МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет "Львівська політехніка"
Інститут комп’ютерних наук та інформаційних технологій

Кафедра САП

Звіт

до розрахунково графічної роботи

на тему: «Фреймворк EpochX»

З курсу: «Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні»

Виконав:
ст.гр. СПКс-11
Гуменний Л.О.

Прийняв:
Кривий Р.З.

ЛЬВІВ 2016

Мета роботи: Дослідити фреймворк EpochX.

Опис фреймворка EpochX.

EpochX - є простим у використанні фреймворком, але він має безліч додаткових функцій. EpochX є основою генетичного програмування з відкритим вихідним кодом . Він розроблений спеціально для виконання завдань аналізу еволюційного автоматичного програмування, та ідеально підходить для дослідників, яким потрібна розширена система для вивчення впливу нових операторів або процедур.

EpochX є основою Java для вивчення еволюції комп'ютерних програм, з використанням генетичних алгоритмів програмування. Версія 1.1+ підтримує 3-три популярні погляди - строго типізоване дерево GP (genetic programming) , контекстно-вільна граматика GP і граматична еволюція . Тим не менше, структура EpochX розширюється і дає можливість реалізувати абсолютно нові погляди в рамках еволюційної структури. Широкий діапазон можливостей для виконання, ініціалізації, кросовера, мутації ..., а також багато іншого в загальних тестових завдань в GP.

EpochX призначений в першу чергу для науковців, які працюють над генетичною теорією програмування. Науковці можуть отримати вигоду з використання EpochX якщо зацікавлені в розподілі глибини / довжини / різноманітності даних або великого кола статистичних даних в перспективі. Також EpochX буде корисним тим хто хоче мати дійсно динамічну систему, в якій статистичні дані доступні в режимі реального часу а параметри можуть бути оновлені. Оскільки EpochX є основою Java, передумовою є здатність програмувати на Java.

Особливості фреймворка EpochX.

- Повна підтримка 3-х популярних поглядів:
  - *строго типізоване дерево GP*

    EpochX забезпечує реалізацію строго типізованого дерева GP скороченоно називається XGP (Epoch X дерево GP). XGP зберігає і

маніпулює програми кандидатів в структуру дерева, після чого можна виконати оцінку шляхом запуску кожного вузла в дереві, щоб оцінити дітей.

Строкове представлення програм, які розробляються мають стандартний формат. Наприклад:

ADD (3.0, MUL (5,0, -1,0))

Ця конкретна мова називається Epox. Epox - є мова з розширеними новими функціями які розроблені і призначені для того щоб впоратися з будь-якими типами даних.

- *контекстно-вільна граматика GP*

  Пітер Уігхем продемонстрував на основі генетичного програмування підхід граматики , яку він назвав контекстно-вільна граматика GP (CFG-GP). Реалізація CFG-GP в EpochX називається XGR. XGR приймає граматику BNF (Бекуса-Наура) в якості одного з її входів , а потім розвиваються дерева, які утворюють синтаксично допустимі строки джерел в відповідно до цієї граматики.

  Теоретично, метод Уігхема уможливлює еволюціонувати програми на будь-якій мові, яка може бути визначена в BNF. Проте, на практиці це вимагає спосіб оцінки програмних фрагментів, щоб призначити бали. Тому для того, щоб розвивалась мова Java, Lisp або Ruby, деякі форми інтерпретатора для цієї мови не вимагаються. EpochX забезпечує спосіб підключення перекладачів для різних мов програмування, а також надає перекладачів з коробки для наступних мов:

  Java

  Ruby

  Groovy

  Epox

- *граматична еволюція*

Граматична еволюція (GE) є популярною граматикою на основі подання Оніла і Райана. Реалізація GE в EpochX називається XGE. XGE приймає граматику BNF в якості одного з входів , а потім розвиваються дерева, які утворюють синтаксично допустимі строки джерела в відповідно до цієї граматики.

