МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет “Львівська політехніка”

Кафедра САПР

Розрахункова робота

На тему:

“Ознайомлення із фреймворком JGAP ”

з курсу “Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні”

Варіант 5

Виконала:

ст. гр. КНC-13

Ладченко О.А.

Прийняв:

Кривий Р.З

Львів – 2017

**Мета**

Ознайомитись із зданим фреймворкоми та навести приклади його використання.

**Індивідуальне завдання (Варіант 5)**

Ознайомитись із фреймворком JGAP**.** Навести теоретичні відомості по роботі із даним фреймворком. Навести конкретні приклади його використання та результати виконання даних прикладів.

**Теоретичні відомості**

JGAP є це Генетичне програмування і Генетичні алгоритми написані на JavaВін забезпечує основні генетичні механізми, які можуть бути легко використані для застосування еволюційних принципів до вирішення завдань. Генетичний алгоритм є потужним засобом для вирішення проблем, з величезним простором розчину, що має обмежений час і потужність процесора.

JGAP був розроблений, щоб бути дуже простим у використанні «з коробки», а також призначений для високим ступенем модульності, так що більш підприємливі користувачі можуть легко вставляються призначені для користувача генетичні оператори, такі як мутації або кросинговеру та інших допоміжних компонентів.

.**Встановлення**

Щоб встановити JGAP, спочатку необхідно переконатись, щопринаймні Java 5 Runtime Environment (JRE) встановлено у вашій системі. (Для цього можна використати команду java --version в командній стрічці (консолі) що покаже вашу версію JRE.). Якщо у вас відсутнє JRE, скачайте та встановіть останню версію для вашої операційної системи, з http://java.sun.com.

Далі вам необхідно завантажити JGAP http://jgap.sourceforge.net/Завантажте останню версію (зараз це, ZIP файл jgap\_3.6.3\_full). Та включити набір бібліотек в проект на java.

**Configurator та Viewer**

Після огляду фреймворку, було виявлено що фреймворк безінтерфейсний. Щоб запустити певний алгоритм на операційній системві Windows потрібно скористатися бат файлом. Для зміни параметрів при розвязуванні задачі потрібно відкрити відповідний java файл і змінити необхідні парамети. Після чого запустити бат файл для запуску алгоритму вирішення задачі.

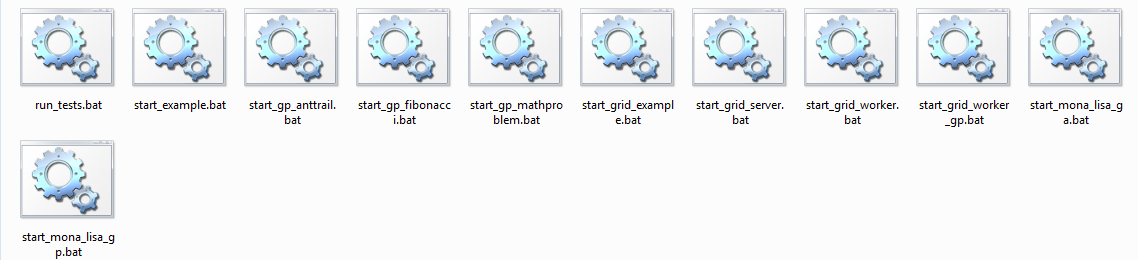


Рис.1 Список бат файлів

Для всіх алгоритмів відсутні бат файли для запуску. Тому потрібно написати їх самому або запускати через консоль.

**Розуміння** **та використання JGAP**

Щоб використовувати JGAP в додатку, є п'ять основних речей, які ви повинні зробити:

1. Планування хромосоми.
2. Реалізувати «фітнес-функції».
3. Налаштування об'єкту конфігурації.
4. Створення популяції потенційних рішень.
5. Розвиватися населення!

