|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| СПКс-11 | РГР | **The MOEA Framework** |  |  |
| Киценюк М.Л. | |
| № залікової: 1508503 | |
| Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні | | Викладач: | |
| Кривий Р.З. | |

**Мета роботи**

Ознайомитись з можливостями та характеристиками The MOEA Framework.

**Теоретичні відомості**

The MOEA Framework – це безкоштовна Java-бібліотека еволюційного моделювання з відкритим вихідним кодом, яка спеціалізується на багатокритеріальній оптимізації. Вона підтримує безліч багатокритеріальних еволюційних алгоритмів (MOEAs), в тому числі генетичні алгоритми, генетичне програмування, граматичні еволюції, диференціальні еволюції, метод рою часток та ін. В результаті, вона була використана для проведення численних порівняльних досліджень з оцінки ефективності, надійності і керованості найсучаснішої технології MOEAs.



Рис.1. Логотип фреймворку

**Основні характеристики фреймвоку:**

* **Швидкий та надійний у реалізації багатьох найсучасніших багатокритеріальних еволюційних алгоритмів.**

MOEA включає в себе NSGA-II, NSGA-III, є-MOEA, є-NSGA-II, PAES, PESA2, SPEA2, IBEA, SMS-EMOA, GDE3, SMPSO, OMOPSO, CMA-ES і MOEA/D. Ці алгоритми оптимізовані для продуктивності, що робить їх доступними для додатків з високою продуктивністю. Також підтримує JMetal і бібліотеки PISА. Забезпечує доступ до 30 алгоритмів багатокритеріальної оптимізації.

* **Можливості розширення за допомогою користувацьких алгоритмів, проблем та операторів.**

MOEA framework надає базовий набір алгоритмів, тестових проблем та операторів пошуку, але також може бути легко розширений, для включення додаткових компонентів. Через інтерфейс постачальника послуг (SPI), нові алгоритми та проблеми можуть бути ефективно інтегровані в MOEA framework.

* **Підтримує технології ведучий-ведений, модель локальної конкуренції на спарювання і гібридне розпаралелювання;**
* **Інструменти для побудови та статистично тестування нових алгоритмів оптимізації;**
* **Професійна підтримка бізнесу;**
* **Відкритий вихідний код.**

The MOEA Framework має ліцензію під безкоштовної та відкритою GNU Lesser General Public License, version 3. Це дозволяє кінцевим користувачам вивчати, модифікувати і вільно розширювати MOEA Framework.

* **Повністю задокументований і перевірений вихідний код**

Вихідний код повністю документований і часто оновлюється,  
щоб залишатися послідовним після будь-яких змін. Крім того, надається детальне керівництво користувача, яке забезпечує зручне використання MOEA Framework.

* **Велика підтримка доступна онлайн**

Так як проект активно підтримується, постійно відбувається виправлення багів та додавання нових функцій. Для цього сайт надає інструменти для повідомлення про помилки, запитів на нові функції та відповідей на питання користувача.

* **Понад 1200 тест-кейсів для забезпечення достовірності**

Кожен реліз фреймворку піддається  
ретельному тестуванню і контролю якості. І, якщо які-небудь помилки виявлено після тестування вони негайно виправляються і випускаються патчі.

Мета MOEA – надати велику колекцію алгоритмів та інструментарію для багатокритеріальної оптимізації.

**Алгоритми The MOEA Framework**

Даний фреймворк має найбільшу колекцію алгоритмів MOEAs , ніж у будь-якої іншої бібліотеки. Крім цих вбудованих алгоритмів, нові алгоритми можуть бути легко створені, використовуючи існуючі компоненти.

