Софийски университет „Св. Климент Охридски“
Факултет по математика и инфоратика

Проект по Мрежово програмиране

Тема: Търсене на зависимост на поведение на потребител чрез реализирането на високотехнологични полета с помощта на генетичен алгоритъм с алгоритъма HUIM-GA.

Съставил:
Ростислав Стоянов
ф-н 45244

Съдържание

1 Теоретично описание на решението на проблема

Мрежовата част от решението на проекта се базира на модела “Клиент - Сървър”, като в случая сървърът е от вид “concurrent” (т.е. всеки път, когато се получи връзка с клиент, сървърът създава нова нишка и делегира работата на нея, докато самият той остава в готовност да приеме нови връзки). Така при работата си сървъра създава TCP socket, който се “закача” за определен порт и чака заявки. При получаването на нова заявка, с цел да не се блокират други клиенти, се създава нова нишка работеща успоредно с главната.

За анализ на информацията от лог файла се използват 3 полета - Event context, Event name и IP address. Тъй като алгоритъмът работи с транзакции, то правим следното : с event context и event name уникално идентифицираме един предмет като от типа на случилото се действие зависи цената на този предмет, а предметите групираме по транзакции като нова транзакция имаме, когато се промени IP адресът, от който е произлязло действието). Получената по този начин база данни, състояща се от транзакции, подаваме на алгоритъма HUIM-GA, който намира множества от предмети със стойност в цялата транзакционна база данни по - голяма от подадено на алгоритъма цяло число - minimum utility threshold.

2 Описание на работата на алгоритъма

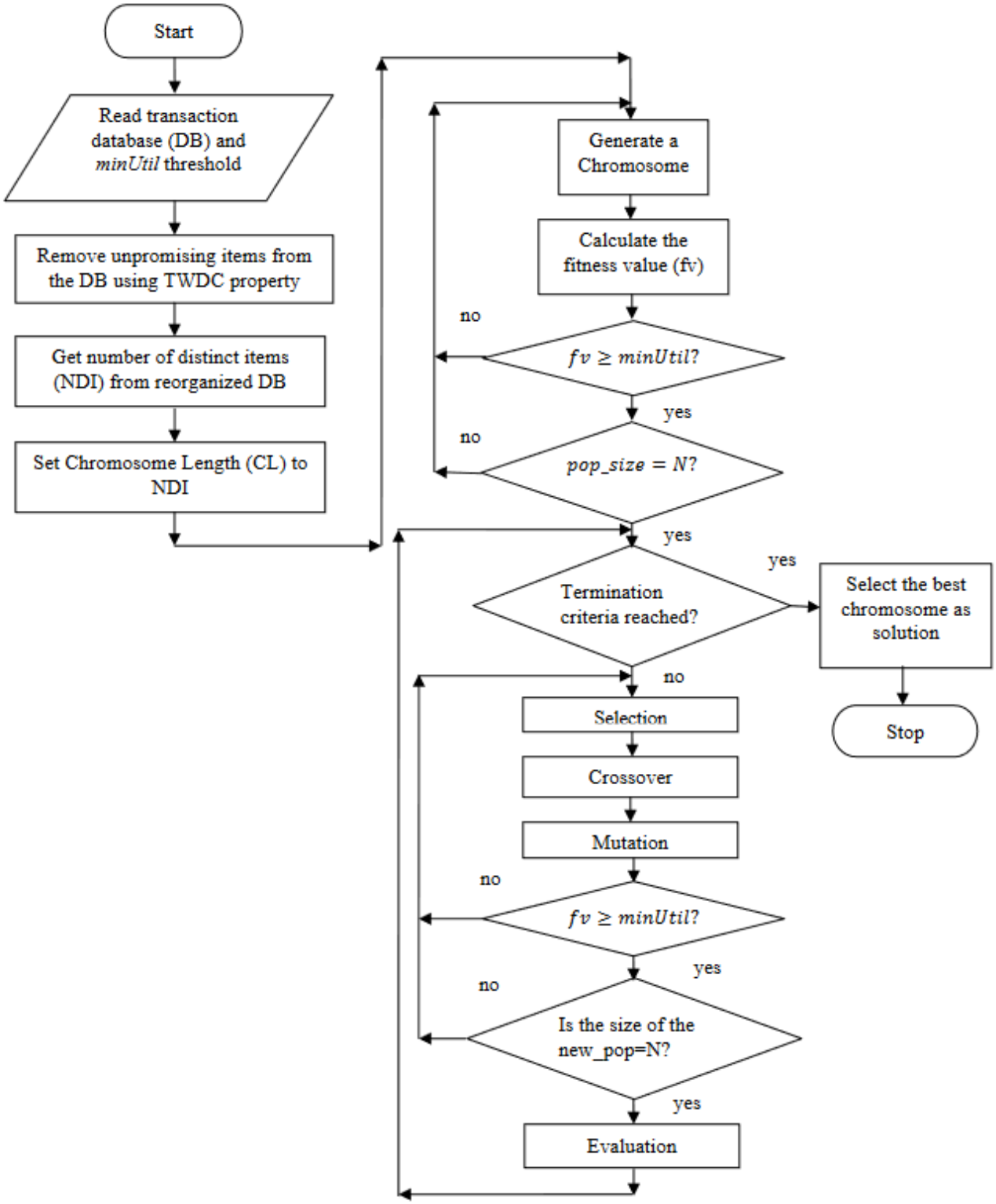
HUIM-GA е генетичен алгоритъм, който се използва за намиране на high - utility множества в база данни от транзакции. Алгоритъмът приема файл, където всеки ред представя 1 транзакция и е в следния формат : първо разделени с интервал са цели числа - предметите в транзакцията, следва двуеточие след което цената на цялата транзакция и след това цената на всеки предмет(отново са разделени с интервал).

