МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РЕАЛИЗАЦИЯ ПОИСКА МАКСИМАЛЬНОГО ПОТОКА СЕТИ ПРИ ПОМОЩИ РАСПРЕДЕЛЕННОГО ГЕНЕТИЧЕСКОГО АЛГОРИТМА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Кочетковой Юлии Андреевны

Научный руководитель
к. ф.-м. н., доцент _____ А. С. Иванов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2022

ВВЕДЕНИЕ

Распределенные системы присутствуют в нашей жизни каждый день. Они могут быть простыми или сложными, например, те, которые можно найти во всемирной паутине, социальных сетях, электронной коммерции и др. Распределенной системой может быть любая система, в которой аппаратные или программные компоненты разделены и могут обмениваться данными между собой, передавая сообщения через какую-то сеть. Основная мотивация для построение этих распределенных систем — это совместное использование ресурсов. Система может использовать ресурсы, которые не находятся в одном и том же месте, и их можно масштабировать. Однако распределенные системы обычно работают одновременно на разных устройствах, что требует их синхронизации, помимо того отдельные устройства могут давать независимые отказы. Это объясняет, почему данная область является сложной и изучается десятилетиями.

Традиционно большинство алгоритмов создавались и реализовывались для работы на одной машине. Со временем, с развитием многоядерных устройств и более быстрых сетей, некоторые из алгоритмов были заново изобретены для работы в распределенном режиме. Эволюционные алгоритмы получили распределенную версию спустя годы после их первой реализации. Они представляют собой разновидность метаэвристики, вдохновленной теорией эволюции Дарвина и являются эффективным методом решения многих типов проблем, в основном связаны с поиском и оптимизацией в различных областях.

Эволюционные алгоритмы — мощное решение для оптимизации различных задач. Они помогают найти подходящее решение в случае, если найти его строгими методами практически невозможно. Одним из самых популярных вариантов эволюционных алгоритмов является генетический алгоритм.

Иногда, если задача сложная, реализовать эффективный генетический алгоритм может оказаться невозможным, потому что вычисления занимают слишком много времени и невозможно хранить все необходимые данные в памяти. Чтобы преодолеть вторую проблему, можно хранить некоторые данные на диске и загружать их в память по мере необходимости. Это не решает проблем с производительностью, более того, приносит еще одну: чтение и, возможно, десериализация отдельного объекта с диска в память может сделать всю систему еще медленнее. Более того, все особи, которые когда-либо

проходили через алгоритм, были созданы с использованием одних и тех же генетических операторов и, таким образом, могут иметь слишком много общего, так что алгоритм будет ходить вокруг локальных оптимумов. Для решения этой проблемы мы можем использовать параллельный или распределенный генетический алгоритм.

Целью работы является построение параллельного распределенного генетического алгоритма поиска максимального потока сети.

Для достижения поставленной цели должны быть решены следующие задачи:

- построить параллельную реализацию распределенного генетического алгоритма;
- при помощи реализованного алгоритма найти оптимальное решение для заданного ориентированного целочисленного взвешенного графа;
- сравнить производительность реализованного алгоритма и классического генетического алгоритма.

Методологические основы исследования параллельных и распределенных генетических алгоритмов для решения проблемы поиска максимального потока сети представлены в работах O. Surakhi, E. Cantu-Paz, S. Medeiros.

Теоретическая значимость бакалаврской работы. Рассмотренные алгоритмы улучшают точность и ускоряют классический генетический алгоритм поиска максимального потока сети. Рассмотрен и реализован способ масштабирования алгоритма, что позволяет использовать данный алгоритм для решения более сложных задач.

Структура и объем работы Бакалаврская работа состоит из введения, 3 разделов, заключения, списка использованных источников и полного кода программы в качестве приложения. Общий объем работы — 72 страниц, из них 42 страницы — основное содержание, включая 12 рисунков,список использованных источников информации — 20 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Сети и задачи о потоках» посвящен общей информации о графах/сетях, приведены основные определения, описана проблема поиска максимального потока сети.

Подраздел 1.1 содержит основные определения и обозначения, касающиеся задачи поиска максимального потока сети.

Подраздел 1.2 содержит описание области применения и актуальности проблемы поиска максимального потока сети.

Второй раздел «Генетический алгоритм и его применение для поиска максимального потока сети» посвящен описанию генетического алгоритма и его применению в отношении проблемы поиска максимального потока сети.