- Відсутні неприваблівi файли параметрів.
- Повністю вставні компоненти і оператори.
- Динамічна конфігурація, яка може бути оновлена в середині проекту.
- Вбудовані моделі для багатьох поширених тестових завдань (в тому числі мультиплексори, мурахи, символьна регресія).
- Великий вибір вбудованих операторів.
- Простий механізм для запису нових операторів кросовера / мутації.
- Доступ до статистичних даних в режимі реального часу
- Документація, та керівництво у повній версії для Java.

Встановлення фреймворка EpochX

Щоб встановити фреймворк EpochX потрбіно попередньо мати встановлений Java JDK 6 або більш нову версію Java JDK.

Відповідно до загального стабільного програмного забезпечення, EpochX рухається в напрямку більш послідовного циклу випуску і версій системи нумерації . Версія 1.4 є останньою версією, і рекомендована для всіх нових користувачів.

На офіційному сайті EpochX на сторінці http://epochx.org/downloads.php потрібно завантажити файл epochx-1.4.1.zip та розпакувати його в зручне місце (рис.1.).

Рисунок 1 Сторінка завантаження epochx-1.4.1.zip

Також попередньо потрібно щоб був встановлений Eclipse Java Mars.

Після того як роз-архівували архів в нас є наступна структура файлів і папок (рис.2.)

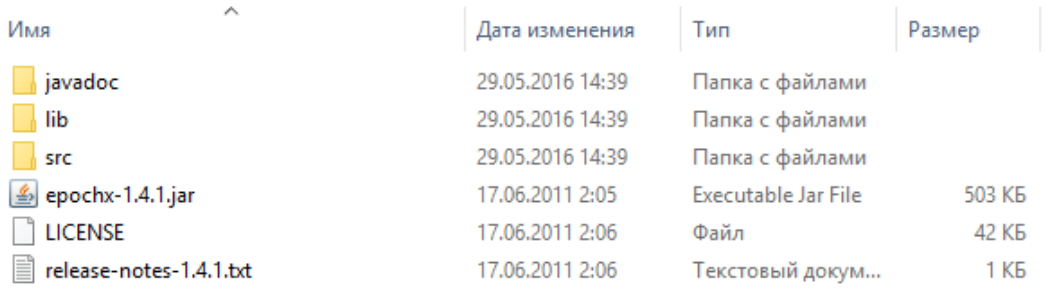

Рисунок 2 Вміст архіва epochx-1.4.1.zip

Папка Javadoc містить інструкції

Папка lib містить бібліотеки EpochX

Папка src містить вихідні коди.

Далі нам потрібно підключити бібліотеки EpochX до Eclipse Java Mars.

Завантажуємо Eclipse Java Mars вибираєм Project →Properties.

Переходимо на вкладку libraries →Add External JARs… і додаємо всі jar файли з папки lib і головний файл epochx-1.4.1.jar з коренья каталогу, та натиснути клавішу Apply → OK (рис.3)

