МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

**Звіт до розрахунково-графічної роботи**

на тему:

«MOEA Framework»

з курсу:

«Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні»

Варіант - 1

Виконала:

ст. гр. СПКм-12

Андрусів М.І.

Перевірив:

Кривий Р. З.

Львів 2016

1. **Опис та основні особливості MOEA Framework**

Еволюційні алгоритми є дуже популярними підходами в багатоцільовій оптимізації. Сьогодні більшість еволюційних методів оптимізації застосовують схеми ранжування, які базу- ються на обробці множини Парето. Багатоцільові еволюційні алгоритми (MOEA) – це стохастичні методи оптимізації. Як і інші алгоритми оптимізації, MOEA використовують для знаходження паретооптимальних рішень для конкретної проблеми, але вони відрізняються підходами. Більшість еволюційних алгоритмів використовують концепцію планування в своїх діях . Оптимізаційний механізм багатоцільового еволюційного алгоритму (MOEA) дуже схожий на еволюційний алгоритм (EA), за винятком використання відносин домінування. На кожній ітерації значення цілі розраховується індивідуально, а потім використовується для визначення домінування відносин всередині сукупності, щоб вибрати потенційно ефективні рішення. Як правило, багатоцільові алгоритми (MOEA) мають справу з двома основними питаннями. Перше питання полягає в тому, щоб наблизитись до множини Парето. Друге питання – як зберегти різноманітність серед рішень в отриманому наборі. Ці два питання стали загальним критерієм для більшості поточних алгоритмів порівняння продуктивності. Різноманітність набору рішень дає більше можливостей для ОПР. Генетичні алгоритми, такі як генетичний алгоритм сортування-II (NSGA-II), що не домінує, і SPEA-2, стали стандартними підходами, хоча деякі зі схем, які основані на оптимізації групи рішень та імітації нормалізації, є багатозначним. Головна перевага генетичних алгоритмів – їх можна використати для пошуку рішень у дуже великих та складних просторах пошуку з різними цілями.

МOЕA Framework є відкритим вихідним кодом еволюційних обчислень у бібліотеці Java, яка спеціалізується на багатоцільовій оптимізації. Він підтримує безліч багатокритеріальних еволюційних алгоритмів (MOEAs), в тому числі генетичні алгоритми, генетичне програмування, граматичну еволюцію, диференціальну еволюцію та оптимізацію багаточисельних частинок. В результаті, він був використаний для проведення численних порівняльних досліджень з метою оцінки ефективності, надійності і керованості в найсучасніших MOEA.

МOЕA Framework являє собою розширену основу для швидкого проектування, розробки, виконання та статистичного тестування еволюційних алгоритмів багатокритеріальності (MOEAs). Він має 25 різних впроваджений MOEAs і більше 80 аналітичних тестових завдань. Крім того, інтегрується з JMetal, платформою і мовами програмування інтерфейсу для пошукових алгоритмів (PISA), і Borg МЕ бібліотеки, щоб забезпечити доступ до всіх популярних MOEAs.

Нові проблеми виникають в МОЕА Framework з використанням одного або декількох змінних рішення про той чи інший тип. Це включає в себе загальні уявлення, такі як виконувані рядки, дійсні числа і перестановки. Після того, як проблема визначена, користувач може оптимізувати вирішення за допомогою будь-якого з підтримуваних MOEAs.

Отже, МЕ Framework є вільним і відкритим вихідним кодом Java бібліотеки для розробки та експериментування з багатокритеріальних еволюційних алгоритмів (MOEAs) та інші алгоритми багатокритеріальної оптимізації загального призначення. МЕ Framework підтримує генетичні алгоритми, диференціальної еволюції, генетичне програмування, граматична еволюція, і багато іншого. Ряд алгоритмів надаються поза цим, в тому числі NSGA-II, NSGA-III, е-МЕ, GDE3, ПАЕС, PESA2, SPEA2, IBEA, SMS-EMOA, SMPSO, OMOPSO, CMA-ES, і МЕ / D. Крім того, МЕ Framework надає інструменти, необхідні для швидкого проектування, розробки, виконання та статистичної оптимізації тестування алгоритмів.

**Його основні Features це:**

* Швидкі, надійні реалізації багатьох алгоритмів.
* Інструменти для побудови і статистичного тестування нових алгоритмів оптимізації.
* Професійна підтримка для підприємств.
* Дозвільні ліцензії з відкритим вихідним кодом.
* Повністю документовані тестування вихідного коду.