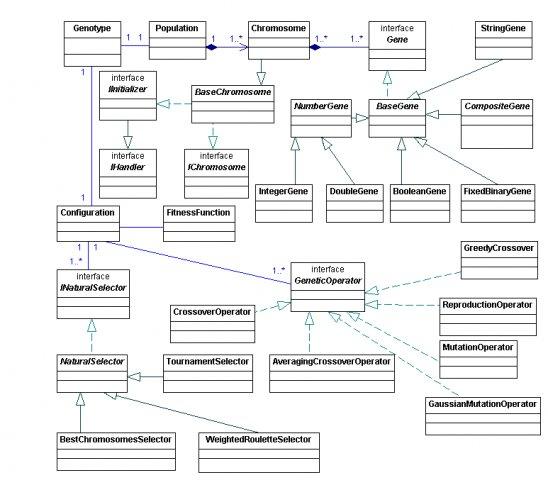


Рис. 2 Діаграма класів JGAP

**Приклади та результати**

**Приклад 1. Genetic Leonardo**

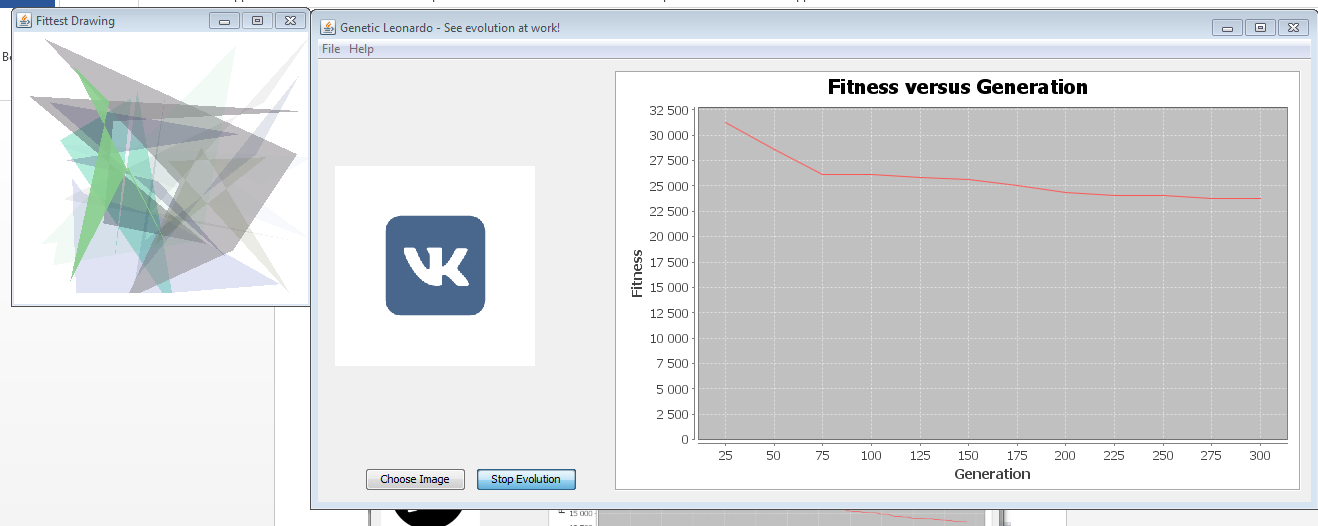


