УДК 004.852

Севідов Павло Миколайович
студент
Національний технічний університет України «Київський політехнічний інститут»
Севидов Павел Николаевич
студент
Национальный технический университет Украины «Киевский политехнический институт»
Sevidov P.
student
National Technical University of Ukraine "Kyiv Polytechnic Institute"

ОПТИМІЗАЦІЯ ЗАПИТІВ ЗА ДОПОМОГОЮ ГЕНЕТИЧНОГО АЛГОРИТМУ

ОПТИМИЗАЦИЯ ЗАПРОСОВ С ПОМОЩЬЮ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

QUERY OPTIMIZATION BY GENETIC ALGORITHM

Анотація. *Генетичні алгоритми являють собою потужний метод пошуку, заснований на механіці природного відбору і природного генетики, які успішно використовуються для вирішення завдань в самих різних дисциплінах.*

У цій статті представляється використання генетичних алгоритмів в одній з найважливіших завдань оптимізації в області комп›ютерної науки, оптимізація запитів бази даних для великого join запиту.

Сучасні методи оптимізації запитів є недостатніми для підтримки, деяких формується додатків, що використовують бази даних. У цій статті описано проблему оптимізації запитів до бази даних і описано адаптацію генетичного алгоритму. І порівняння між простими SQL запитами, що мають п›ять join і той же запит з використанням генетичного підходу. А також дано основний огляд Carquinyoli Genetic Optimizer, заснованого на генетичному програмуванні.

Ключові слова: *Генетичні алгоритми, оптимізація запитів.*

Аннотация. *Генетические алгоритмы представляют собой мощный метод поиска, основанный на механике естественного отбора и природной генетики, которые успешно используются для решения задач в самых разных дисциплинах.*

В этой статье представляется использование генетических алгоритмов в одной из самых важных задач оптимизации в области компьютерной науки, оптимизация запросов базы данных для большого join запроса.

Современные методы оптимизации запросов являются недостаточными для поддержки, некоторых формирующейся приложений, использующих базы данных. В этой статье описано проблему оптимизации запросов к базе данных и описано адаптацию генетического алгоритма. И сравнение между простыми SQL запросами, имеющих пять join и тот же запрос с использованием генетического подхода. А также дано основной обзор Carquinyoli Genetic Optimizer, основанного на генетическом программировании.

Ключевые слова: *Генетические алгоритмы, оптимизация запросов.*

Abstract. *Genetic Algorithms are a powerful search technique based on the mechanics of natural selection and natural genetics that are used successfully to solve problems in many different disciplines.*

In this paper we present genetic algorithms in one of the most important optimization problems in computer science, database query optimization for large join query.

Current query optimization techniques are inadequate to support some of the emerging database application. In this paper, we outline a database query optimization problem and describe the adaptation of genetic algorithm. And comparison between simple SQL queries having five join and same query using genetic approach. And also give basic overview of the Carquinyoli Genetic Optimizer based on Genetic Programming.

Keywords: *Genetic Algorithms, Query Optimization.*

Введение

Генетический алгоритм становится широко используемым и принятым методом для решений очень сложных задач оптимизации. Он был использован для решения широкого спектра проблем, таких как оптимизации, интеллектуальный анализ данных, игры, эволюционировали поведение в биологических сообществах и т.д.

Пользователи действительно нужно использовать очень большие запросы на соединение (join queries), чтобы поддержать их в своих бизнес-решениях [1]. Кроме того, сложность этих запросов будет увеличиваться, если СУБД может справиться с ними легко. С помощью этих общих сценариев, в настоящее время коммерческие СУБД становятся не в состоянии поддерживать удовлетворительные результаты, сохраняющие минимальные требования к производительности [2]. А именно, динамические методы программирования, примененные к оптимизации запросов, накладывают ограничения на время и память. Так как пространство поиска растет экспоненциально с линейным увеличением числа соотношений, участвующих в запросе, эти алгоритмы должны сохранить экспоненциально большее число частичных планов в памяти. Этот процесс отнимает очень много времени и обычно заканчивается без решения, когда у оптимизатора заканчивается память. Различные подходы были предложены для исправления этой ситуации [3, 5, 6].