*Таблиця 1. Алгоритми MOEA*

|  |  |
| --- | --- |
| ***Назва*** | ***Опис*** |
| **AbYSS** | Багатокритеріальний пошук врозсип 1 |
| **Borg MOEA** | Адаптивний багатооператорний пошук з ε-домінуванням і ε-Прогресом спрацьовуючих перезавантажень3 |
| **CellDE** | Клітинний генетичний алгоритм з диференціальною еволюцією1 |
| **CMA-ES** | Коваріаційна матриця стратегії еволюції адаптації |
| **DBEA** | Покращений еволюційний алгоритм, заснований на декомпозиції |
| **DENSEA** | Еволюційний алгоритм дублюючої ліквідації недомуніючого сортування1 |
| **ECEA** | Еволюційний алгоритм ε- обмеження |
| **ε-MOEA** | ε-домінючий еволюційний алгоритм |
| **ε-NSGA-II** | NSGA-II з ε-домінування, Рандомізованимии перезавантаженнями і Адаптивним розміром населення |
| **FastPGA** | Швидкий генетичний алгоритм Паретто1 |
| **FEMO** | Справедливий еволюційний багатокритеріальний оптимізатор2 |
| **GDE3** | Узагальнена диференціальна еволюція |
| **HypE** | Алгоритм оцінки багатовимірного об`єму для багатовимірної оптимізації2 |
| **IBEA** | Еволюційний алгоритм на основі індикатора |
| **MOCell** | Багатокритеріальний клітинний генетичний алгоритм1 |
| **MOCHC** | Багатокритеріальний алгоритм CHC 1 |
| **MOEA/D** | Багатокритеріальний еволюційний алгоритм з декомпозицією |
| **NSGA-II** | Генетичний ІІ алгоритм на основі недомінантного сортування |
| **NSGA-III** | Генетичний алгоритм на основі недомінантного сортування початкової точки |
| **OMOPSO** | Багатокритеріальна оптимізація методом рою частинок |
| **PAES** | Еволюційна стратегія на основі статичного Парето |
| **PESA2** | Селекційний алгоритм на основі обгортки Парето |
| **Random** | Випадковий пошук |
| **RVEA** | Еволюційний алгоритм направлений опорним вектором |
| **SEMO2** | Простий еволюційний багатокритеріальний оптимізатор2 |
| **SHV** | Алгоритм орієнтований на багатовимірному об`ємі на основі вибірки2 |
| **SIBEA** | Еволюційний алгоритм на основі звичайного індикатору2 |
| **SMPSO** | Багатокритеріальної Оптимізація Методом Рою Частинок з обмеженою швидкістю |
| **SMS-EMOA** | S-метрична вибірка MOEA |
| **SPAM** | Алгоритм встановлення переваги для багатокритеріальної оптимізації2 |
| **SPEA2** | Еволюційний алгоритм на основі сили |
| **VEGA** | Генетичний алгоритм оцінки вектора |

***Примітки:***

1 - алгоритми надаються бібліотекою JMetal (в комплекті з фреймворком).  
2 - алгоритми надаються бібліотекою PISA.  
3 – доступний JAR-плагін з borgmoea.org.

***Мета-алгоритми***

Мета-алгоритми є обгортками навколо існуючих алгоритмів (враппери) для забезпечення додаткових функціональних можливостей .

*Таблиця 2. Мета-алгоритми MOEA*

|  |  |
| --- | --- |
| ***Назва*** | ***Опис*** |
| **Adaptive Time Continuation** | Періодично перезавантажує алгоритм, можливо, для адаптації параметрів |
| **Epsilon Progress Continuation** | Моніторинг прогресу пошуку, викликавши перезавантаження, якщо пошук застоюється |
| **Checkpoints** | Періодично зберігає стан алгоритму, щоб відновити перервані запуски |

**Налаштування MOEA Framework**

При розробці за допомогою даної бібліотеки, необхідно правильно налаштувати середовище java для того щоб забезпечити доступ до всіх ресурсів. Щоб допомогти у цьому процесі, вихідні коди включають в себе необхідні файли для імпортування безпосередньо в Eclipse. Покрокове налаштування:

1) Скачати бібліотеку та розархівувати отриманий файл http://moeaframework.org/downloads.html

2)Для імпортування в Eclipse, спочатку необхідно вибрати

File->Import…(рис.2).

3)Після цього з’явиться спливаюче вікно (рис.3).

4)Запевніться, що пункт General->Existing Projects вибраний та натискаємо Next (рис.4).

5)У наступному вікні за допомогою кнопки “Browse”, що знаходиться біля “Select root directory”(рис.5) вибираємо папку MOEAFramework-2.10 (яка було розархівована у 1 кроці), у якій знаходиться завантажена бібліотека(рис.6), клікаємо ОК та Finish.