Рис. 3 Робота алгоритму після ~300 популяцій

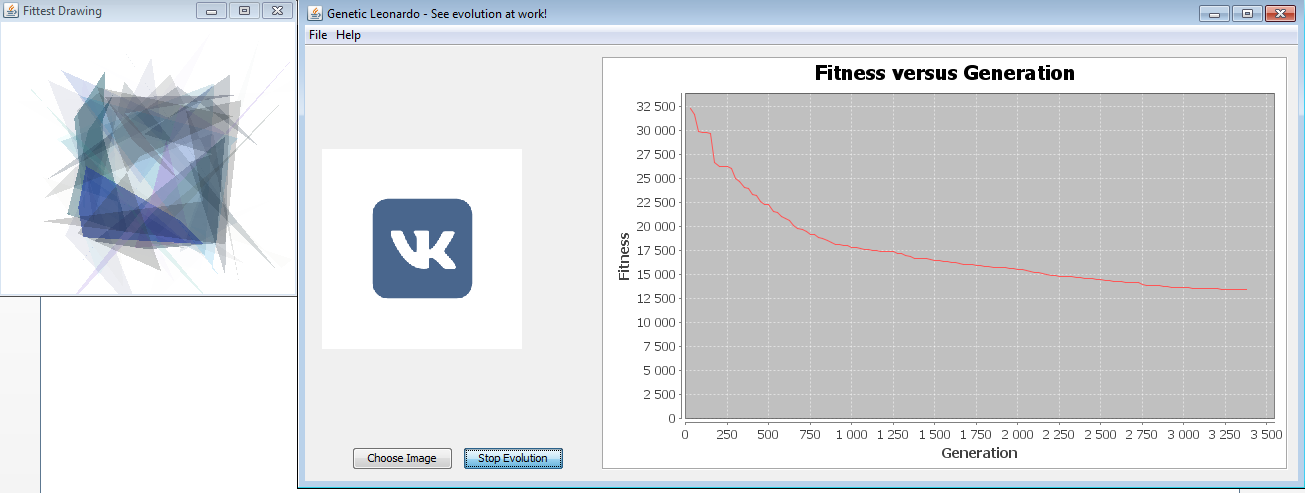


Рис. 4 Робота алгоритму після ~3400 популяцій

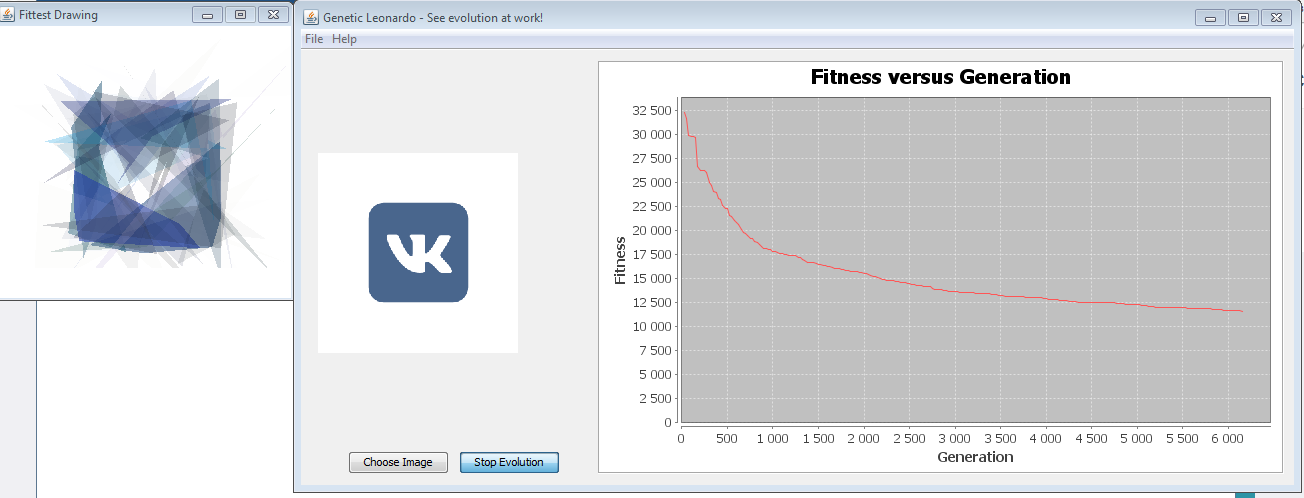
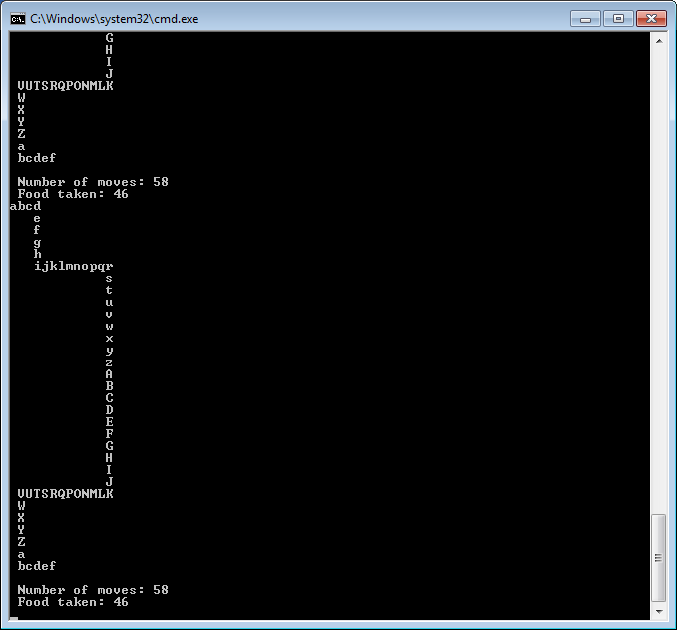