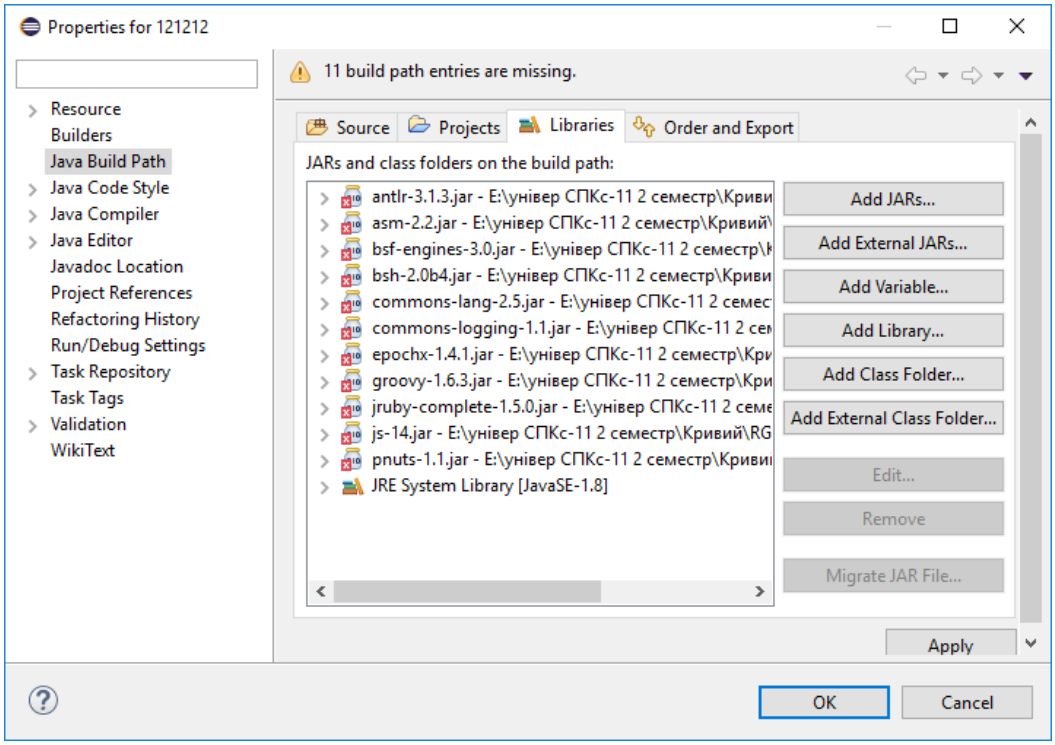

Рисунок 3 Додавання бібліотек EpochX

Ось і все підключення відбулось і ми можемо створюбвати проекти використовуючи нові бібліотеки від фраймворка EpochX.

Приклад програми написаної за допомогою фраймворка EpochX.

Для прикладу було взято метод точкового схрещування в генетичних алгоритмах.

Точкове схрещування відбувається наступним чином. Вибираються пари хромосом з батьківської популяції. Далі для кожної пари відібраних таким чином батьків розігрується позиція гена (локус) у хромосомі, що визначає так звану точку схрещування l_k. Якщо хромосома кожного з батьків складається з L генів, то очевидно, що точка схрещування l_k є натуральне число, менше L. Тому фіксація точки схрещування зводиться до випадкового вибору числа з інтервалу $[1,\ L-1]$. В результаті схрещування пари батьківських хромосом виходить наступна пара нащадків:

- нащадок, хромосома якого на позиціях від 1 до l_k складається з генів першого з батьків, а на позиціях від $l_k + 1$ до L - з генів другого з батьків;
- нащадок, хромосома якого на позиціях від 1 до l_k складається з генів другого з батьків, а на позиціях від $l_k + 1$ до L - з генів першого з батьків.

Дія оператора схрещування проілюстрована наступним прикладом(рис.4).

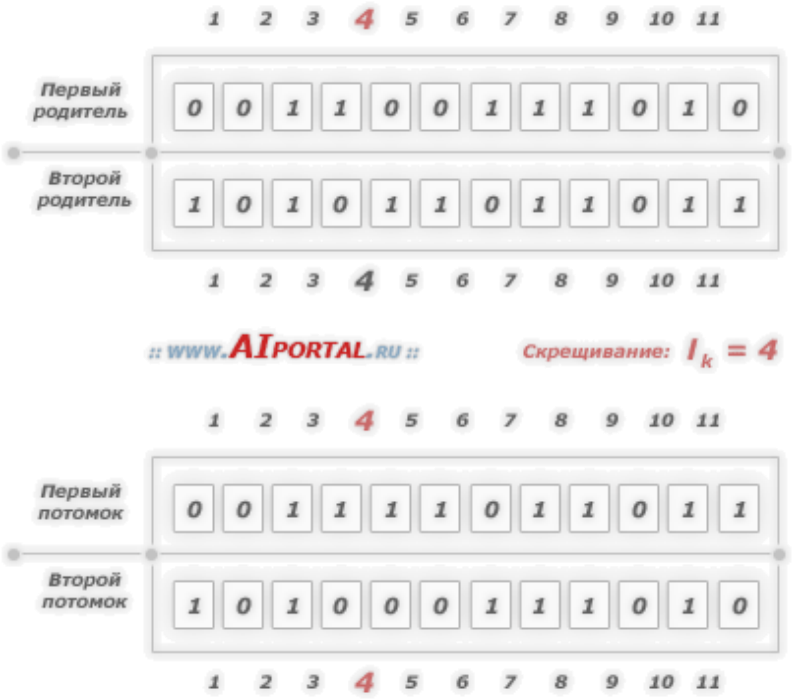

Рисунок 4. Ілюстрація точкового схрещення.