Эта статья организована следующим образом. Раздел 2 содержит вводный материал, обеспечивающий некоторые общие принципы работы генетического алгоритма. Раздел 3 посвящен разнообразным алгоритмам оптимизации для оптимизации проблемы большого запроса на соединение, со сравнением с точки зрения оценки работы времени между простым SQL-запросом, имеющего пять запросов на соединение и тот же запрос с использованием генетического подхода. И раздел 4 посвящен основной обзор Carquinyoli Genetic Optimizer на основе генетического программирования.

2. Генетическое программирование в оптимизации запросов

В Генетическое программирование (GP) Основная идея заключается в том, чтобы получить лучшее решение с помощью эволюционных методов [3, 4].

Основное поведение этого типа алгоритмов заключается в следующем.

Первоначальный набор программ создается с нуля. В данной работе мы представляем их в виде древовидных структур, так как они являются наиболее приемлемым подходом, учитывая, что план выполнения запроса (QEP) в СУБД, как правило представлен в виде tree-shaped структуры. Этот набор также называется начальной популяцией.

После того, как начальная популяция создана, мы итерационно применяем набор генетических преобразований на членов популяции. Операторами первичного преобразования являются **Кроссинговер** и **Мутация**.

Первый работает путем изменения двух (или более) программ (или древовидных структур) и объединения их каким-либо образом; последний путем изменения одной древовидной структуры.

Каждая итерация алгоритма называется **Поколением**. В конце каждого поколения, третья генетическая операция, называемая **Отбор** применяется для того, чтобы устранить худших членов в популяции. После применения этих операций алгоритм получает следующее поколение членов.

Условие остановки гарантирует, что алгоритм завершается. После того, как критерий остановки удовлетворяется, мы берем лучшее решение от конечной популяции. Одним из типичных применений этого типа алгоритма для решения задач оптимального поиска пути. В этих проблемах, каждый член в популяции представляет собой путь для достижения конкретной цели и имеет соответствующую стоимость.

Оптимизация запросов может быть сведена к задаче поиска, где СУБД должна найти оптимальный QEP в обширном пространстве поиска. Каждый план выполнения может рассматриваться в качестве возможной программы для решения проблемы поиска хорошего пути доступа для получения необходимых данных. Таким образом, в генетическом оптимизаторе, каждый член популяции является действительным планом выполнения запроса. Интуитивно понятно, что по мере развития населения, средняя стоимость плана членов уменьшается [3, 5].

3. Проблема большего запроса на соединение

Значительный рост объема данных, необходимых для того, чтобы принимать правильные решения, делает текущие оптимизаторы запросов неадекватные в некоторых ситуациях. Компании, предоставляющие банковские услуги являются хорошим примером типичного клиента, который нуждается в очень большой емкости для хранения очень больших и сложных конструкций баз данных. В таких пользователей действительно нужно использовать очень большие запросы на соединение (join queries), чтобы поддержать их в своих бизнес-решениях. Различные подходы были предложены для исправления этой ситуации [3, 5, 6].

Эвристические алгоритмы: пространство поиска сокращается с использованием оценок. Они, как

правило, очень быстрые, но редко находят оптимальное решение.

Случайные алгоритмы: случайная прогулка по пространству поиска выполняется для того, чтобы найти почти оптимальное решение. Различные политики приводят к различным алгоритмам, а именно итеративного усовершенствования, имитации отжига, гибридных алгоритмов и т.д.

Генетические алгоритмы: вдохновленные в естественном отборе, генетические алгоритмы пытаются найти оптимальное среди популяции. Эта популяция страдает от постоянных преобразований, осуществляется с использованием трех основных видов деятельности: отбор, сочетание и мутации.