Рис.2. File->Import

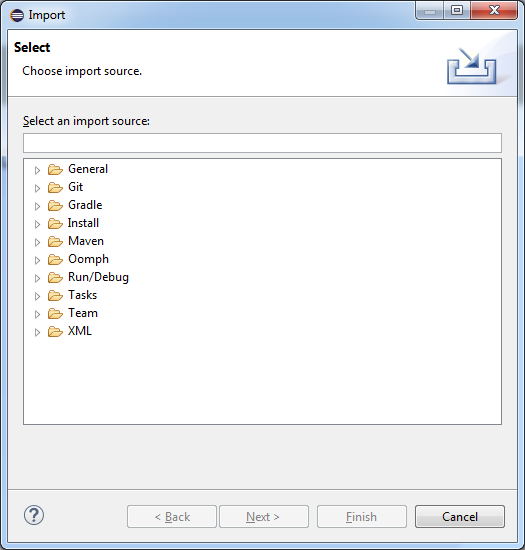


Рис.3. Спливаюче вікно

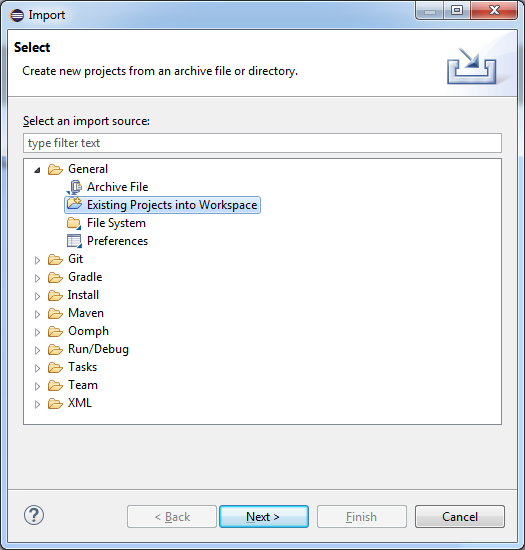


Рис.4. Вибір пункту General->Existing Projects

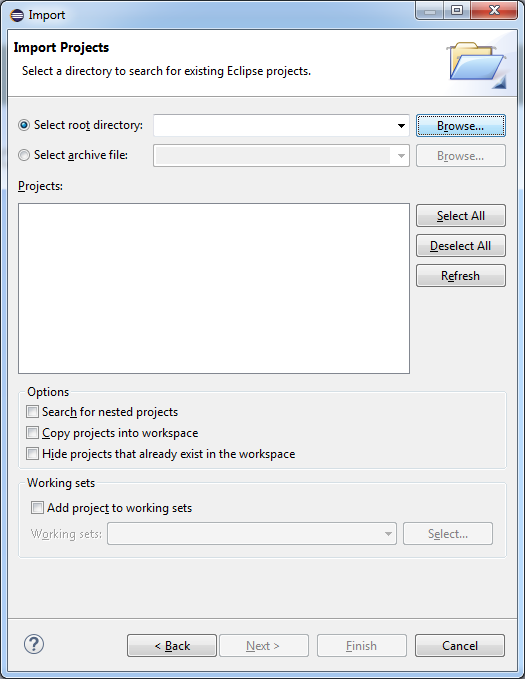


Рис.5. Select root directory

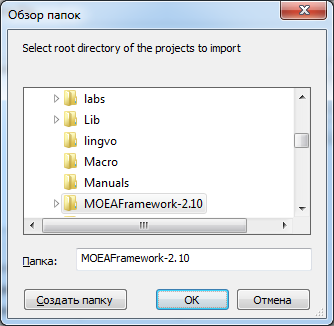


Рис.6. Вибір папки MOEA Framework

**Приклад програми 1**

Задача комівояжера (комівояжер — бродячий торговець; англ. Travelling Salesman Problem, TSP) полягає у знаходженні найвигіднішого маршруту, що проходить через вказані міста хоча б по одному разу. В умовах завдання вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій тощо) і відповідні матриці відстаней, вартості тощо.

В даному прикладі демонструється розв'язок задачі на основі карти з 76 містами, розробленої Манфредом Падберґом та Ґіованні Рінальді, двох відомих дослідників даної проблеми. Графічний інтерфейс користувача показує шлях, визначений даним фреймворком (рис. 7.), червоним кольором. Оптимальний маршрут має довжину 108159.