Рис. 4 Робота алгоритму після ~6000 популяцій

**Приклад 2. Gp anttrail**

**Демонструє, як рухатиметься мураха, вибираючи стільки їжі, скільки можливо**



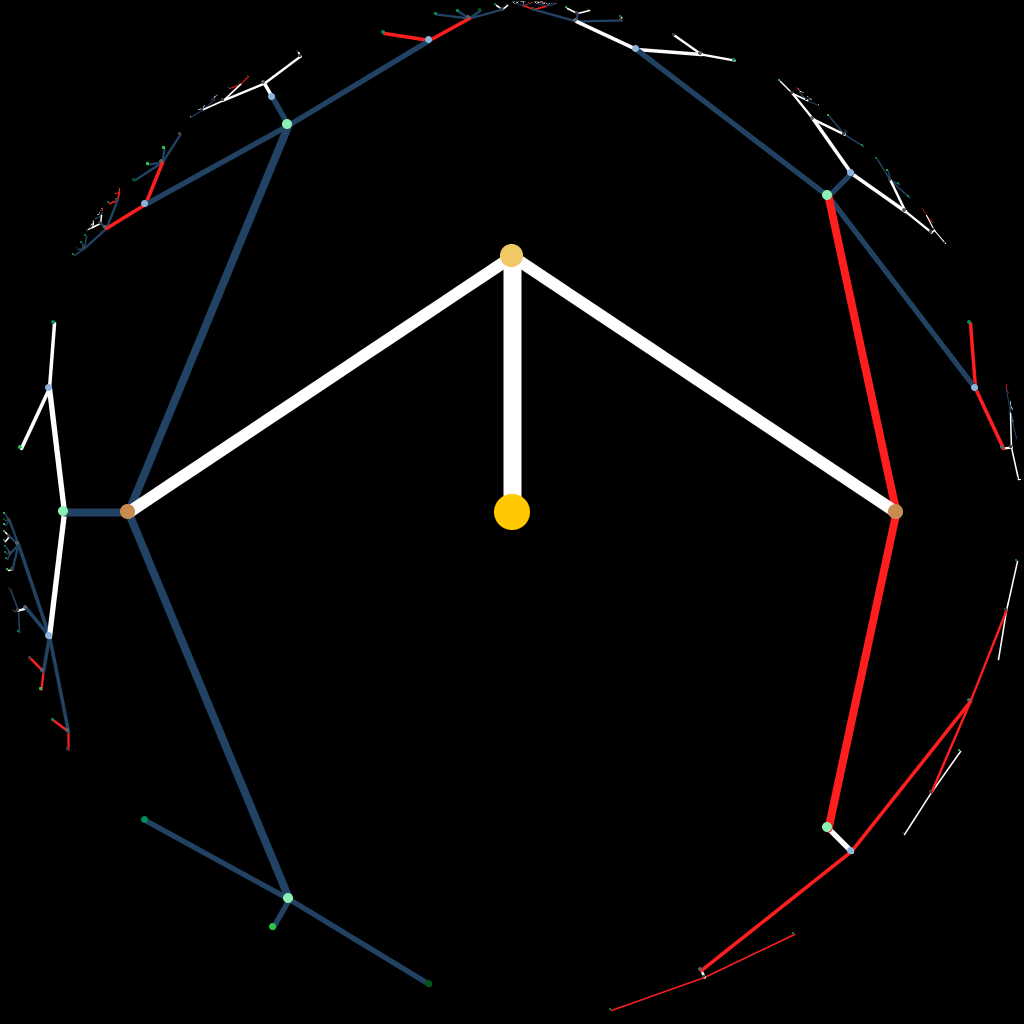


Рис.5 Результат виконання

**Приклад 3. MinimizationMakeChange**

**public class MinimizingMakeChange {**

**private final static String CVS\_REVISION = "$Revision: 1.27 $";**

**/\*\***

**\* The total number of times we'll let the population evolve.**

**\*/**

**private static final int MAX\_ALLOWED\_EVOLUTIONS = 50;**

**public static EvolutionMonitor m\_monitor;**

**public static void makeChangeForAmount(int a\_targetChangeAmount,**

**boolean a\_doMonitor)**

**throws Exception {**

**// Start with a DefaultConfiguration, which comes setup with the**

**// most common settings.**

**// -------------------------------------------------------------**

**Configuration conf = new DefaultConfiguration();**

**// Care that the fittest individual of the current population is**

**// always taken to the next generation.**

**// Consider: With that, the pop. size may exceed its original**

**// size by one sometimes!**

**// -------------------------------------------------------------**

**conf.setPreservFittestIndividual(true);**

**conf.setKeepPopulationSizeConstant(false);**

**// Set the fitness function we want to use, which is our**

**// MinimizingMakeChangeFitnessFunction. We construct it with**

**// the target amount of change passed in to this method.**

**// ---------------------------------------------------------**

**FitnessFunction myFunc =**

**new MinimizingMakeChangeFitnessFunction(a\_targetChangeAmount);**

**conf.setFitnessFunction(myFunc);**

**if (a\_doMonitor) {**

**// Turn on monitoring/auditing of evolution progress.**

**// --------------------------------------------------**

**m\_monitor = new EvolutionMonitor();**

**conf.setMonitor(m\_monitor);**

**}**

**// Now we need to tell the Configuration object how we want our**

**// Chromosomes to be setup. We do that by actually creating a**

**// sample Chromosome and then setting it on the Configuration**

**// object. As mentioned earlier, we want our Chromosomes to each**

**// have four genes, one for each of the coin types. We want the**

**// values (alleles) of those genes to be integers, which represent**

**// how many coins of that type we have. We therefore use the**

**// IntegerGene class to represent each of the genes. That class**

**// also lets us specify a lower and upper bound, which we set**

**// to sensible values for each coin type.**

**// --------------------------------------------------------------**

**Gene[] sampleGenes = new Gene[4];**

**sampleGenes[0] = new IntegerGene(conf, 0, 10); // Quarters**

**sampleGenes[1] = new IntegerGene(conf, 0, 2); // Dimes**

**sampleGenes[2] = new IntegerGene(conf, 0, 1); // Nickels**

**sampleGenes[3] = new IntegerGene(conf, 0, 5); // Pennies**

**IChromosome sampleChromosome = new Chromosome(conf, sampleGenes);**

**conf.setSampleChromosome(sampleChromosome);**

**// Finally, we need to tell the Configuration object how many**

**// Chromosomes we want in our population. The more Chromosomes,**

**// the larger number of potential solutions (which is good for**

**// finding the answer), but the longer it will take to evolve**

**// the population (which could be seen as bad).