Генетическое программирование и оптимизация запросов

Оптимизация запросов может быть сведена к задаче поиска, где СУБД должна найти оптимальный план выполнения запроса (QEP) в обширном пространстве поиска. Каждый QEP можно рассматривать в качестве возможного решения (или программы) для задачи нахождения хорошего пути доступа для извлечения данных, необходимых в запросе. Таким образом, в генетическом оптимизаторе запросов, каждый член популяции является QEP. Оптимизация запросов с участием большого количества объединений с использованием генетических алгоритмов был введен Беннетом Эль-Аль и испытано позже Steinbrunn и др. Покали, что это очень конкурентный подход [5, 7].

Генетический алгоритм и SQL запрос на соединение с 5 join

Имеется реляционные таблицы со следующими таблицами: Person, Orders, Product, Department, и Quantity с помощью подходящего набора данных инициализации. Ниже приводится простой запрос с соединением для извлечения из него данные на заданных определенных условиях:

```
select
p.LastName, p.FirstName, o.OrderNo, o.p_id,
pd.productname, pd.prd_id, s.sales_id,
dept.departname, qty.quantity
from Persone p
inner join Orders o on p.p_id = o.p_id inner join
Product pd on o.prd_id = pd.prd_id inner join
Sales s on s.prd_id = pd.prd_id, inner join
Department dpt on s.sales_id = dpt.sales_id inner join
Quentity qty on dpt.dept_id = qty.dept_id
and qty.prd_id = qty.prd_id order by p.LastName
group by dpt.dept_id;
```

Запустив простое соединение запроса в SQL Server 2012, получаем график, показывающий время, необходимое для выполнения запроса:

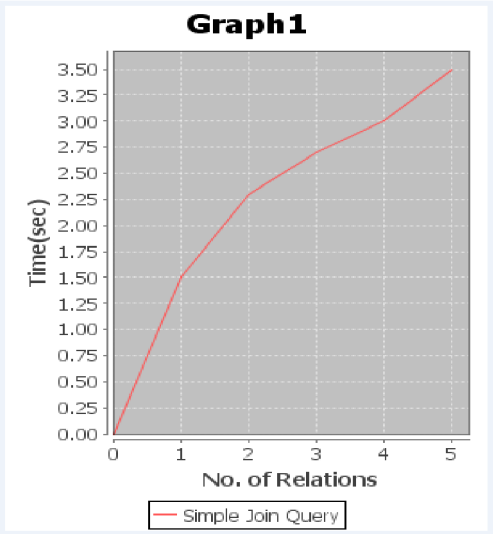


Рисунок 1. Зависимость количества join до времени выполнения запроса

Применение генетического алгоритма

Теперь применим генетический алгоритм на том же запросе с использованием 5 join.

Процедура:

Здесь мы рассмотрим приведенный выше запрос с N отношений, где (N = 5)

1. Принцип работы генетического алгоритма является создание популяции (решение пространства).

2. После создания населения установить No. Поколения и No. Потомка, который создает основу для населения.

3. Затем, взяв цикл для всех отношений в запросе создаем пространство решений т.е. популяцию путем случайного выбора соотношения, используя rand(). Здесь для его отношения должны быть соединены случайным образом выбирается другое соотношение учитывая Left Deep дерево [8].

4. После того, что в соответствии с принципом ГА выбор родителей из популяции и подсчет их хромосом.

5. Вычислить фитнесс значение каждой хромосомы и выбрать наиболее пригодную хромосому

6. Для расчета в фитнес-функцию следует рассматривать как P (x) = x2.

7. После того, как фитнесс значения рассчитывают для хромосом эти хромосомы скрещиваются друг с другом.

8. После завершения кросинговера происходит мутация.

Все вышеперечисленные шаги реализуются в SQL хранимой процедуре.