Напишемо програму яка реалізує точкове схрещення.

EpochX поставляється з набором вбудованих моделей, які визначають відповідні налаштування і функції для загальних тестових завдань. Щоб приступити до роботи, ми повинні отримати один з них, ми будемо використовувати 4-парності з використанням дерева GP. Ми також можемо знайти моделі паритету в пакеті org.epochx.gp.models.

Створюємо новий клас Java, з основним методом. Усередині основного методу нам просто необхідно додати наступні рядки коду для того, щоб почати розвиватися деякі програми GP.

```
1.  GPModel model = new EvenParity(4);
2.  Life.get().addGenerationListener(new GenerationAdapter(){
3.      public void onGenerationEnd() {
4.          Stats.get().print(StatField.GEN_NUMBER,
5.                            StatField.GEN_FITNESS_MIN,
6.                            StatField.GEN_FITTEST_PROGRAM);
7.      }
8.  });
9.  model.run();
```

Рядок 1 будує модель, модель визначають рядки 2-8 , у яких статистичні дані які ми хотіли б надрукувати кожне покоління і рядок 7 виконує нашу модель. Ми могли б залишити рядки 2-8 , і наша модель буде розвивалися, але

там не було б ніяких вихідних даних . Якщо ми використовуємо цей код то повинні переконатися , що імпортували правильний клас EvenParity.

Якщо виконати цей клас висновок консолі повинен бути потік з 3-х колонок, щось на кшталт:

```
0     5.0    OR(IF(OR(AND(OR(AND(D2 D3) AND(D2 D1)) ...
1     5.0    OR(IF(OR(AND(OR(AND(D2 D3) AND(D2 D1)) ...
2     3.0    IF(IF(NOT(IF(OR(IF(D3 D1 D3) NOT(D1)) ...
3     3.0    IF(IF(NOT(IF(OR(IF(D3 D1 D3) NOT(D1)) ...
4     2.0    IF(IF(NOT(IF(OR(IF(D3 D1 D3) NOT(D1)) ...
5     2.0    IF(IF(NOT(IF(OR(IF(D3 D1 D3) NOT(D1)) ...
6     2.0    IF(IF(NOT(IF(OR(IF(D3 D1 D3) NOT(D1)) ...
...   ...    ...
```

Перший стовпець має номер покоління, коли покоління 0 це є результатом ініціалізації. Потім другий і третій стовпці мінімальна придатність (тобто краща придатність при використанні стандартизованої придатності) і краща програма, в відповідності до рядків 5 і 6 нашого коду. Також можна експериментувати поля з різною статистикою, доступною в StatField класі.

Якщо все йде правельно, то ми повинні побачити мінімальне значення придатності яке поступово знижується до нуля в наступних поколіннях.

Це просто голий мінімум для запуску моделі, і в даний час ми використовуємо всі стандартні значення, як зазначено в GPModel класів, але тепер нам потрібно, забезпечити власний GPModel який має багато методів. Давайте встановимо розмір популяції, число поколінь і максимальну глибину програми.

```
1.  GPModel model = new EvenParity(4);
2.  model.setPopulationSize(500);
3.  model.setNoGenerations(100);
4.  model.setMaxDepth(8);
5.  Life.get().addGenerationListener(new GenerationAdapter(){
6.      public void onGenerationEnd() {
7.          Stats.get().print(GEN_NUMBER,
8.                            GEN_FITNESS_MIN,
9.                            GEN_FITTEST_PROGRAM);
10.     }
11. });
12. model.run();
```

Рядок 2, 3 і 4 були вставлені , для встановлення параметрів. Ще один цікавий параметр, який ми можемо встановити це число setNoRuns (INT) , який ускладнює виконання 50 або 100 експериментів.