Алгоритъмът започва работа като сканира базата данни за да пресметне TWU - transaction-weighted utilisation за всеки предмет. TWU(X) се дефинира като сумата на теглата на всички транзакции съдържащи X.[2]. След това базата данни се обхожда още веднъж като целта е да се намерят само обещаващите предмети т.е предметите с TWU>= minutil. Следващата стъпка е генерирането на начална популация(псевдокод илюстриращ генерирането е представен на фиг.1 [1]) и пресмятането на на скоростта на мутация.

```
population initialize (n, k)
begin
    i←0;
    for j← 0 to n do
    begin
        pop[j]←0;
    end
    while i ≤ kdo
    begin
        rand_no←rand(k);
        if pop[rand_no]≠1 then
        begin
            pop[rand_no]←1;
            i←i+1;
        end
    end
    return pop;
end
```

Фигура 1: Псевдокод представящ генерирането на началната популация

Следващата част от алгоритъма се изпълнява, докато не се достигнат критериите за терминиране, т.е фиксиран брой генерации, полученото решение е достатъчно добро сред последните n генерации (n - фиксирано число), ръчно разглеждане на решението или комбинация на всички посочени. Същността на този цикъл е прилагане на 3 генетични операции - селекция, кръстосване, мутация до постигане на необходимия резултат. Работата на целия алгоритъм е илюстрирана на блок-схемата на фигура 2.



Фигура 2: Блок - схема илюстрираща работата на алгоритъма

3 Начин на обработка на файл

Клиентът чете файла ред по ред и го изпраща на сървъра. Съответната WorkerThread нишка приема файла, като всеки ред бива обработен. Това става чрез използването на няколко класа, най - ключов, от които е класа Encoder, който изгражда съответствие между предмет и цяло пол. число (всеки предмет се идентифицира от контекста и вида на събитието). Предметите се групират по транзакции - разглеждайки структурата на дадения ни файл може да допуснем, че нова транзакция има всеки път, когато се смени origin ip адреса.

4 Намерени зависимост спрямо алгоритъма

Разглеждаме minUtil = 100. Спрямо зададените стойности на предметите, операцията разглеждане на файл в курс е операцията, която е с най - голяма важност за получаването на high - utility множества с голяма стойност.

5 Ръководство за инсталация

- Разархивирайте архива съдържащ решението на поставената задача;
- Използвайки терминал променете текущата си директория на : HUIM-GA/out/production/HUIM-GA;
- Изпълнете java Server.Server <port>, където port е портът, на който искате да се закачи сървърът;
- Изпълнете java Client.Client <host> <port>, където host е ip адресът на сървъра, а port е портът, на който слуша сървъра. При съответните запитвания въведете пътя до лог файла и minUtil.

6 Резултати

При наличие на подходящите условия - успешно свързване със сървър и коректно подадени входни данни, клиентското приложение печата на конзолата списък от high utility itemset - ове, със стойност по - голяма от minUtil. При настъпване на грешка, програмата връща информация за грешката и прекратява своята работа.

7 Приложение

Кодът на приложението и използваните библиотеки могат да бъдат намерени тук: https://github.com/RostislavStoyanov/NetworkProgramming.

8 Използвана литература:

Литература

[1] Kannimuthu S, Premalatha K. *Discovery of High Utility Itemsets Using Genetic Algorithm with Ranked Mutation*. Applied Artificial Intelligence,2014, 28(4): 337-359.

[2] Ying Liu, Wei-keng Liao, and Alok Choudhary. *A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets*. NCS (LNAI), vol. 3518, pp. 689–695. Springer, Heidelberg (2005).

[3] Java 8 API Specification. https://docs.oracle.com/javase/8/docs/api/

[4] SPMF Documentation. http://www.philippe-fournier-viger.com/spmf/index.php?link=documentation.php