УДК 681.3

АНАЛИЗ ЭФФЕКТИВНОСТИ ИСПОЛЬЗОВАНИЯ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ В ЗАДАЧАХ ПРИ ИСПОЛЬЗОВАНИИ РАЗЛИЧНЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

К.А. Генералов

ГОУ ВПО «Пензенский государственный университет», г. Пенза

Рецензент В.Е. Подольский

Ключевые слова и фразы: генетические алгоритмы; метрики Холстеда; язык программирования генетических алгоритмов; C++; Pascal; Lisp.

Аннотация: Рассмотрены проблемы использования генетических алгоритмов средствами различных языков программирования. Предлагается язык генетического программирования как наиболее эффективное средство использования генетических алгоритмов.

Введение

Впервые генетический алгоритм (**ГА**) был предложен в 1975 году Джоном Холландом (John Holland) в Мичиганском университете. Он получил название «репродуктивный план Холланда» и лег в основу практически всех вариантов генетических алгоритмов [5]. Генетические алгоритмы предназначены для решения задач оптимизации. Примерами таких задач могут служить обучение нейронной сети, задачи линейного программирования.

Основу генетических алгоритмов составляет метод случайного поиска. Принцип работы основан на идеях эволюции живой природы. Недостатками генетических алгоритмов является сложность использования при решении практических задач ввиду следующих обстоятельств:

– сложности данного механизма;

– вероятностного характера ГА, отсутствия гарантии сходимости процесса;

– полученного результата, который является приближенным.

На сегодняшний день отсутствуют лингвистические и инструментальные программные средства, позволяющие описывать задачи и процесс

Генералов К.А. – аспирант кафедры «Математическое обеспечение и применение ЭВМ» ПензГУ, г. Пенза.

их решения с помощью ГА. В работах [1, 2] в качестве наиболее универсального и гибкого инструмента использования генетических алгоритмов предложен язык программирования генетических алгоритмов. Он ориентирован на обработку объектов генетических алгоритмов.

Постановка задачи

В данной работе проводится анализ эффективности решения задачи средствами нескольких языков программирования. В качестве задачи необходимо найти глобальный экстремум следующей функции:

$$f(\bar{x}) = \sum_{i=1}^{15} (x^2 - \cos(x_i)) + 16 . \quad (1)$$

Функция достигает минимума $f(\bar{x}) = 1$ при $x_i = 0,\ i = \overline{1, ..., 15}$. Заданы следующие параметры генетического алгоритма:

– количество итераций алгоритма: 300;

– область поиска: $-1000 \le x_i \le 1000, i = \overline{1, ..., 15}$;

– количество хромосом в популяции: 80;

– стратегия выбора хромосом при скрещивании: случайным образом;

– селекция: метод «рулетки»;

– тип оператора скрещивания: «плоский» кроссовер;

– тип оператора мутации: случайная мутация;

– количество мутируемых генов: 1.

В качестве языков программирования для сравнительного анализа были выбраны языки C++, Lisp, Pascal.

Метод анализа программного кода

Для анализа кода программы возможно применение следующих характеристик [4]:

– количество строк кода (англ.: *Lines of Code* (**LOC**));

– метрические характеристики Холстеда;

– индекс разработки (англ.: *Maintainability Index* (**MI**));

– циклическая сложность (англ.: *Cyclomatic Complexity* (**CC**));

– расширенная циклическая сложность (англ.: *Extended Cyclomatic Complexity* (**ECC**)).

Для сравнительного анализа реализаций генетического алгоритма были выбраны метрические характеристики Холстеда, так как они наиболее полно описывают полученный код. Необходимо вычисление следующих характеристик [3]: длина программы; объем программы; уровень программы; уровень языка программирования; ожидаемое время реализации алгоритма; интеллектуальное содержание алгоритма; количество переданных ошибок; работа.

Для расчета указанных характеристик необходимы следующие основные данные [3]:

– η_1 – число простых (или отдельных) операторов, появляющихся в данной реализации;

– η_2 – число простых (или отдельных) операндов, появляющихся в данной реализации;

– N_1 – общее число всех операторов, появляющихся в данной реализации;

– N_2 – общее число всех операндов, появляющихся в данной реализации.

