Т.С. Емельянова

ОБ ОДНОМ ГЕНЕТИЧЕСКОМ АЛГОРИТМЕ РЕШЕНИЯ ТРАНСПОРТНОЙ ЗАДАЧИ

Введение. Транспортные задачи (ТЗ) – это широкий круг задач исследования операций. Цель ТЗ найти такой план перевозок, при котором цена перевозок была бы минимальной, а также выполнялись ограничения на спрос и предложения [1-3]. Если цена перевозок пропорциональна количеству перевозимого груза, то такая транспортная задача называется линейной, в противном случае – нелинейной [3].

Для решения ТЗ, как для линейных, так и для нелинейных, разработаны генетические алгоритмы (ГА) [4]. В данной статье описан новый ГА для решения ТЗ. Алгоритм реализован в программе, написанной на языке С++, позволяющей решать линейные ТЗ. Эксперименты проводились в операционной системе Windows 2000 SP4 и были ограничены следующими характеристиками персонального компьютера: процессор: AMD Athlon XP1700+ (частота ядра 1466 МГц, шины 133 МГц); оперативная память: PC3200 256 МБ (частота шины 200 МГц). Данная архитектура позволяет выбрать требуемые параметры ГА, описанные ниже, и за приемлемое время решить ТЗ, порядок задачи ограничивается только приведенными выше характеристиками

Данный генетический алгоритм дает множество субоптимальных решений задачи, что не позволяют алгоритмы линейного программирования. Поэтому, его можно применять параллельно и, получив результаты, выбрать наилучший для данной задачи, в смысле целевой функции. Распараллеливание вычислений очень актуально т. к. можно получить почти не ограниченные вычислительные ресурсы.

1. Структура генетического алгоритма. Для решения ТЗ будем использовать следующий ГА:

1. Создание начальной популяции решений.
2. Определение целевой функции для каждого решения.
3. Пока не проведено необходимое число генераций или не закончилось время, заданное на реализацию алгоритма:
   а) выбор элементов для репродукции (оператор селекции);
   б) оператор транслокации;
   в) оператор инверсии;
4. Реализация новой генерации.

Опишем каждый из операторов данного ГА подробно.

Создание начальной популяции решений. Для построения начальной популяции решений воспользуемся методом «псевдо северо-западного» угла, как простым в вычислительном смысле, так и удобным для применения генетических алгоритмов. Метод «северо-западного» угла для построения опорного решения подробно описан в [2,4]. В отличие от метода «северо-западного» угла, построение решения будем начинать не с левой верхней ячейки, а с ячейки номер строки и столбца которой генерируется вероятностным способом. Псевдокод работы алгоритма приведен ниже.

Введем следующие обозначения X [0 … nSour-1][0 … nDest-1] – исходная матрица, где nSour – количество пунктов отправления, nDest – количество пунктов назначения.

Index_0[0 … 1] – массив, в котором начальные, случайно выбранные номер строки и номер столбца в матрице X. Index_0[0] – случайно выбранный номер строки. Index_0[1] – случайно выбранный номер столбца.

Index_Cur[0 … 1] – массив, в котором содержатся текущие номер строки и номер столбца для матрицы X. Index_Cur[0] – номер строки. Index_Cur[1] – номер столбца.

Index_Max[0 … 1] – массив, в котором содержаться количество строк и столбцов в матрице X . Index_Max[0] – количество строк в матрице. Index_Max[1] – количество столбцов в матрице.