Заметка. Для следующей процедуры создайте таблица 'results', чтобы сохранить результат вычисления ГА, на котором выполняется QEP и вычисляется граф. Создать процедуру GeneticAlgowithjoins.

После выполнения вышеописанной процедуры в Sql Server 2010 время, необходимое для QEP для вышеуказанного запроса SQL с 5 операциями на присоединение показано на следующем графике:

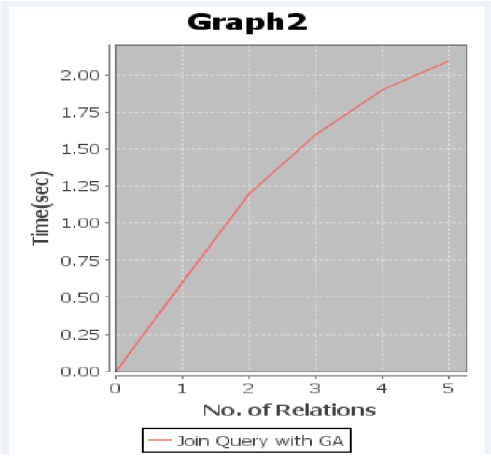


Рисунок 2. Зависимость количества join до времени выполнения запроса после применения ГА

Далее представлен график сравнения времени, необходимого для выполнения простого запроса до и после применения ГА.

На приведенном выше графике показано применение GA.

1. Фитнес значения показывают здоровье хромосом, отобранных в каждом Поколении.

2. График показывает значение фитнесс функции как 1,99 за 100 итераций.

3. В графике выше точки в нижней части (черные цветные точки) обозначают наилучшие значения пригодности, а точки над ними (синего цвета) обозначают средние фитнес значения в каждом поколении. Также показывает лучшие и средние значения в текущем поколении численно на самом верху.

4. На приведенном выше графике показано наилучшую пригодность в каждом поколении, показывает незначительный прогресс в снижении значения пригодности и среднее расстояние между отдельными генами в каждом поколении, что является хорошим показателем разнообразия населения.

5. Установка начального диапазона [1; 100], слишком мало разнообразия для алгоритма, чтобы добиться прогресса.

6. Фитнес значение индивидуума является значение функции пригодности для данного индивида. Лучшее значение пригодности для популяции является наименьшим значением пригодности для любого индивидуума в популяции.

7. В этом случае значение целевой функции каждой отдельной (хромосомы) является вычисление с помощью функции

$$P(x) = x2$$

8. Где x здоровье случайно выбранного индивида (хромосомы) в каждом поколении.

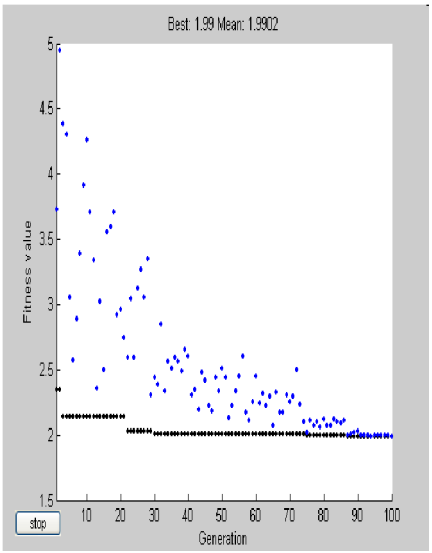
9. На каждом поколении случайным образом выбираются отдельные индивиды и фитнес значение рассчитывается.

10. Индивиды (хромосомы) с наилучшими значениями пригодности быстро размножаются, и также предотвращают генетический алгоритм от поиска других областей пространства решений, пропуская слабые неподходящие хромосомами.

11. Число индивидуумов с наилучшими значениями пригодности в текущем поколении, которые гарантированно выживут в следующем поколении. Генетический алгоритм использует индивидуумов в текущем поколении, чтобы создать детей, которые составляют следующее поколения.