Таким образом, общими характеристиками программы для ЭВМ являются используемый словарь η и длина реализации N, причем:

$$\eta = \eta_1 + \eta_2 , \tag{2}$$

$$N = N_1 + N_2 . \tag{3}$$

Величины η и N из выражений (2) и (3) являются экспериментальными.

Ожидаемое значение длины программы $\overline{N}$ вычисляется следующим образом:

$$\overline{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2 . \tag{4}$$

Важной характеристикой реализации алгоритма является его размер. При переводе данного алгоритма с одного языка на другой его размер меняется. Аналогично алгоритмы, написанные на одном языке, имеют разные размеры. Изучение всех этих изменений количественными методами требует, чтобы размер был измеримой величиной.

Соответствующая метрическая характеристика размера любой реализации какого-либо алгоритма, называемая объемом *V*, может быть определена как

$$V = (N_1 + N_2) \log_2 (\eta_1 + \eta_2) . \tag{5}$$

Величина, характеризующая уровень программы, определяется по формуле

$$\overline{L} = \frac{2}{\eta_1} \frac{\eta_2}{N_2} . \tag{6}$$

Мера информационного содержания реализации алгоритма на некотором языке количественно описывает интеллектуальное содержание *I*, вычисляемое по формуле

$$I = \overline{L}\, V . \tag{7}$$

Интеллектуальное содержание есть величина постоянная для некоторого алгоритма, независимо от языка программирования, на котором он реализован.

Одной из важнейших величин является время реализации алгоритма на заданном языке программирования. Ожидаемое время реализации алгоритма вычисляется по формуле

$$\overline{T} = \frac{\eta_1 N_2 (\eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2) \log_2 \eta}{2\eta_2 S}, \quad (8)$$

где S – число Страуда, причем $5 \le S \le 20$. Число Страуда определяет скорость обработки информации человеческим мозгом.

Работа, проделанная в ходе реализации алгоритма, определяется по формуле:

$$E = \frac{V^2}{(2+\eta_2^*)\log_2(2+\eta_2^*)}, \quad (9)$$

где η_2^* представляет собой суммарное количество входных и выходных параметров алгоритма. Данная характеристика есть количество элементарных различий (или элементарных операций), которые производит человеческий мозг в ходе реализации программы.

Характеристикой языка программирования является уровень языка, определяемый как

$$\lambda = L(2+\eta_2^*)\log_2(2+\eta_2^*). \quad (10)$$

При реализации одного и того же алгоритма на языках программирования разного уровня меняются величины: длина программы, объем программы, время программирования.

Число ошибок программы определяется по формуле

$$B = \frac{V}{3000}. \quad (11)$$

Важно отметить, что речь идет о «переданных ошибках», то есть ошибках, остающихся в реализации программы после некоторой идентифицирующей фазы, такой как публикация или переход к следующему этапу разработки (передача модуля для сборки программного продукта).

При использовании вышепредставленных формул в разработанной программе для ЭВМ должны отсутствовать следующие «несовершенства» [4]:

– дополняющие друг друга операции (последовательное применение двух дополняющих друг друга операторов к одному и тому же операнду);

– неоднозначные операнды (имя операнда используется для обозначения разных объектов);

– синонимичные операнды (указание двух различных имен для одного и того же объекта);

– общие подвыражения (подвыражения, которые встречаются несколько раз);

– ненужное присваивание;

– использование выражений, не представленные в виде произведений множителей (то есть произведение представляется в виде суммы, при этом выполняются ненужные операции).

Результаты эксперимента

Реализация генетического алгоритма на языке генетического программирования представлена ниже:

```
cr:pcross;
mt:pmut;
sl:psel;
num_gen:int;
num_ch:int;
iter:int;

mt.MTYPE = REAL_MUT_RND;
cr.CTYPE = REAL_CROSS_FLAT;
sl.STYPE = SEL_RND;
sl.FIT = fitness;

ch:list(15) of real;
p:pop(80) ch;

float fitness(ch:list(15) of real)
{
 sum:real;
 res:real;
 sum=0;
 num_gen = 0;
 while (num_gen < 15)
 {
  sum = sum +pow(ch[num_gen],2)-0.25*cos(ch[num_gen]);
  num_gen = num_gen+1;
 }
 res = sum+16;
 return res;
}
 num_gen = 0;
 num_ch = 0;
 p.TARGET = MIN;
 while (num_ch < 15)
 {
  while (num_ gen < 80)
  {
   p[num_ch][num_gen] = rnd*2000 - 1000;
   num_gen = num_gen + 1;
  }
  num_ch = num_ch +1 ;
 }
 p.STEPS = 300;
 while (iter<300)
 {
  p.STEP = iter;
  crossover(cr,p);
  mutation(mt,p);
  sellection(sl,p);
  iter = iter + 1;
 }
print bestfit(p);
```