МOЕA Framework є єдиною відомою основою для еволюційних обчислень, який забезпечує підтримку для аналізу чутливості. В якості альтернативи, аналіз чутливості вимірює стійкість до МOЕA на зміни його параметрів. Шляхом вимірювання чутливості кожного МOЕA, МOЕA Framework може визначити керуючі параметри для кожного МЕ і дати вказівки для тонкої настройки параметрів.

**Використання**

Статистичний аналіз забезпечується аналізатором. Аналізатор може відображати хвилини, середнє, максимальне і сумарні значення для кількох показників, в тому числі гіпероб'єма, відстань між поколіннями, перевернутої відстані, поколінння адитивного е-індикатора, інтервал і внесок. Крім того, Крускала U-тести забезпечують статистичні результати значущості. У наведеному нижче прикладі ми порівнюємо алгоритми з використанням гіпероб'єм метрики.

**Приклад.**

String problem = "UF1";

String[] algorithms = { "NSGAII", "GDE3", "eMOEA" };

//setup the experiment

Executor executor = new Executor()

    .withProblem(problem)

    .withMaxEvaluations(10000);

Analyzer analyzer = new Analyzer()

    .withProblem(problem)

    .includeHypervolume()

    .showStatisticalSignificance();

//run each algorithm for 50 seeds

for (String algorithm : algorithms) {

    analyzer.addAll(algorithm,

        executor.withAlgorithm(algorithm).runSeeds(50));

}

//print the results

analyzer.printAnalysis();

1. **Процес встановлення**

Нюансів при встановленні немає. Потрібно скачати фреймворк, розархівувати його і добавити jar файл як бібліотеку до проекту

**Приклад 1**

public class Example1 {

public static void main(String[] args) throws IOException {

String problem = "UF1";

String[] algorithms = { "NSGAII", "GDE3", "eMOEA" };

//setup the experiment

Executor executor = new Executor()

.withProblem(problem)

.withMaxEvaluations(10000);

Analyzer analyzer = new Analyzer()

.withProblem(problem)

.includeHypervolume()

.showStatisticalSignificance();

//run each algorithm for 50 seeds

for (String algorithm : algorithms) {

analyzer.addAll(algorithm,

executor.withAlgorithm(algorithm).runSeeds(50));

}

//print the results

analyzer.printAnalysis();

}

}

**Результат виконання:**

GDE3:

    Hypervolume:

        Min: 0.4389785065649592

        Median: 0.4974186560778636

        Max: 0.535166930530847

        Count: 50

        Indifferent: [NSGAII]

eMOEA:

    Hypervolume:

        Min: 0.35003766343295073

        Median: 0.47633216464734596

        Max: 0.53311360537305

        Count: 50

        Indifferent: []

NSGAII:

    Hypervolume:

        Min: 0.38868701091987184

        Median: 0.5040946740799506

        Max: 0.5371138081508796

        Count: 50

        Indifferent: [GDE3]

**Приклад 2**

Instrumenter instrumenter = new Instrumenter()

    .withProblem("UF1")

    .withFrequency(100)

    .attachElapsedTimeCollector()

    .attachGenerationalDistanceCollector();

new Executor()

    .withProblem("UF1")

    .withAlgorithm("NSGAII")

    .withMaxEvaluations(10000)

    .withInstrumenter(instrumenter)

    .run();

Accumulator accumulator = instrumenter.getLastAccumulator();

for (int i=0; i<accumulator.size("NFE"); i++) {

  System.out.println(accumulator.get("NFE", i) + "\t" +

      accumulator.get("Elapsed Time", i) + "\t" +

      accumulator.get("GenerationalDistance", i));

}

**Результат виконання:**

NFE    Time      Generational Distance

100    0.0256    0.7616

200    0.0421    0.6645

300    0.0543    0.4847

400    0.0636    0.4425

500    0.0713    0.4178

**Приклад 3.**

NondominatedPopulation result = new Executor()

    .withProblem("UF1")

    .withAlgorithm("NSGAII")

    .withMaxEvaluations(10000)

    .run();

**Результат виконання:**

Objective1  Objective2

0.9465      0.0466

0.1355      2.0513

0.1403      1.9960

0.2520      0.5120

**Висновок**

Ознайомились із основними поняттями при роботі з фреймворками, навчились практично застосовувати при виконанні індивідуального завдання.