**Код програми**

**PR76.java**

package moeaframework.example.ga.tsplib;

import java.io.IOException;

import java.io.InputStream;

public class PR76Example {

/\*\*

Завантаження прикладу проблеми комівояжера

\* @param args the command line arguments

\* @throws IOException if an I/O error occurred

\*/

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

InputStream is = null;

try {

is = PR76Example.class.getResourceAsStream("pr76.tsp");

if (is == null) {

System.err.println("Unable to find the file pr76.tsp");

System.exit(-1);

}

TSP.solve(is);

} finally {

if (is != null) {

is.close();

}

}

}

}

**TSP.java**

public class TSP {

/\*\*

\*. Задати колір членів популяції

\*/

private static final Color lightGray = new Color(128, 128, 128, 64);

/\*\*

\* Коневертування рішення MOEA Framework до {@link Tour}.

\* @param solution the MOEA Framework solution

\* @return the tour defined by the solution

\*/

public static Tour toTour(Solution solution) {

int[] permutation = EncodingUtils.getPermutation(

solution.getVariable(0));

// increment values since TSP nodes start at 1

for (int i = 0; i < permutation.length; i++) {

permutation[i]++;

}

return Tour.createTour(permutation);

}

/\*\*

\* Зберігає{@link Tour} у рішення MOEA Framework

\*

\* @param solution the MOEA Framework solution

\* @param tour the tour

\*/

public static void fromTour(Solution solution, Tour tour) {

int[] permutation = tour.toArray();

// значення декремента, щоб отримати перестановку

for (int i = 0; i < permutation.length; i++) {

permutation[i]--;

}

EncodingUtils.setPermutation(solution.getVariable(0), permutation);

}

public static class TSPProblem extends AbstractProblem {

/\*\*

\*Приклад проблеми \*/

private final TSPInstance instance;

/\*\*

\* TSP евристичного за сприяння оптимізації процесу .

\*/

private final TSP2OptHeuristic heuristic;

/\*\*

\* Конструює нову проблему оптимізації для проблеми комівояжера

\* instance.

\*

\* @param instance the TSP problem instance

\*/

public TSPProblem(TSPInstance instance) {

super(1, 1);

this.instance = instance;

heuristic = new TSP2OptHeuristic(instance);

}

@Override

public void evaluate(Solution solution) {

Tour tour = toTour(solution);

// застосування еврист. і збереження та модифікація туру

heuristic.apply(tour);

fromTour(solution, tour);

solution.setObjective(0, tour.distance(instance));

}

@Override

public Solution newSolution() {

Solution solution = new Solution(1, 1);

solution.setVariable(0, EncodingUtils.newPermutation(

instance.getDimension()));

return solution;

}

}

/\*\*

\* Вирішує TSPLIB під час відображення графічним інтерфейсом проблеми

\* оптимізації

\*

\* @param instance the TSPLIB instance to solve

\*/

public static void solve(TSPInstance instance) {

TSPPanel panel = new TSPPanel(instance);

panel.setAutoRepaint(false);

// створення інших компонентів на дисплеї

StringBuilder progress = new StringBuilder();

JTextArea progressText = new JTextArea();

JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL\_SPLIT);

splitPane.setTopComponent(panel);

splitPane.setBottomComponent(new JScrollPane(progressText));

splitPane.setDividerLocation(300);

splitPane.setResizeWeight(1.0);

// відображення панелі на вікні

JFrame frame = new JFrame(instance.getName());

frame.getContentPane().setLayout(new BorderLayout());

frame.getContentPane().add(splitPane, BorderLayout.CENTER);

frame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

frame.setSize(500, 400);

frame.setLocationRelativeTo(null);

frame.setIconImages(Settings.getIconImages());

frame.setVisible(true);

// створення проблеми оптимізації і еволюційного алгоритму

Problem problem = new TSPProblem(instance);

Properties properties = new Properties();

properties.setProperty("swap.rate", "0.7");

properties.setProperty("insertion.rate", "0.9");

properties.setProperty("pmx.rate", "0.4");

Algorithm algorithm = AlgorithmFactory.getInstance().getAlgorithm(

"NSGAII", properties, problem);

int iteration = 0;

// завантаження еволюційного алгоритму

while (frame.isVisible()) {

algorithm.step();

iteration++;

// видалення існуючого туру на дисплеї

panel.clearTours();

// відображення популяціх сірими лініями

if (algorithm instanceof EvolutionaryAlgorithm) {

EvolutionaryAlgorithm ea = (EvolutionaryAlgorithm)algorithm;

for (Solution solution : ea.getPopulation()) {

panel.displayTour(toTour(solution), lightGray);