12. Увеличение численности населения позволяет генетическому алгоритму поиск большего точек и тем самым получить лучший результат.

13. Лучшее соотношение фитнес быстро улучшается в начале поколений, когда особи находятся дальше от оптимального.



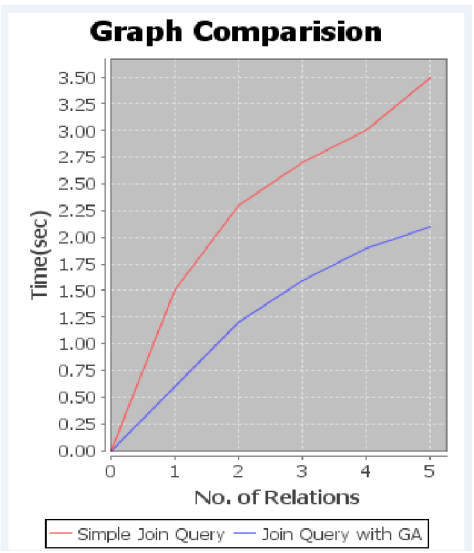


Рисунок 3 и 4. Применение ГА на данный запрос и сравнение с результатами без использования

Лучшее значение пригодности улучшается медленнее в последующих поколениях, население которых ближе к оптимальной точке.

14. Фитнес масштабирование преобразует сырые фитнес оценки, которые возвращаются функции пригодности для значений в диапазоне, который подходит для функции выбора. Функция выбора использует масштабируемые значения пригодности для выбора родителей следующего поколения.

4. Carquinyoli genetic optimizer (CGO)

Дается краткий обзор CGO оптимизаторов и доказывается, что они могут превзойти классические оптимизаторы, когда число запросов на объединение (join queries) в SQL-запросе велико. Объектом исследования являются оптимизаторы на основе генетического программирования. Для того чтобы выполнить анализ и представить новые идеи для улучшения оптимизации больших запросов на объединения, внедрено новый генетический оптимизатор, основанный на генетическом программировании. Данный оптимизатор называется Carquinyoli Genetic Optimizer (CGO) [9].

Основные компоненты CGO

Представлено четыре базовых структур, используемых CGO:

База данных. CGO предполагается работа на схеме реляционной базы данных. Как обычно, схема содержит набор отношений, которые связаны через первичные и внешние ключи (PK и FK, соответственно). Каждое отношение имеет связанный с ним мощность, которая соответствует количеству кортежей или записей, сохраненных в этом отношении.

Запрос. CGO предполагает SQL, является стандартным языком, используемым для выражения запроса. Тем не менее, для того, чтобы упростить оптимизатор, CGO работает с пониженным подмножеств стандартного языка SQL, который позволяет выбрать объект, проекции, соединения и операций сортировки. Язык псевдо-SQL, используемый CGO называется CGO-SQL.

План выполнения запроса (QEP). В QEPs являются членами популяции в генетическом оптимизатора. CGO предполагает QEP структурирован в виде tree-shaped дерева.

Популяция. В QEPs организованы в популяциях. Первоначально первая популяция создается с нуля случайным образом и, после этого QEPs в популяции изменяется или удаляется.

CGO Оперативное Описание

Процесс оптимизации начинается с создания исходного набора программ (в нашем случае, программа является QEP), как правило, называют членов исходной популяции по CGO. Начальная популяция содержит N элементов или программ. Каждый член в популяции представляет собой способ для достижения конкретной цели и имеет соответствующую стоимость. В случае оптимизации запросов каждый QEP представляет собой способ решения запроса предоставленной системе. Начиная с этой исходной популяции, как правило, создаются с нуля, две операции используются для производства новых членов в популяции:

Операции ***Кроссинговера***, которые сочетают в себе свойства существующих членов в популяции, а также *операции* ***Мутации***, которые вводят новые свойства в популяцию.