z – вспомогательная переменная. delta[0…1] – вспомогательный массив.

```
1. begin
  Index_Max[0] ← nSour
  Index_Max[1] ← nDest
  Index_0[0] ← случайное число от 0 до nSour - 1
  Index_0[1] ← случайное число от 0 до nDest - 1
  Index_Cur[0] ← Index_0[0]
  Index_Cur[1] ← Index_0[1]

2. do
   if aSour[Index_Cur[0]] < aDest[Index_Cur[1]]
   then z = 0
  X[Index_Cur[0]][Index_Cur[1]] ← aSour[Index_Cur[0]]
  aSour[Index_Cur[0]] ← 0
  aDest[Index_Cur[1]] ← aDest[Index_Cur[1]] -
aSour[Index_Cur[0]]
  else z = 1
  X[Index_Cur[0]][Index_Cur[1]] ← aDest[Index_Cur[1]]
  aDest[Index_Cur[1]] ← 0
  aSour[Index_Cur[0]] ← aSour[Index_Cur[0]] -
aDest[Index_Cur[1]]
   Index_Cur[z] ← Index_Cur[z] + delta[z]
   if Index_Cur[z] = Index_Max[z]
   then delta[z] ← - delta[z]
  Index_Cur[z] ← Index_Cur[0]
until Index_Cur[z] ≠ -1
```

1. В данном блоке выбирается вероятностным образом элемент матрицы (номер строки и номер столбца) с которого будет начинаться заполнение матрицы.

2. В данном блоке последовательно заполняются элементы матрицы, начиная с заданного, до тех пор, пока не будет заполнена вся матрица

Приведем пример конструирования начального решения для транспортной матрицы приведенной ниже.

Пусть заданы следующие условия транспортной задачи. Запасы в пунктах отправления a_i ($i = 1 \ \ldots 3$): $a_1 = 40$, $a_2 = 20$, $a_3 = 15$. Заявки b_i в пунктах назначения ($j = 1, \ldots 4$): $b_1 = 10$, $b_2 = 5$, $b_3 = 30$, $b_4 = 30$. Задана матрица стоимостей c_{ij} ($i = 1, \ldots 3$, $j = 1, \ldots 4$) перевозки единицы товара от каждого пункта отправления a_i до каждого пункта назначения b_i. Матрица стоимостей перевозок дана ниже. Выбираем $i = 2$; $j = 2$. Получаем следующее решение:

$$\begin{Vmatrix} 6 & 5 & 9 & 10 \\ 12 & 3 & 6 & 12 \\ 3 & 8 & 11 & 7 \end{Vmatrix} \Rightarrow \begin{Vmatrix} 10 & 0 & 0 & 30 \\ 0 & 5 & 15 & 0 \\ 0 & 0 & 15 & 0 \end{Vmatrix}.$$

Данная матрица решений удовлетворяет ограничениям на спрос и предложения.

Оператор селекции. Для выбора двух матриц (родителей) из текущей популяции воспользуемся оператором турнирной селекции [5]. Для этого из популяции

решений выберем случайным образом четыре матрицы решений, две лучшие матрицы из четырех выбранных используем для операции транслокации.

Оператор транслокации. Оператор транслокации позволяет на основе скрещивания и инвертирования пары родителей получать потомков. Другими словами он представляет собой комбинацию операторов кроссинговера и инверсии [6].

Алгоритм работы приведен ниже. Где: X1[0…nSour-1][0 … nDest-1] – матрица первого родителя; X2[0…nSour-1][0 … nDest-1] – матрица второго родителя; C1[0…nSour-1][0 … nDest-1] – матрица потомка; nSum – сумма элементов матрицы соответствующая определенному столбцу.

```
1.begin
    C1 ← X1 + X2

2.for i ← 0 to nSour –1
 j ← 0
 do nSour[i] ← nSour[i] - C1[i][j]
  C1[i][j] ← 0
 while nSour[i] > 0
 C1[i][j] ← - nSour[i]
 nSour[i] ← 0
3. for j ← 0 to nDest –1
 nSum ← 0
 ▷ находи сумму элементов j-го столбца
 for i ← 0 to nSour
  nSum ← nSum + C1[i][j]
  ▷ выполняем ограничение на спрос
  if nSum > nDest[j]
   then nSum ← nSum - nDest[j]
   for i ← 0 to nSour
    nSum ← nSum - C1[i][j]
    C1[i][j] ← 0
    C1[i][j + 1] ← C1[i][j + 1] + C1[i][j]
    if nSum < 0
     then C1[i][j] ← -nSum
      C1[i][j + 1] ← C1[i][j + 1] + nSum
     break
  else nSum ← nSum - nDest[j]
   for i ← 0 to nSour
    for k ← j to nDest
     nSum ← nSum + C1[i][k]
     C1[i][k] ← 0
     C1[i][j] ← C1[i][j] + C1[i][k]
     if nSum > 0
      then C1[i][k] ← C1[i][k] + nSum
     C1[i][j] ← C1[i][j] – nSum
     break
end
```

1. В данном блоке получаем матрицу C1 – сумма матриц родителей.