У наступному коді ми змінимо оператор кросовера, селектор програм і генератор випадкових чисел.

```
1.  GPModel model = new EvenParity(4);
2.  model.setCrossover(new UniformPointCrossover(model));
3.  model.setProgramSelector(new TournamentSelector(model, 7));
4.  model.setRNG(new MersenneTwisterFast());
5.  Life.get().addGenerationListener(new GenerationAdapter(){
6.      public void onGenerationEnd() {
7.          Stats.get().printGenerationStats(GEN_NUMBER,
8.                                           GEN_FITNESS_MIN,
9.                                           GEN_FITTEST_PROGRAM);
10.     }
11. });
12. model.run();
```

Рядок 2, 3 і 4 були змінені , щоб встановити нові компоненти. Основна відмінність полягає в тому , що ми повинні пройти в екземплярі компонента для використання, і він є загальним для компонентів , щоб мати доступ до моделі, яка використовується, так як він може визначити відповідні параметри для компонента. Деякі компоненти також мають свої власні варіанти, які встановлюється з аргументами для конструктора або сетер-методів, таких як новий TournamentSelector , який ми створили , і який приймає розмір турніру в якості аргументу.

Повний лістинг програмного коду:

```
import static org.epochx.stats.StatField.*;

import org.epochx.gp.model.*;
import org.epochx.gp.op.crossover.OnePointCrossover;
import org.epochx.life.*;
import org.epochx.op.selection.TournamentSelector;
import org.epochx.stats.Stats;
import org.epochx.tools.random.MersenneTwisterFast;

public class Example1 {
    public static void main(String[] args) {
        // Construct the model.
        final GPModel model = new EvenParity(4);

        // Set parameters.
        model.setPopulationSize(500);
        model.setNoGenerations(100);
        model.setMaxDepth(8);

        // Set operators and components.
        model.setCrossover(new OnePointCrossover(model));
        model.setProgramSelector(new TournamentSelector(model, 7));
        model.setRNG(new MersenneTwisterFast());

        // Request statistics every generation.
        Life.get().addGenerationListener(new GenerationAdapter(){
```

```
            @Override
            public void onGenerationEnd() {
                  Stats.get().print(GEN_NUMBER, GEN_FITNESS_MIN,
GEN_FITTEST_PROGRAM);
            }
         });
         // Run the model.
         model.run();
     }
        }
```