Для того, чтобы сохранить размер постоянной популяции. CGO выполняет операции кроссинговер C, выбирая два случайных QEPs в популяции каждый раз, и операции мутации, выбирая одну QEP, на поколение.

После применения генетических операций, операции кроссинговера породили новые 2C потомств и операции мутации породили M новых потомств. Таким образом, численность населения увеличилась, содержащая N + 2C + M элементов.

После этого шага, стоимость для каждого нового QEP в популяции вычисляется, а QEPs сортируются по их стоимости. Третья операция, как правило, называют, как выбор, используется, чтобы отбросить худшие подогнанные элементы, с помощью этой функции пригодности.

Этот процесс порождает новую популяцию, содержащую элементы N, называемый также поколение, которое включает в себя как старые и новые члены, которые выжили операцию выбора. Этот процесс повторяется итеративно для G поколений, пока условие остановки не заканчивает выполнение.

Каждая итерация возвращает новое население, которое эволюция популяции вернулась с предыдущей итерации.

После того, как критерий остановки удовлетворяется, лучшее решение берется из конечной популяции. Интуитивно понятно, что по мере развития населения, средняя стоимость QEP в популяции уменьшается.

Алгоритм 1, представляет собой простое описание главной процедурой, выполняемой CGO.

Во-первых, начальная популяция создается с нуля (строка 3).

После того, как P заполняется, мы входим в основной цикл в строке 4 и итерацию для G поколений.

Каждая итерация в этом цикле представляет эволюцию и создание нового поколения QEPs.

Для каждого поколения, кроссинговер и мутация операции применяются (строки 5 и 6).

После того, как все операции выполняются, текущая популяция объединяется с новыми QEPs порожденных операций (строка 7).

И операция выбора применяется сортировка членов в популяции по стоимости и отбрасывая наивысшим сметой расходов QEPs из полученного населения

Algorithm 1 CGO псевдо код

1: procedure CGO
main function
2: Population P, P1, P2, P3;
3: P = creatInitialPopulation ();
4: while (stop criterion is not met) do
5: P1 ← applyCrossoverOperations (P);
6: P2 ← applyMutationOperations (P);
7: P ← P ∪ P1 ∪ P2;
8: P ← applySelectionOperation (P);
9: end while
10: end procedure

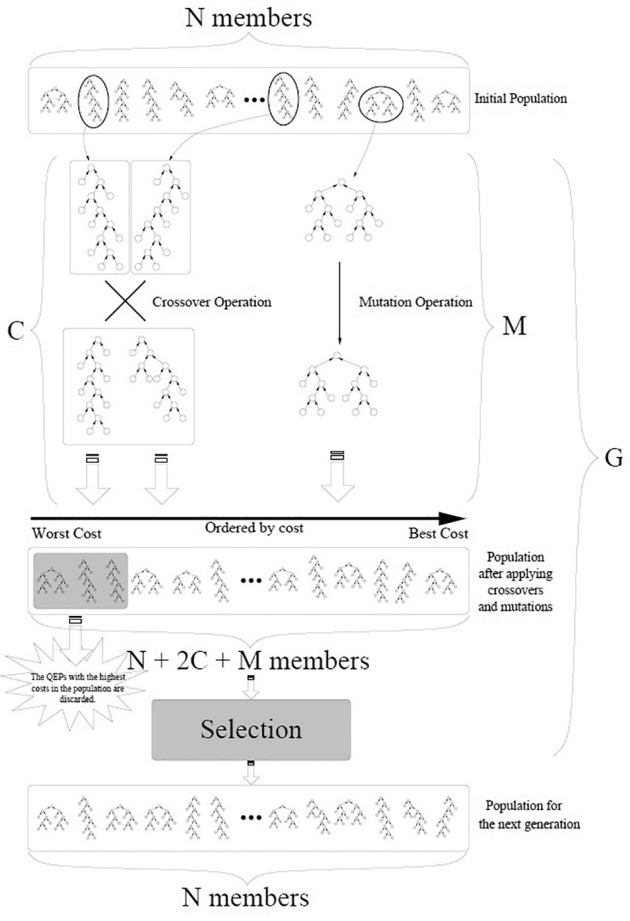


Рисунок 5. Общий вид поведения CGO

Выводы

В этой статье рассмотрено применение генетических алгоритмов в одной из самых важных задач оптимизации в области компьютерной науки, оптимизация запросов базы данных для большого join запроса.