2. Выполняем ограничения на предложения.

3. В данном блоке выполняется ограничение на спрос путем перемещения товара между строками матрицы внутри одного столбца.

Работу алгоритма поясним на примере: пусть даны две матрицы решений.

$$A_1 = \begin{Vmatrix} 10 & 0 & 0 & 30 \\ 0 & 5 & 15 & 0 \\ 0 & 0 & 15 & 0 \end{Vmatrix} \text{ и } A_2 = \begin{Vmatrix} 0 & 0 & 30 & 10 \\ 0 & 0 & 0 & 20 \\ 10 & 5 & 0 & 0 \end{Vmatrix}.$$

Сумма матриц равна: $A_1 + A_2 = \begin{Vmatrix} 10 & 0 & 30 & 40 \\ 0 & 5 & 15 & 20 \\ 10 & 5 & 15 & 0 \end{Vmatrix} \Rightarrow \begin{Vmatrix} 10 & 0 & 30 & 0 \\ 0 & 0 & 0 & 20 \\ 10 & 5 & 0 & 0 \end{Vmatrix}$.

В результате получаем матрицу: $\begin{Vmatrix} 10 & 0 & 20 & 10 \\ 0 & 0 & 0 & 20 \\ 0 & 5 & 10 & 0 \end{Vmatrix}$.

Для данной матрицы выполняются ограничения на спрос и предложение.

Оператор инверсии. Для транспортной матрицы оператор инверсии (ОИ) может осуществить различными способами. ОИ в данном случае выполняется следующим образом:

1. Выбираются случайным образом два числа i_1 и i_2 в диапазоне (*1* до *n-1*), где n – число пунктов отправления (количество столбцов в транспортной матрице).
2. Переставим столбцы местами.
3. Перемещая груз между столбцами, в пределах одной строки выполним ограничения.

Пусть $i_1 = 1$, а $i_2 = 4$.

$$\begin{Vmatrix} 10 & 0 & 20 & 10 \\ 0 & 0 & 0 & 20 \\ 0 & 5 & 10 & 0 \end{Vmatrix} \Rightarrow \begin{Vmatrix} 10 & 0 & 20 & 10 \\ 20 & 0 & 0 & 0 \\ 0 & 5 & 10 & 0 \end{Vmatrix} \Rightarrow \begin{Vmatrix} 0 & 0 & 20 & 20 \\ 10 & 0 & 0 & 10 \\ 0 & 5 & 10 & 0 \end{Vmatrix}.$$

2. Статистика. В результатах экспериментов представленных ниже см. рис. 2 целевые функции индивидов в популяции отсортированы он меньших к большим с использованием алгоритма из [7]. Следовательно, в верхней строчке будет наименьшая целевая функция для данной генерации, а в нижнем столбце – наибольшая. Отдельной строкой посчитана средняя целевая функция популяции для текущей генерации.

Для экспериментов взяты следующие параметры ГА: количество особей в популяции – 12, количество генераций – 10. Данные параметры обеспечивают необходимое многообразие популяции и достаточны для сходимости алгоритма для данного примера. Вероятность применения оператора транслокации возьмем равной 1, т.е. при выборе двух родителей всегда применяется оператор транслокации, вероятность оператора инверсии 0,05. Эксперименты проводились для транспортной матрицы размера 6x10. Матрица стоимостей имеет вид, в нижней строке записаны количества заявок, в последнем столбце количества запасов в пунктах отправления см. рис.1.