}

}

// відображення поточного оптимального рішення червоною лінією

Tour best = toTour(algorithm.getResult().get(0));

panel.displayTour(best, Color.RED, new BasicStroke(2.0f));

progress.insert(0, "Iteration " + iteration + ": " +

best.distance(instance) + "\n");

progressText.setText(progress.toString());

// перерисування вікна комівояжера

panel.repaint();

}

}

/\*\*

\* Завантаження прикладу проблеми комівояжера

\*/

public static void solve(File file) throws IOException {

solve(new TSPInstance(file));

}

/\*\*

\* Завантаження прикладу проблеми комівояжера

\*/

public static void solve(Reader reader) throws IOException {

solve(new TSPInstance(reader));

}

/\*\*

\* Завантаження прикладу проблеми комівояжера

\*/

public static void solve(InputStream stream) throws IOException {

solve(new TSPInstance(new InputStreamReader(stream)));

}

}

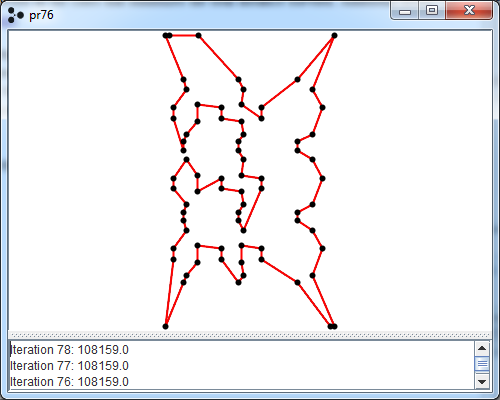


Рис.7. Розв'язок задачі комівояжера

**Приклад програми 2.**

Задача пакування рюкзака ( Knapsack problem) — задача комбінаторної оптимізації. Задача полягає у наповнені рюкзака, що здатен витримати деяку максимальну масу, предметами, кожен з яких має вартість (або корисність) та масу. Необхідно обрати об'єкти в такий спосіб, аби максимізувати сумарну вартість (або користь), але не перевищити максимально припустиму масу.

У цьому прикладі ми вирішуємо багатокритеріальний варіант завдання про пакування рюкзака . Для цього прикладу , ми повинні вибрати , які елементи ( 100 доступні ) помістити у два рюкзаки. Кожен рюкзак визначає свою власну вагу і прибуток для кожного пункту . Вхідна інформація зчитується з файлу.

**Код програми**

public class Knapsack implements Problem {

/\*\*

\* Кількість рюкзаків

\*/

private int nsacks;

/\*\*

\* Кількість речей

\*/

private int nitems;

/\*\*

\*

\* Прибуток від речі у рюкзаку

\*/

private int[][] profit;

/\*\*

\* Вага

\*/

private int[][] weight;

/\*\*

\* Ємність ваги рюкзака

\*/

private int[] capacity;

//зчитування з файлу

public Knapsack(File file) throws IOException {

this(new FileReader(file));

}

public Knapsack(InputStream inputStream) throws IOException {

this(new InputStreamReader(inputStream));

}

public Knapsack(Reader reader) throws IOException {

super();

load(reader);

}

/\*\*

\* Завантаження проблеми з спеціального рідера

\*/

private void load(Reader reader) throws IOException {

Pattern specificationLine = Pattern.compile("knapsack problem specification \\((\\d+) knapsacks, (\\d+) items\\)");

Pattern capacityLine = Pattern.compile(" capacity: \\+(\\d+)");

Pattern weightLine = Pattern.compile(" weight: \\+(\\d+)");

Pattern profitLine = Pattern.compile(" profit: \\+(\\d+)");

CommentedLineReader lineReader = null;

String line = null;

Matcher matcher = null;

try {

lineReader = new CommentedLineReader(reader);

line = lineReader.readLine(); // стрічка, що містить специфікацію проблеми

matcher = specificationLine.matcher(line);

if (matcher.matches()) {

nsacks = Integer.parseInt(matcher.group(1));

nitems = Integer.parseInt(matcher.group(2));

} else {

throw new IOException("knapsack data file not properly formatted: invalid specification line");