Современные методы оптимизации запросов недостаточные для поддержки, некоторых формирующейся приложений, использующих базы данных. В этой статье решено проблему оптимизации запросов к базе данных с использованием генетического алгоритма.

Такой подход ГА подходит для больших баз данных, которые имеют огромный объем информации, и их цель состоит в том, чтобы выполнить выполнение больших запросов, имеющих большое количество запросов на соединение (более 16 join). Метод отбора и лучшее значение функции пригодности используется для обработки хромосом (особей), и мутационный процесс, который сокращает время и объем работы процессора в зависимости количества отношений.

Функция выбора присваивает более высокую вероятность отбора особям с более высокими значениями фитнесс функции. Диапазон вариаций значений влияет на производительность генетического алгоритма.

Наконец график, который показывает лучшее фитнес значение для хромосом в каждом поколении, может быть использован для изучения поведения функции и может быть полезным для определения значения функции. Этот метод можно использовать для оптимизации QEPs и затрат времени и средств, необходимых для выполнения соединения. CGO оптимизатор на основе генетического программирования способен справиться с большой проблемой запросов на соединение.

Будущие области применения

Современные методы оптимизации запросов являются недостаточными для поддержки некоторых новых приложений баз данных. Генетические алгоритмы, однако, идеально подходят для обработки, классификации и контроля сложных запросов для очень-больших и разнообразных данных.

Такой подход ГА может быть использован для оптимизации запроса, имеющий большое количество запросов на соединение (более чем 16 join). Опять же можно использовать, чтобы минимизировать объем памяти, необходимый для QEP.

Литература

1. A. Swami and A. Gupta. Optimization of large join queries. In Proc. of the 1988 ACM-SIGMOD Conference on the Management of Data, pages 8–17, Chicago, IL, June 1988.

2. M. Jarke and J. Koch. Query optimization in database systems. ACM Computing Surveys, 16(2): pages 111–152, June 1984.

3. Melanie Mitchell, “An introduction to Genetic Algorithms”, Prentice Hall of India, 2004.

4. Hsiung Sam, Matthews James, “An introduction to Genetic Algorithms”, 2008.

5. Y. E. Ioannidis and Y. Kang. Randomized algorithms for optimizing large join queries. In Proc. of the 1990 ACM-SIGMOD Conference on the Management of Data, pages 312–321, Atlantic City, NJ, May 1990.

6. Optimization algorithms [Електронный ресурс]. — Режим доступу: http://en.wikipedia.org/wiki/Category: Optimization_algorithms. — Дата доступу: 10.07.2016.

7. A. Swami and A. Gupta. Optimization of large join queries. In Proc. of the 1988 ACM-SIGMOD Conference on the Management of Data, pages 8–17, Chicago, IL, June 1988.

8. Left-depp vs. Bushy trees [Електронный ресурс]. — Режим доступу: http://www.deepdyve.com/lp/association-for-computing-machinery/left-deep-vs-bushy-trees-an-analysis-ofstrategy-spaces-and-its-2GTEPmpJUv. — Дата доступу: 05.06.2016.

9. V. Muntes-Mulero, J. Aguilar-Saborit, C. Zuzarte, and J-L. Larriba-Pey. Cgo: a sound genetic optimizer for cyclic query graphs. In Proceedings of the International Conference on Computer Science.