Подраздел 2.1 содержит информацию о работе генетического алгоритма. Генетические алгоритмы имитируют процесс естественного отбора в природе. Для решения задачи, более оптимального с точки зрения некоторого критерия, процедуры описываются набором чисел или величин нечисловой природы. Поиск оптимального решения похож на эволюцию популяции индивидов, которые представлены их наборами генов.

Случайным образом генерируется начальная популяция – набор возможных решений задачи. Работа ГА – итерационный процесс, продолжающийся до тех пор, пока не выполнится заданное число поколений или какой-либо критерий остановки. На каждом поколении осуществляется отбор, пропорциональный функции приспособленности (коэффициенту выживаемости или Fitness function). В функциональной оптимизации сама целевая функция выступает в качестве функции приспособленности. Затем проводится скрещивание (кроссовер,рекомбинация) родительских генов. Полученные потомки заменяют родителей и подвергаются мутации. Полученное решение добавляется в популяцию, решение с наименьшим значением целевой функции – отбрасывается.

Общую схему такого алгоритма запишем следующим образом:

1. Выбрать начальную популяцию

$$S_k(0) = \{S_{k1}, S_{k2}, ...S_{kN}\}, \text{ где N - длина цепочки.}$$

Считать, что

$$f* = max\{f(S_k)|S_k \in S_k(t)\}t = 0$$

2. Пока не выполнен критерий остановки, делать следующее:
   — выбрать родителей S_{k1}, S_{k2} , из популяции $S_{k1}(t)$ (отбор);
   — построить новое решение с S'_k по S_{k1}, S_{k2} , (скрещивание);
   — модифицировать с S'_k(мутация);
   — если $f* < f(S'_k)$, то $f* := f(S'_k)$;
   — обновить популяцию и считать t := t + 1 .

Чтобы реализовать генетический алгоритм, надо сначала выбрать структуру для представления этих решений. Структура данных ГА состоит из одной или большего количества хромосом. Как правило, хромосома – это битовая строка либо векторное представление вещественных чисел. Генетический алгоритм оптимизации является множественно-вероятностным, т.е. он позволяет находить множество значений, которые примерно соответствуют искомому условию. Это существенно для решения задач с неявно выраженными максимумами или минимумами. Существует множество модификаций генетического алгоритма, которые отличаются в методах отбора, скрещивания, хромосомной кодировки. Традиционный (основной) ГА работает с двоичной строкой определенной длины, использует следующие свойства:

— на каждом поколении реализуется отбор пропорционально функции приспособленности;
— одноточечный кроссовер (скрещивание);
— мутация.

Члены популяции с более высокой приспособленностью с большей вероятностью чаще будут отбираться для проведения дальнейшей генетической обработки, чем особи с меньшей приспособленностью. После отбора проводим скрещивание, аналогично биологической эволюции. Пусть имеем двух родителей:

$$S_1 = (S_{11}, S_{12}, ..., S_{1N}) \text{ и } S_2 = (S_{21}, S_{22}, ..., S_{2N}) .$$

Случайным образом выбираем точку разрыва. Обе родительские особи разрываются на два сегмента по этой точке. Затем соответствующие сегменты различных родителей соединяются и получаются два потомка:

$$(S_{11}, ..., S_{1m}, S_{2,m+1}, ..., S_{2N}) \text{ и } S_2 = (S_{21}, ..., S_{1m}, S_{1,m+1}, ..., S_{2N}),$$

т.е. голова и хвост гена берутся от разных родителей. После проведения скрещивания выполняется мутация. В каждой строке, подвергшейся мутации,

произвольный ген заменяем на противоположный. Символы гена первого создаваемого потомка взяты от родителей S_1 или S_2 произвольно для любой символьной позиции. Второй индивидуум получает символы разности. Например, двое потомков S_1 и S_2 имеют следующий вид:

$$(S_{11}, S_{22}, S_{13}, S_{14}, ..., S_{2N}) \text{ и } (S_{21}, S_{12}, S_{23}, S_{24}, ..., S_{1N})$$

Подраздел 2.2 описывает пошаговое применение генетического алгоритма для поиска максимального потока сети.