Результат:

```
0       5.0     OR(NAND(NOT(AND(NAND(NOT(d2)  NAND(d0  d3))  NOT(AND(d2  d3))))  OR(NAND(AND(AND(d0  d1)
AND(d0 d1)) AND(NAND(d2 d3) OR(d0 d2))) NOT(OR(NOT(d2) AND(d3 d2))))) AND(OR(NOT(AND(AND(d1 d1) AND(d2
d1))) NOT(NAND(OR(d1 d1) NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
1       5.0     OR(NAND(NOT(AND(NAND(NOT(d2)  NAND(d0  d3))  NOT(AND(d2  d3))))  OR(NAND(AND(AND(d0  d1)
AND(d0 d1)) AND(NAND(d2 d3) OR(d0 d2))) NOT(OR(NOT(d2) AND(d3 d2))))) AND(OR(NOT(AND(AND(d1 d1) AND(d2
d1))) NOT(NAND(OR(d1 d1) NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
2       5.0     OR(NAND(NOT(AND(NAND(NOT(d2)  NAND(d0  d3))  NOT(AND(d2  d3))))  OR(NAND(AND(AND(d0  d1)
AND(d0 d1)) AND(NAND(d2 d3) OR(d0 d2))) NOT(OR(NOT(d2) AND(d3 d2))))) AND(OR(NOT(AND(AND(d1 d1) AND(d2
d1))) NOT(NAND(OR(d1 d1) NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
3       5.0     NAND(NAND(NOT(AND(NAND(NOT(d2) NAND(d0 d3)) NOT(AND(d2 d3)))) NOT(NAND(NOT(NAND(d0 d1))
NAND(NAND(d1  d0)  AND(d2  d1))))))  NAND(AND(AND(OR(AND(d0  d1)  NOT(d0))  NAND(NOT(d0)  AND(d0  d3)))
AND(NOT(OR(d0 d3)) NAND(NAND(d1 d0) OR(d2 d1)))) OR(AND(AND(OR(d0 d1) NAND(d3 d3)) OR(OR(d1 d2) NAND(d2
d1))) AND(OR(NOT(d1) OR(d1 d0)) OR(NAND(d0 d1) AND(d2 d1))))))
4       4.0     OR(NAND(NOT(AND(NAND(NAND(d3 d2) NAND(d0 d3)) NOT(AND(d2 d3)))) OR(NAND(AND(AND(d0 d1)
AND(d0 d1)) d1) NOT(OR(NOT(d2) AND(d3 d2))))) AND(OR(NOT(AND(AND(d1 d1) AND(d2 d1))) NOT(NAND(OR(d1
d1) NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
5       4.0     OR(NAND(NOT(AND(NAND(NAND(d3 d2) NAND(d0 d3)) NOT(AND(d2 d3)))) OR(NAND(AND(AND(d0 d1)
AND(d0 d1)) d1) NOT(OR(NOT(d2) AND(d3 d2))))) AND(OR(NOT(AND(AND(d1 d1) AND(d2 d1))) NOT(NAND(OR(d1
d1) NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
6       4.0     OR(NAND(NOT(AND(NAND(NAND(d3 d2) NAND(d0 d3)) NOT(AND(d2 d3)))) OR(NAND(AND(AND(d0 d1)
AND(d0 d1)) d1) NOT(OR(NOT(d2) AND(d3 d2))))) AND(OR(NOT(AND(AND(d1 d1) AND(d2 d1))) NOT(NAND(OR(d1
d1) NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
7       4.0     OR(NAND(NOT(AND(NAND(NAND(d3 d2) NAND(d0 d3)) NOT(AND(d2 d3)))) OR(NAND(AND(AND(d0 d1)
AND(d0 d1)) d1) NOT(OR(NOT(d2) AND(d3 d2))))) AND(OR(NOT(AND(AND(d1 d1) AND(d2 d1))) NOT(NAND(OR(d1
d1) NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
8       4.0     OR(NAND(NOT(AND(NAND(NAND(d3 d2) NAND(d0 d3)) NOT(AND(d2 d3)))) OR(NAND(AND(AND(d0 d1)
AND(d0 d1)) d1) NOT(OR(NOT(d2) AND(d3 d2))))) AND(OR(NOT(AND(AND(d1 d1) AND(d2 d1))) NOT(NAND(OR(d1
d1) NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
9       4.0     OR(NAND(NOT(AND(NAND(NOT(d2)  NAND(d0  d3))  NOT(AND(d2  d3))))  OR(NAND(AND(AND(d0  d1)
AND(d0 d1)) d1) NOT(OR(NOT(d2) AND(d3 d2))))) AND(OR(NOT(AND(AND(d1 d1) AND(d2 d1))) NOT(NAND(OR(d1
d1) NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
10      4.0     OR(NAND(NOT(AND(NAND(NOT(d2)  NAND(d0  d3))  NOT(AND(d2  d3))))  OR(NAND(AND(AND(d0  d1)
AND(d0 d1)) d1) NOT(OR(NOT(d2) AND(d3 d2))))) AND(OR(NOT(AND(AND(d1 d1) AND(d2 d1))) NOT(NAND(OR(d1
d1) NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
…
98      4.0     OR(NAND(NOT(AND(NAND(NOT(d2)  NAND(d0  d3))  NOT(AND(d2  d2))))  OR(NAND(AND(AND(d0  d1)
AND(d0 d1)) d1) NOT(OR(NOT(d2) AND(d3 d2))))) AND(OR(NOT(AND(AND(d1 d1) AND(d2 d1))) NOT(NAND(OR(d0
d1) NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
99      4.0     OR(NAND(NOT(AND(NAND(NAND(d3 d2) NAND(d0 d3)) NOT(AND(d2 d2)))) OR(NAND(AND(AND(d0 d1)
d0)  d1)  NOT(OR(NOT(d2)  AND(d3  d2))))) AND(OR(NOT(AND(AND(d1  d1)  AND(d2  d1)))  NOT(NAND(OR(d1  d1)
NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
100     4.0     OR(NAND(NOT(AND(NAND(NAND(d3 d2) NAND(d0 d3)) NOT(AND(d2 d3)))) OR(NAND(AND(AND(d0 d1)
AND(d0 d1)) d1) NOT(OR(NOT(d2) AND(d3 d2))))) AND(OR(NOT(AND(AND(d1 d1) AND(d2 d1))) NOT(NAND(OR(d1
d1) NOT(d1)))) AND(NOT(OR(AND(d0 d3) NOT(d3))) NOT(NOT(OR(d1 d2))))))
```

