання запиту. Далі кожен варіант перестановки кодувався окремою хромосомою і згідно з принципами ГА над хромосомами здійснювалися операції схрещування, мутації і відбір хромосом, які дають мінімальні оцінки вартості за визначену кількість ітерацій. В результаті відбору залишаються лише ті хромосоми, які дають менше, порівняно з попередньою ітерацією, значення функції вартості. Алгоритм повторюється декілька разів, після чого вибирається найбільш ефективне рішення. Цей підхід порівнюється з алгоритмом оптимізації *System-R* [2] і дає результати достатні для пошуку шляхів його вдосконалення, що і є метою даної роботи.

Постановка задачі

Декларативний запис запиту може бути поданий у вигляді графа запиту, що нагадує дерево, у якого листки – це таблиці, які беруть участь в

операціях об’єднання, внутрішні вузли – це точки, які об’єднують елементи даних, а дуги, які сполучають листки із внутрішніми вузлами і внутрішні вузли між собою, – це власне потік даних, що вибираються з відповідних структур даних, задіяних у об’єднанні. До структур даних відносяться як власне набір полів таблиці, так і сформоване представлення даних, яке утворюється після об’єднання даних з двох таблиць. Якщо всі внутрішні вузли такого дерева мають лише по одному підпорядкованому листку, тоді дерево – лінійне, або дерево з лівосторонньою вкладеністю, далі просто лівостороннє дерево. У іншому випадку дерево називається *кущем*. Наприклад для запиту (1) лівостороннє дерево і кущ будуть мати вигляд, як зображено на рис. 1 (а) і (б) відповідно.

(A Join C) and (B Join C) and (C Join D) and (D Join E) and (D Join F) (1)

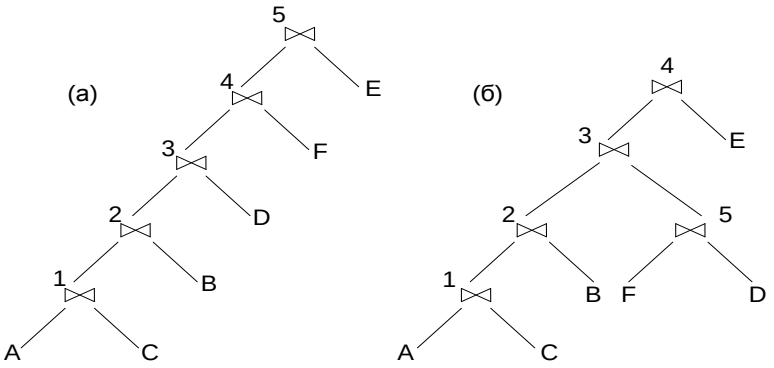

Рис. 1. Граф дерева з лівосторонньою вкладеністю (а), кущ (б)

Внутрішні вузли графа дерева – оператори об’єднання, мають цілочисельний ідентифікатор в діапазоні *[1…N]*, де *N* позначає загальну кількість операторів об’єднання в запиті. Хромосома являє собою рядок цілих чисел, який відображає один план виконання запиту з одним порядком розташування об’єднань таблиць в запиті.

Основні етапи ГА

Ініціалізація початкової популяції відбувається випадковим чином. Відбір хромосом, які будуть брати подальшу участь у процесі пошуку оптимального розв’язку виконується також випадковим чином. Дегенерації наступного покоління запобігає стратегія заміщення хромосом: заміщується найгірша копія або найгірша хромосома, якщо копії немає.

Оскільки задача оптимізації запиту зводиться до мінімізації функції вартості використання ресурсів, цю функцію можна використати як міру пристосованості хромосоми або фітнесс-функцію. Згідно з нею пристосованість хромосоми буде тим вища, чим менша вартість

використання ресурсів відповідного їй плану виконання запиту. У роботі використано просту модель функції вартості [3]. Вона визначає вартість дерева об'єднання за сумою кортежів для кожного внутрішнього вузла дерева графу і розміром проміжних зв'язків між вузлами.

У роботі запропоновані різні оператори кроссовера. В однорідному впорядкованому кроссовері (ОВК), спочатку створюється випадковий двійковий рядок такої ж довжини, як і батьківська хромосома. Потім в першому нащадку генами першої батьківської хромосоми заповнюються позиції, що відповідають одиницям двійкового рядка. Решта позицій, тобто ті, що відповідають нулям двійкового рядка, першого нащадка заповнюються генами другої батьківської хромосоми в порядку їх появи. Хромосома другого нащадка формується аналогічним чином.