**

**// ------------------------------------------------------------**

**conf.setPopulationSize(20);**

**// Create random initial population of Chromosomes.**

**// Here we try to read in a previous run via XMLManager.readFile(..)**

**// for demonstration purpose only!**

**// -----------------------------------------------------------------**

**Genotype population;**

**try {**

**Document doc = XMLManager.readFile(new File("JGAPExample32.xml"));**

**population = XMLManager.getGenotypeFromDocument(conf, doc);**

**}**

**catch (UnsupportedRepresentationException uex) {**

**// JGAP codebase might have changed between two consecutive runs.**

**// --------------------------------------------------------------**

**population = Genotype.randomInitialGenotype(conf);**

**}**

**catch (FileNotFoundException fex) {**

**population = Genotype.randomInitialGenotype(conf);**

**}**

**// Now we initialize the population randomly, anyway (as an example only)!**

**// If you want to load previous results from file, remove the next line!**

**// -----------------------------------------------------------------------**

**population = Genotype.randomInitialGenotype(conf);**

**// Evolve the population. Since we don't know what the best answer**

**// is going to be, we just evolve the max number of times.**

**// ---------------------------------------------------------------**

**long startTime = System.currentTimeMillis();**

**for (int i = 0; i < MAX\_ALLOWED\_EVOLUTIONS; i++) {**

**if (!uniqueChromosomes(population.getPopulation())) {**

**throw new RuntimeException("Invalid state in generation "+i);**

**}**

**if(m\_monitor != null) {**

**population.evolve(m\_monitor);**

**}**

**else {**

**population.evolve();**

**}**

**}**

**long endTime = System.currentTimeMillis();**

**System.out.println("Total evolution time: " + ( endTime - startTime)**

**+ " ms");**

**// Save progress to file. A new run of this example will then be able to**

**// resume where it stopped before! --> this is completely optional.**

**// ---------------------------------------------------------------------**

**// Represent Genotype as tree with elements Chromomes and Genes.**

**// -------------------------------------------------------------**

**DataTreeBuilder builder = DataTreeBuilder.getInstance();**

**IDataCreators doc2 = builder.representGenotypeAsDocument(population);**

**// create XML document from generated tree**

**XMLDocumentBuilder docbuilder = new XMLDocumentBuilder();**

**Document xmlDoc = (Document) docbuilder.buildDocument(doc2);**

**XMLManager.writeFile(xmlDoc, new File("JGAPExample26.xml"));**

**// Display the best solution we found.**

**// -----------------------------------**

**IChromosome bestSolutionSoFar = population.getFittestChromosome();**

**double v1 = bestSolutionSoFar.getFitnessValue();**

**System.out.println("The best solution has a fitness value of " +**

**bestSolutionSoFar.getFitnessValue());**

**bestSolutionSoFar.setFitnessValueDirectly(-1);**

**System.out.println("It contains the following: ");**

**System.out.println("\t" +**

**MinimizingMakeChangeFitnessFunction.