Генетический алгоритм для поиска максимального потока сети выглядит следующим образом:

1. Генерируется популяцию случайных решений. Поток в дугах этого графа назначен случайным образом от 0 до пропускной способности дуги.
2. Выбираются пары лучших особей и скрещиваются для получения новой популяции того же размера. Выбор происходит с учетом фитнес-функции. Фитнес функция зависит от трех критериев:
   а) Количество сбалансированных вершин
   б) Сумма избыточного потока
   в) Насыщенность потока
3. Происходит скрещивание. Из родительских особей выбирается доминирующая вершина. Доминирующая вершина определяется путем сравнения уровня энергии одной и той же вершины каждой особи. Уровень энергии определяется сбалансированным состоянием и пропускной способностью через вершину.

   Если вершина сбалансирована и поток проходящий через эту вершину равен вместимости этой вершины, то уровень энергии такой вершины равен 0 и она считается стабильной. Вершина, которая не сбалансирована и поток, проходящий через нее меньше пропускной способности - имеет уровень энергии отличный от 0 и считается нестабильной. Вершина переносится вместе со своими ребрами.
4. Происходит мутация. Все ребра вершины имеют вероятность мутации. Для вершины, с уровнем энергии выше нуля, мутация всегда происходит в направлении снижения уровень энергии. Для вершины с нулевым уровнем энергии мутация происходит в любом направлении случайным образом.

При добавлении нового ребра идет проверка, было ли значение ребра добавлено ранее. Если нет, то значение ребра становится: ребро = ребро + корректировка мутации. Если предыдущее значение было определено во время добавление вершины ранее, то берется среднее арифметическое значений ребер с поправкой на мутацию.

Ребро = (старое ребро + новое ребро) / 2 + корректировка мутации.

Значение ребра - целое число. Когда (старое ребро + новое ребро) /2 представляет собой дробь 0.5, результат может быть округлен в меньшую сторону. Процесс ассимиляции позволяет вершине получать информацию о пропускной способности от смежных вершин, которые могут быть использованы в будущем поколении.

Подраздел 2.3 содержит описание параллельного генетического алгоритма. Параллельный генетический алгоритм — это такой алгоритм, который использует несколько генетических алгоритмов для решения одной задачи. Все эти алгоритмы пытаются решить одну и ту же задачу, и после того, как они выполнили свою работу, выбирается лучший из каждого алгоритма, затем выбирается лучший из них.

Эти генетические алгоритмы не зависят друг от друга, в результате они могут работать параллельно, используя преимущества многоядерного процессора. Каждый алгоритм имеет свой набор особей, в результате эти особи могут отличаться от особей другого алгоритма, потому что у них разная история мутаций/кроссоверов. Более того, некоторым особям с наивысшей фитнес - функцией разрешено «мигрировать» с одного алгоритма на другой. Иногда этого может быть достаточно, но когда задача очень сложна для решения или особь представляет собой сложную сущность, может понадобиться еще больше разнообразия внутри популяции.

Подраздел 2.4 посвящен описанию моделей распределенного генетического алгоритма. Распределенный генетический алгоритм на самом деле представляет собой параллельный генетический алгоритм, независимые алгоритмы которого выполняются на отдельных машинах. В этом случае каждый из этих алгоритмов может быть в свою очередь параллельным генетическим алгоритмом.

В разделе описаны основные модели распределенного генетического алгоритма, такие как:

1. модель ведущий-ведомый;
2. модель островов;
3. модель соты;
4. модель бассейн.

Третий раздел «Применяемый стек технологий» включает в себя полное описание программной реализации распределенного параллельного генетического алгоритма, решающего задачу поиска максимального потока сети в заданном графе.

Подраздел 3.1 содержит информацию о применяемых технологиях в программной реализации. Сборка проекта происходит при помощи Apache Maven, который фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM. Для упрощения работы с Redis используется Spring Framework. Координация всех компонентов системы реализуется при быстрого хранилища данных типа «ключ-значение» в памяти с открытым исходным кодом - Redis.

Подраздел 3.2 описывает реализацию распределенного параллельного генетического алгоритма для поиска максимального потока сети. Было создано приложение, рассчитывающее максимальный поток для заданного графа. Реализация алгоритма на языке Java версии 12. Основываясь на нескольких доступных моделях распределенных генетических алгоритмов, реализация в данной практической работе в основном вдохновлена моделью ведущий - ведомый, но с некоторыми характеристиками модели острова. Основное внимание уделялось тому, чтобы как можно больше операций выполнялось параллельно и асинхронно, а не одновременно для повышения общей производительности.