У пріоритетному захисному кроссовері (ПЗК) випадковим чином генеруються два вектори довжини батьківських хромосом з елементами множини {1,2}. Цей вектор визначає послідовність, згідно з якого беруться гени з першої та другої батьківських хромосом. Після того як ген вибирається з однієї батьківської хромосоми, він більше не допускається до вибору з другої батьківської хромосоми та додається до хромосоми нащадка. Цей крок повторюється доки в обох батьківських хромосомах не залишиться генів, доступних для вибору, та нащадки матимуть всі гени в хромосомі. Цей метод зберігає порядок в генах батьківських хромосом, але при цьому може виникнути додаткова пара батьківських генів. Вона може з'явитися між першою парою сусідніх генів з різних батьківських хромосом. В роботі запропоновано модифікацію методу пріоритетного захисного кросовера (МПЗК) з метою уникнення дублювання. Вона полягає в тому, що перший ген другої батьківської хромосоми розміщується в першій батьківській хромосомі на останній позиції. Потім застосовується пріоритетний захисний кросовер. Оператор мутації обирає випадково два гени та міняє їх місцями.

Висновки

Для перевірки ефективності використовувались різні комбінації типу кроссовера, ймовірності кроссовера і мутації та розміру популяції. Було виконано тести на десяти різних запитах з наступними параметрами: 1) тип кроссоверу: ОВК, ПЗК, МПЗК; 2) розмір популяції L = 30 і L = 60; 3) ймовірність кроссовера - 0,75; 4) ймовірність мутації - 0,25. Згідно цих параметрів були вибрані комбінації для тесту, що наведені у табл. 1.

Дані тесту (рис. 2) показують, що з операторів кроссовера, згідно обох значень показників, ОВК має кращі показники ніж ПЗК, який в свою чергу має кращі показники, ніж МПЗК. Це відбувається тому, що МПЗК

надмірно витримує порядок генів батьківських хромосом, що робить хромосоми нащадків близькими до батьківських в пошуковій області.

Табл. 1. Параметри генетичних алгоритмів

Назва алгоритма	Розмір популяції, L	Кроссовер	Ймовірність кроссовера	Ймовірність мутації
ОВК	30	ОВК	0,75	0,25
ПЗК	30	ПЗК	0,75	0,25
МПЗК	30	МПЗК	0,75	0,25
ПЗК6	60	ПЗК	0,75	0,25

Важливим результатом є те, що генетичні алгоритми з розміром популяції $L = 30$ мають менший час оптимізації ніж ГА з $L = 60$, але ГА

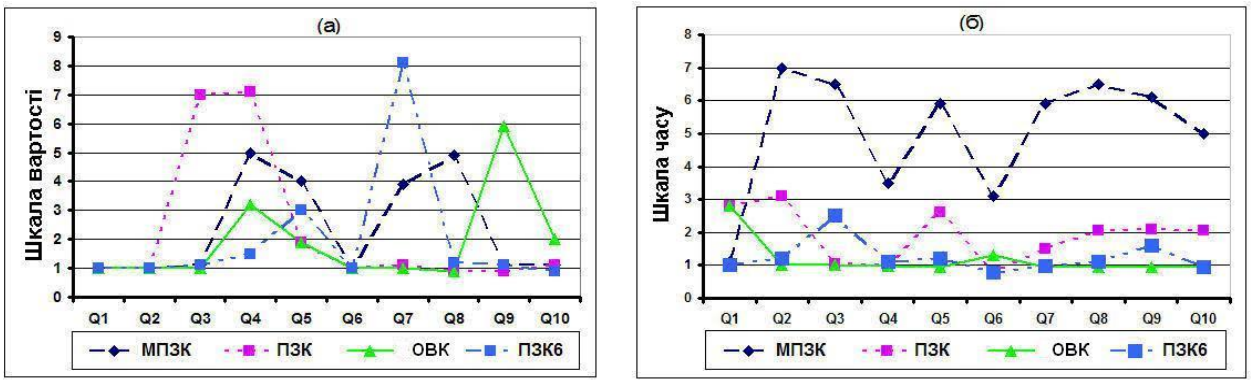

Рис. 2. Вартість виконання алгоритмів (а), час виконання алгоритмів (б)

з найкращими результатами відносно дуже близькі. Це має місце тому, що запропонований ГА скорочує простір пошуку.

Література

1. *Bennett, K., Ferris, M. C., and Ioannidis, Y. A* genetic algorithm for database query optimization. – Tech. Rep.TR1004, Univ. Wisconsin, Madison, 1991. – 235p.
2. *P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price*. Access path selection in a relational database management system // In Proc. of the ACM SIGMOD Conf. on Management of Data, Boston, USA, Jun 1979. – pp. 23-34.
3. *Lahiri, T.* Genetic Optimization Techniques for Large Join Queries. //In Proceedings of the Third Genetic Programming Conference (Univ. Of Wisconsin, Madison, July 22-25, 1998). Morgan Kaufmann, 1998. – pp.535-540.