}

capacity = new int[nsacks];

profit = new int[nsacks][nitems];

weight = new int[nsacks][nitems];

for (int i = 0; i < nsacks; i++) {

line = lineReader.readLine(); // стрічка, що містить "="

line = lineReader.readLine(); // стрічка, що містить "knapsack i:"

line = lineReader.readLine(); // стрічка, що містить ємність

matcher = capacityLine.matcher(line);

if (matcher.matches()) {

capacity[i] = Integer.parseInt(matcher.group(1));

} else {

throw new IOException("knapsack data file not properly formatted: invalid capacity line");

}

for (int j = 0; j < nitems; j++) {

line = lineReader.readLine(); // стрічка, що містить "item j:"

line = lineReader.readLine(); // стрічка, що містить вагу item

matcher = weightLine.matcher(line);

if (matcher.matches()) {

weight[i][j] = Integer.parseInt(matcher.group(1));

} else {

throw new IOException("knapsack data file not properly formatted: invalid weight line");

}

line = lineReader.readLine(); // прибуток item

matcher = profitLine.matcher(line);

if (matcher.matches()) {

profit[i][j] = Integer.parseInt(matcher.group(1));

} else {

throw new IOException("knapsack data file not properly formatted: invalid profit line");

}

}

}

} finally {

if (lineReader != null) {

lineReader.close();

}

}

}

@Override

public void evaluate(Solution solution) {

boolean[] d = EncodingUtils.getBinary(solution.getVariable(0));

double[] f = new double[nsacks];

double[] g = new double[nsacks];

// розрахунки прибутку і ваг

for (int i = 0; i < nitems; i++) {

if (d[i]) {

for (int j = 0; j < nsacks; j++) {

f[j] += profit[j][i];

g[j] += weight[j][i];

}

}

}

// перевірка чи якісь ваги перевищують ємність

for (int j = 0; j < nsacks; j++) {

if (g[j] <= capacity[j]) {

g[j] = 0.0;

} else {

g[j] = g[j] - capacity[j];

}

}

// negate the objectives since Knapsack is maximization

solution.setObjectives(Vector.negate(f));

solution.setConstraints(g);

}

@Override

public String getName() {

return "Knapsack";

}

@Override

public int getNumberOfConstraints() {

return nsacks;

}

@Override

public int getNumberOfObjectives() {

return nsacks;

}

@Override

public int getNumberOfVariables() {

return 1;

}

@Override

public Solution newSolution() {

Solution solution = new Solution(1, nsacks, nsacks);

solution.setVariable(0, EncodingUtils.newBinary(nitems));

return solution;

}

@Override

public void close() {

}

}

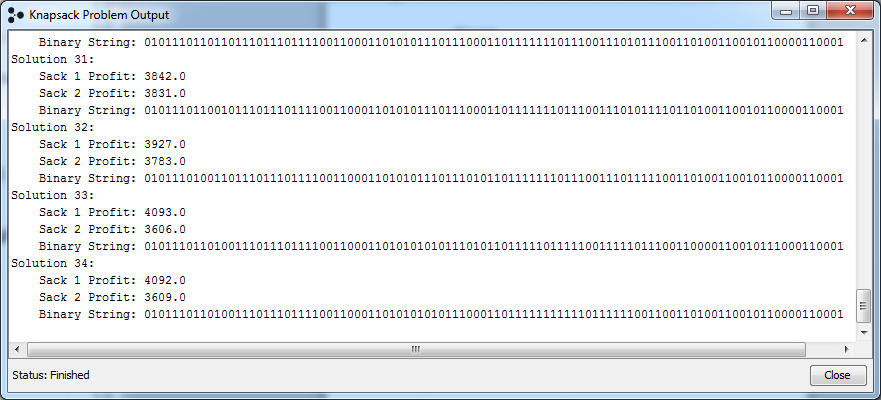


Рис.8. Розв'язок задачі пакування рюкзака

Так як це є багатокритеріальний варіант задачі, в нас вийшов більше ніж один оптимальний розв’язок. Бінарні стрічки вказують на те, яка річ включена в рюкзак(1), або виключена з нього (0).

**Висновок**

Під час виконання даної розрахунково-графічної роботи я розглянув можливості, характеристики та алгоритми The MOEA Framework. Розробив покроковий урок налаштування даної бібліотеки для Eclipse та навів приклад програми з використанням фреймворку на мові Java.