10	8	5	6	9	6	7	8	6	5	48
6	7	8	6	5	15	3	40	2	7	30
8	7	10	8	7	6	5	15	3	3	27
7	5	4	6	86	5	15	3	12	4	20
2	3	24	34	2	4	10	5	6	8	40
8	7	1	8	17	6	5	14	3	3	30
18	27	42	12	26	20	10	20	10	10	

Рис.1. Исходная транспортная матрица

На рис.2 приведены значения целевых функций индивидов в популяции для каждой генерации. Минимальная стоимость перевозок для данного эксперимента равна 1071-ой условной единице. Соответствующая данной стоимости транспортная матрица, описывающая перемещения указанного количества товара в условных единица из пунктов отправления в пункты назначения, представлена на рис.4. На рис.3 показаны два графика зависимости целевой функции от номера генерации. Нижний график L2 соответствует лучшему значению целевой функции в каждой генерации, верхний L1 – худшему.

Из результатов экспериментов (см. рис.2,3) видно, что на 5-ой генерации наблюдается как бы сходимость ГА однако, продолжая работу алгоритма мы находим лучшее решение (решение с меньшей целевой функцией). Это позволяет сделать вывод о возможности выхода данного алгоритма из локальных оптимумов.

	1	2	3	4	5	6	7	8	9	10
Fit_ 1	1118.0	1573.0	1310.0	1121.0	1121.0	1067.0	1079.0	1075.0	1067.0	1071.0
Fit_ 2	1118.0	1624.0	1310.0	1121.0	1121.0	1067.0	1079.0	1075.0	1067.0	1071.0
Fit_ 3	2017.0	1708.0	1323.0	1121.0	1121.0	1121.0	1079.0	1075.0	1067.0	1071.0
Fit_ 4	2300.0	1852.0	1323.0	1189.0	1121.0	1121.0	1087.0	1075.0	1067.0	1071.0
Fit_ 5	2339.0	1852.0	1370.0	1194.0	1121.0	1121.0	1087.0	1079.0	1067.0	1079.0
Fit_ 6	2712.0	1914.0	1370.0	1202.0	1121.0	1121.0	1087.0	1079.0	1067.0	1079.0
Fit_ 7	2712.0	1914.0	1430.0	1214.0	1121.0	1121.0	1087.0	1079.0	1073.0	1079.0
Fit_ 8	2883.0	1942.0	1464.0	1233.0	1123.0	1121.0	1121.0	1079.0	1079.0	1079.0
Fit_ 9	3053.0	2041.0	1464.0	1233.0	1123.0	1121.0	1121.0	1097.0	1079.0	1079.0
Fit_10	3053.0	2151.0	1482.0	1235.0	1125.0	1121.0	1121.0	1097.0	1079.0	1079.0
Fit_11	3149.0	2181.0	1506.0	1306.0	1126.0	1121.0	1121.0	1097.0	1079.0	1079.0
Fit_12	3169.0	2183.0	1515.0	1316.0	1131.0	1121.0	1121.0	1117.0	1079.0	1079.0
AverFit	2468.6	1911.3	1405.6	1207.1	1122.9	1112.0	1099.2	1085.3	1072.5	1076.3

Рис.2. Результаты эксперимента

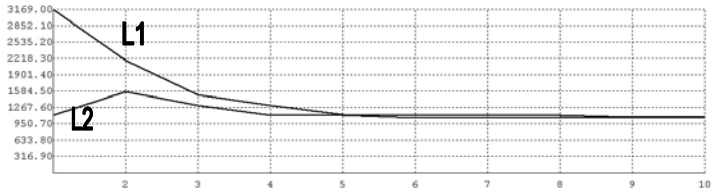

Рис.3. Зависимость целевой функции от номера генерации

2	16	0	0	0	20	10	0	0	0	48
0	0	0	4	26	0	0	0	0	0	30
8	11	0	8	0	0	0	0	0	0	27
8	0	12	0	0	0	0	0	0	0	20
0	0	0	0	0	0	0	20	10	10	40
0	0	30	0	0	0	0	0	0	0	30
18	27	42	12	26	20	10	20	10	10	

Рис.4. Решение транспортной задачи

Преимущество данного ГА в том, что, так как начальные условия ГА (построение начальной популяции) задаются вероятностным образом, можно независимо провести серию экспериментов и получить множество решений задачи, что не позволяет сделать ни один из традиционных алгоритмов линейного программирования.