Для расчетов характеристик реализаций алгоритма экспериментально определены данные, представленные в табл. 1.

Таблица 1

Экспериментальные данные

Название характеристики	Программа на C++	Программа на Pascal	Программа на Lisp	Программа на языке генетического программирования
η_1	34	28	30	23
η_2	35	30	34	28
N_1	161	149	158	60
N_2	215	221	174	70

На основе данных табл. 1 были произведены расчеты характеристик каждой реализации генетического алгоритма. Число Страуда в расчетах бралось равным 18. Результаты расчетов представлены в табл. 2.

Таблица 2

Сводные данные о проделанном эксперименте

Название характеристики	Программа на C++	Программа на Pascal	Программа на Lisp	Программа на языке генетического программирования
Длина программы	352	282	320	239
Объем программы	2297	2167	1992	737
Уровень программы	0,01	0,011	0,013	0,035
Уровень языка программирования	0,23	0,233	0,313	0,835
Ожидаемое время реализации алгоритма, с	12490	9459	8193	2162
Интеллектуальное содержание алгоритма	22	21	26	25,6
Количество переданных ошибок	0,766	0,722	0,664	0,246
Работа	261700	255300	141400	19840

Выводы

Анализ полученных данных позволяет сделать следующие выводы:

– код, получаемый при разработке программ на языке программирования генетических алгоритмов обладает более низкой длиной и объемом, по сравнению с существующими языками программирования высокого уровня;

– уровень программы на новом языке более высокий, чем на существующих языках;

– уровень нового языка программирования выше, чем у существующих языков программирования высокого уровня;

– при использовании нового языка программирования получен существенный выигрыш в работе и по времени программирования;

– ожидаемое количество ошибок значительно ниже при решении задач с помощью языка программирования генетических алгоритмов.

Таким образом, на основании результатов анализа экспериментальных данных можно сделать вывод, что разработанный язык программирования генетических алгоритмов значительно превосходит существующие языки программирования высокого уровня при решении экстремальных задач с помощью разработанного языка.

Список литературы

1. Генералов, К.А. Язык генетического программирования / К.А. Генералов // Новые информационные технологии и системы : тр. 7 междунар. науч.-техн. конф., часть 2. – Пенза : Информационно-издательский центр ПГУ, 2006. – С. 98.

2. Генералов, К.А. Программная среда для разработки и распараллеливания генетических алгоритмов / К.А. Генералов // Технологии Microsoft в теории и практике программирования : материалы конф. – Изд-во Нижегород. гос. ун-та, 2007. – С. 31–34.

3. Холстед, М.Х. Начала науки о программах / М.Х. Холстед ; пер. с англ. В.М. Юфы. – М. : Финансы и статистика, 1981. –128 с.

4. Magnus Andersson Patrik Vestergren ,Object-Oriented Design Quality Metrics // Uppsala Master's Theses in Computer Science 276, Information Technology Computing Science Department Uppsala University, Sweden, 2004-06-07.

5. Herrera F., Lozano M., Verdegay J.L. Tackling real-coded genetic algorithms: operators and tools for the behaviour analysis // Artificial Intelligence Review. – 1998. – Vol. 12. – No. 4.– P. 265–319.

Analysis of Efficiency of Genetic Algorithms Application in Tasks with Different Programming Languages

K.A. Generalov

Penza State University, Penza

Key words and phrases: genetic algorithms, genetic programming language, Halstead metricks, C++,Pascal, Lisp.

Abstract: The paper deals with problems of genetic algorithms application by means of different programming languages. Genetic programming language as the most effective instrument of genetic algorithms application is offered.