Приложение состоит из трех компонентов: Coordinator, Redis и Worker. Coordinator отвечает за прием задания, отправки ее в Redis и выбора лучшей особи из полученных от Worker. Redis отвечает за передачу сообщений. Worker отвечает за выполнение параллельного генетического алгоритма.

Выполнение программы происходит следующим образом:Coordinator получает задание, назначает ему идентификатор и отправляет его в специальную очередь TasksQueue в базе данных Redis. Redis является координатором для всей системы. Он хранит количество зарегистрированных Woker(счетчик) и передает сообщения между Worker и Coordinator. Счетчик необходим для того, чтобы понимать сколько Worker взяли задачу.

Так же в Redis хранятся очереди для обмена особями между Woker. В каждом Worker выполняется генетический алгоритм независимо друг от друга. В рамках реализации распределенного генетического алгоритма рабочие обмениваются между собой особями с заданной частотой и с заданным количеством особей. Каждому Worker случайным образом выбирается другой Worker, которому он будет отправлять особей. Когда Worker закончит работу и найдет лучшую особь среди своей популяции, он отправит ее в Coordinator и он выберет лучшую особь из полученных.

В ответ на отправленный запрос Coordinator присылает идентификатор, по которому можно получить результат, когда операция завершится. Отправка запроса и получение результата происходят через REST Endpoint.

Таким образом, после анализа всех операций было замечено, что операции генетического алгоритма независимы и могут выполняться параллельно и асинхронно.

Подраздел 3.3 содержит сравнение производительности классических алгоритмов и генетического. Было произведено сравнение генетического алгоритма с другими популярными алгоритмами для поиска максимального потока сети, которые были реализованы в прошлых курсовых работе, такие как: алгоритм Диница, Эдмондса-Карма и Форда-Фалкерсона, поиск максимального потока сети при помощи классического генетического алгоритма.

Общая сложность генетического алгоритма не так хороша, как у традиционных подходов. Из двух целей, балансировки и максимизации потока, самая трудная часть - это балансировка. Максимизация потока происходит довольно быстро, остальные поколения генерируются, в попытках сбалансировать вершины, медленно снижая поток всей популяции. В классическом генетическом алгоритме повышение эффективности может быть достигнуто путем разработки лучшего способа настройки параметров и модификации функций приспособленности, схем размножения, скрещивания и мутации. Распределенный параллельный алгоритм работает гораздо быстрее классического алгоритма.

Преимущество в скорости работы распределенного алгоритма достигается за счет того, что в нем можно снизить количество итераций.

ЗАКЛЮЧЕНИЕ

Используя параллельные и распределенные генетические алгоритмы, можно повысить производительность системы, использующей эволюционные алгоритмы. В любом случае, необходимо иметь ввиду, что эволюционные алгоритмы не гарантируют, что решение когда-либо будет найдено, и что не будет более оптимальных решений.

Одной из основных проблем, с которой приходится сталкиваться при использовании генетических алгоритмов, является предварительная конвергенция к подмножеству особей, которые доминируют над другими. Параллельные и распределенные генетические алгоритмы пытаются решить эту проблему, вводя различия между алгоритмами, которые заставляют их иметь разный набор особей. При использовании параллельных и распределенных генетических алгоритмов особи больше расходятся, в результате можно создать меньше особей, чем при использовании непараллельного генетического алгоритма, сохраняя при этом качество решения на том же уровне.

Основные источники информации:

1. Belding, C. The distributed genetic algorithm / C. Belding // Universi-ty of Michigan. — 2016. — С. 132
2. Medeiros, S. Distributed genetic algorithms for low-power, low-cost and small-sized memory devices / S. Medeiros // MDPI. — 2020. — С. 346
3. Вороновский, Г. Генетические алгоритмы, искусственные нейрон-ные сети и проблемы виртуальной реальности / Г. Вороновский // Основа. — 1997. — С. 765
4. Surakhi, O. A parallel genetic algorithm for maximum flow problem /O. Surakhi // International Journal of Advanced Computer Science and Applications. — 2017. — С. 218
5. Гладков, Л. Генетические алгоритмы / Л. Гладков // ФИЗМАТ-ЛИТ.—2010. — С. 643