3. Выводы. Исходя из результатов экспериментов, можно отметить, что данный генетический алгоритм позволяет решать линейные транспортные задачи. Средняя целевая функция популяции уменьшается на каждой генерации ГА, пока не сходится к определенному значению стоимости перевозок, которое будет являться наилучшим для данного эксперимента. Как видно из графика (см. рис.3),

данный генетический алгоритм позволяет выходить из локальных оптимумов. Спецификой ГА является то, что он дает множество решений задачи, что не позволяет сделать ни один другой алгоритм. Получив такое множество решений можно выбрать наилучшее (решение, дающее минимальное значение целевой функции). Изменяя параметры ГА алгоритма, для каждой конкретной задачи можно улучшить полученный результат. При этом для данного ГА не имеет значения порядок задачи (размер исходной транспортной матрицы) с таким же успехом он может решать задачи больших порядков, ограничиваясь лишь оперативной памятью вычислительной машины. К тому же, хотя данный ГА предназначен для решения линейной ТЗ, при небольшой модификации (в процедуре вычисления целевой функции) его можно применить и для решения нелинейной ТЗ, т.к. он не накладывает никаких ограничений на целевую функцию. Все вышесказанное позволяет сделать вывод, что данный ГА перспективен для решения ТЗ большой размерности, а также ТЗ с нелинейной целевой функцией. При этом путем распараллеливания вычислений, позволяет получать множество независимых друг от друга решений.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Вентцель Е.С.* Исследование операций: задачи, принципы, методология. – М.: Наука. 1988. – 208 с.
2. *Вентцель Е.С.* Исследование операций. М., «Советское радио», 1972, 552 с.
3. *Хэмди А. Таха.* Введение в исследование операций. 6-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 912 с.
4. Genetic Algorithms + data structures = evolution programs Zbignew Michalewicz – 3rd ed. 1996. – 387 p.
5. Генетические алгоритмы : Учебное пособие. Под ред. В. М. Курейчика. – Ростов-на-Дону: ООО «Ростиздат», 2004. – 400с
6. *Курейчик В.М.* Генетические алгоритмы и их применение. – Таганрог: Изд-во ТРТУ, 2002. – 242 с.
7. *Д. Кнут.* Искусство программирования для ЭВМ т.3. – М.: Мир. 1978.

М.А. Бакало

ВАРИАНТЫ ПРЕДСТАВЛЕНИЯ РАЗЛИЧНЫХ ТИПОВ ДАННЫХ, ИХ КОДИРОВАНИЕ И ДЕКОДИРОВАНИЕ В ВИДЕ ХРОМОСОМ*

Введение. Разработка генетического алгоритма включает три основных компонента: разработка структуры, принципов кодирования и декодирования хромосомы; разработка основных генетических операторов; разработка общей структуры генетического поиска [1-4].

Очевидно, что решение вопроса о способе представления различных типов данных в виде кода хромосомы во многом определяет содержание отдельных операторов и генетического алгоритма в целом. Кроме того, правильный выбор представления, способов кодирования и декодирования обуславливает не только пространственную и временную сложность алгоритма, но и его эффективность.

Авторами проведено исследование способов представления различной информации в виде кода хромосом, применяемых авторами различных генетических алгоритмов. Результаты проведенной работы, обобщены при построении следующей классификации:

* Работа выполнена при частичной финансовой поддержке программы развития научного потенциала высшей школы 2006-2008 годы (проекты РНП.2.1.2.2238, РНП 2.1.2.3193).