**

**getNumberOfCoinsAtGene(**

**bestSolutionSoFar, 0) + " quarters.");**

**System.out.println("\t" +**

**MinimizingMakeChangeFitnessFunction.**

**getNumberOfCoinsAtGene(**

**bestSolutionSoFar, 1) + " dimes.");**

**System.out.println("\t" +**

**MinimizingMakeChangeFitnessFunction.**

**getNumberOfCoinsAtGene(**

**bestSolutionSoFar, 2) + " nickels.");**

**System.out.println("\t" +**

**MinimizingMakeChangeFitnessFunction.**

**getNumberOfCoinsAtGene(**

**bestSolutionSoFar, 3) + " pennies.");**

**System.out.println("For a total of " +**

**MinimizingMakeChangeFitnessFunction.amountOfChange(**

**bestSolutionSoFar) + " cents in " +**

**MinimizingMakeChangeFitnessFunction.**

**getTotalNumberOfCoins(**

**bestSolutionSoFar) + " coins.");**

**}**

**/\*\***

**\* Main method. A single command-line argument is expected, which is the**

**\* amount of change to create (in other words, 75 would be equal to 75**

**\* cents).**

**\***

**\* @param args amount of change in cents to create**

**\* @throws Exception**

**\***

**\* @author Neil Rotstan**

**\* @author Klaus Meffert**

**\* @since 1.0**

**\*/**

**public static void main(String[] args)**

**throws Exception {**

**if (args.length < 1) {**

**System.out.println("Syntax: MinimizingMakeChange <amount>");**

**}**

**else {**

**int amount = 0;**

**try {**

**amount = Integer.parseInt(args[0]);**

**}**

**catch (NumberFormatException e) {**

**System.out.println(**

**"The <amount> argument must be a valid integer value");**

**System.exit(1);**

**}**

**if (amount < 1 ||**

**amount >= MinimizingMakeChangeFitnessFunction.MAX\_BOUND) {**

**System.out.println("The <amount> argument must be between 1 and "**

**+**

**(MinimizingMakeChangeFitnessFunction.MAX\_BOUND - 1)**

**+ ".");**

**}**

**else {**

**boolean doMonitor = false;**

**if (args.length > 1) {**

**String monitoring = args[1];**

**if(monitoring != null && monitoring.equals("MONITOR")) {**

**doMonitor = true;**

**}**

**}**

**makeChangeForAmount(amount, doMonitor);**

**}**

**}**

**}**

**/\*\***

**\* @param a\_pop the population to verify**

**\* @return true if all chromosomes in the populationa are unique**

**\***

**\* @author Klaus Meffert**

**\* @since 3.3.1**

**\*/**

**public static boolean uniqueChromosomes(Population a\_pop) {**

**// Check that all chromosomes are unique**

**for(int i=0;i<a\_pop.size()-1;i++) {**

**IChromosome c = a\_pop.getChromosome(i);**

**for(int j=i+1;j<a\_pop.size();j++) {**

**IChromosome c2 =a\_pop.getChromosome(j);**

**if (c == c2) {**

**return false;**

**}**

**}**

**}**

**return true;**

**}**

**}**

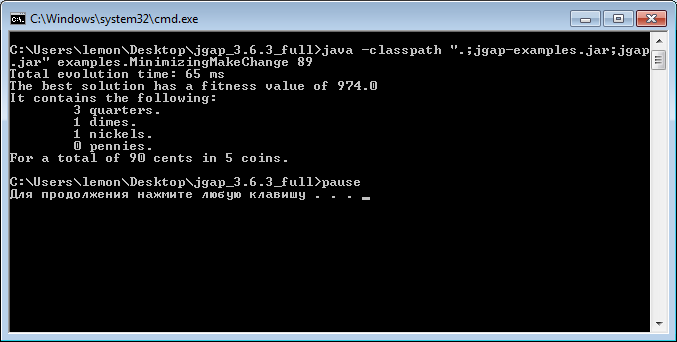


Рис.6 Результат виконання **MinimizationMakeChange**

**Висновок**

В результаті виконання розрахункової роботи озайомилася з фреймворком для еволюційних обчислень JGAP. Навела короткі теоретичні відомості, щодо структури та роботи. Навела приклади роботи даного фреймворку з результатами виконання.