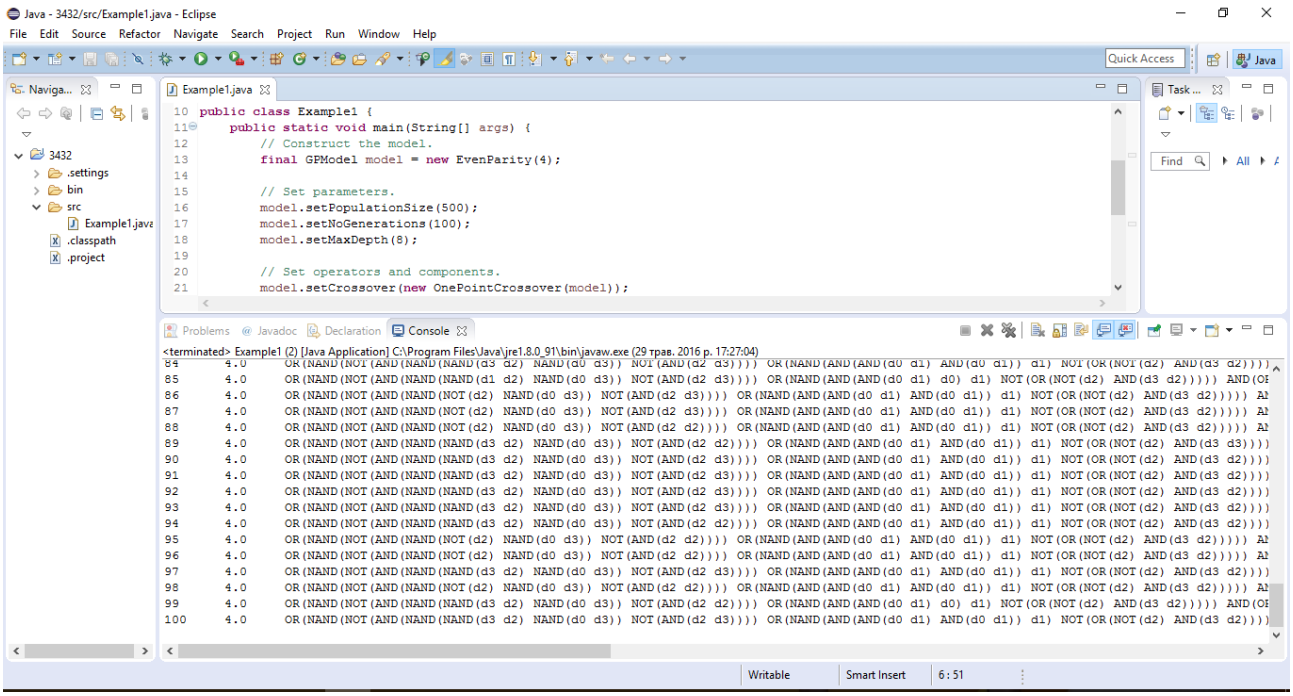

Рисунок 5 Робота програми

Висновки:

Було описано фреймворк EpochX, та його особливості, розглянуто та реалізовано процес його встановлення(підключення) і приклад роботи точкового схрещення на даному фреймворку.