Применение генетических алгоритмов при моделировании транспортных задач (на примере задачи коммивояжера)

Р.Р. Муксимова
Кафедра прикладной математики
Санкт-Петербургский государственный университет гражданской авиации
Санкт-Петербург, Россия
e-mail: rose.r.mux@gmail.com

А.В. Кострицкая
Кафедра прикладной математики
Санкт-Петербургский государственный университет гражданской авиации
Санкт-Петербург, Россия
e-mail: k.anastasiav@yandex.ru

Аннотация[1]

При решении задачи коммивояжера с применением генетического алгоритма применялись различные способы отбора, кроссинговера и мутаций. Описаны особенности кодирования алгоритма для данной задачи. Приведены результаты для различных модификаций алгоритма на популяциях разного размера и времени жизни.

1. Введение

Генетические алгоритмы традиционно применяются для задач, которые решаются методом полного перебора возможных вариантов. Под данную классификацию (определение) попадает большинство транспортных оптимизационных задач. И задача коммивояжера в том числе. При реализации поиска решения с помощью генетического алгоритма особый интерес представляют способы мутаций и варианты кроссинговера.

2. Постановка задачи

Исторически задача формулируется следующим образом: коммивояжер (торговый агент) должен выйти из заданного пункта, посетить заданное количество городов, побывав в каждом только один раз и вернуться в исходный пункт. Задача коммивояжера является классической комбинаторной задачей оптимизации и может быть решена методом перебора всех возможных путей. Если количество городов взять равным n, то количество маршрутов будет равно

$$M = (n-1)!$$

и время решения будет пропорционально

$$T = \left(n + n^2\right) \cdot (n-1)! = (n+1)!,$$

где в первом множителе первое слагаемое n это вычисление длины каждого маршрута, а второе слагаемое n^2 это поиск маршрута минимальной длины среди всех вариантов. Для количества пунктов, равных 10, количество маршрутов будет равно $M = 9! = 362880$. Поиск решения этим способом потребует времени, которое займет время жизни нескольких поколений исследователей.

Исходными данными будет количество пунктов назначения V и стоимость путей между ними (матрица весов) $matr\_v[i, j]$, где $i, j = 0,\ldots,V-1$. Для решения задачи с применением генетического алгоритма необходимо задать размер популяции S и число поколений P (время жизни популяции). Популяцией в нашей задаче будет набор возможных маршрутов, а числом поколений – количество итераций алгоритма, в результате которых популяция обновляется за счет появления более приспособленного к жизни потомства.

Хромосома составляется из полного набора пунктов назначения без повторений. В нашем случае хромосома состоит из последовательности V генов, каждый из которых кодирует определенный пункт.

3. Реализация алгоритма

Для создания исходной популяции берется начальная хромосома, кодирующая маршрут по возрастанию генов (городов), и с помощью рандомных перестановок получаем первую родительскую популяцию S_0. Приспособленность каждой особи находим с помощью фитнес-функции путем последовательного сложения длин всех путей маршрута [1].

$$Fit\_function = \sum_{k,l} matr\_v[k,l], \qquad \textbf{(1)}$$

где l и k –это коды соседних генов в хромосоме.

Труды **четвертой международной конференции "Информационные технологии интеллектуальной поддержки принятия решений", 17 - 19 мая, Уфа, Россия, 2016**

Поскольку в нашей задаче хромосомы представляют замкнутую цепь, то первый и последний ген в хромосоме считаем также соседними.

На этом создание родительской популяции завершено, и можно приступить к размножению.

Для скрещивания сначала отсортировали популяцию по фитнес-функции и далее скрещивали соседей: сильные особи с сильными, а слабые со слабыми. Данный способ быстро приводит к вырождению популяции и появляется необходимость в серьезных масштабных мутациях, чтобы восстановить разнообразие генофонда. Поэтому был реализован еще один вариант скрещивания, когда особи для скрещивания выбираются случайным способом.

Точка разреза для кроссинговера в первых реализациях располагалась по центру хромосомы, но такая позиция приводит к однообразию потомства уже в 3-4 поколении. Далее точка выбиралась случайно между вторым и предпоследним геном[2].

Особенность кодирования алгоритма для задачи коммивояжера заключается в том, что двоичная кодировка генов и хромосом сильно усложняет обработку и потомства, и мутировавших особей, и этим увеличивает длительность поиска решения, т.к. в маршруте коммивояжера (хромосоме) города (то есть гены) не должны повторяться.

Поэтому приходится все операции над популяцией проводить над десятичными числами.

В нашем случае во время кроссинговера один родитель делился на две части, которые переписывались в потомков в неизменном виде, а второй дополнял каждого потомка до полноценной особи.

В новую популяцию отбирались только лучшие из родителей и потомков, при этом родитель имеет приоритет в случае равных фитнес-функций. Данное условие связано с необходимостью сохранить уже найденных лидеров.

Далее с целью улучшения генетического материала вводили определенный процент мутаций в новой популяции. Мутация производились разными способами. Например, для инверсии во время мутации используется обмен значениями для двух случайных генов внутри исходной хромосомы. Таким образом «инвертируются» значения двух генов. При этом вычисляется значение фитнес-функции для каждой мутации. После чего новая популяция снова отсортировывалась с уже мутировавшими хромосомами.

Для исследования влияния на результат сначала применяли мутацию на потомстве, а в следующей версии – на уже отсортированной новой популяции.

В другом варианте мутация проводилась при помощи случайной перестановки генов в одной из особей с плохой фитнес-функцией. Данная мутация позволила сократить численность популяции и количество поколений, необходимых для определения особей-победителей. Наблюдая положительную динамику в работе алгоритма, решили подвергнуть мутации не одну, а две слабые особи.

3. Результаты вычислительного эксперимента

Тестировалась программа на примере для 6 городов из учебника по теории графов [3].

Исходные данные представлены на рисунке 1. На первом шаге задается количество пунктов назначения, размер популяции и количество поколений, затем вводится матрица стоимости путей между городами.

```
Number of cities: 6

Population size: 12

Number of generations: 4


Weight matrix:
0   3   93  13  33  9
4   0   77  42  21  16
45  17  0   36  16  28
39  90  80  0   56  7
28  46  88  33  0   25
3   88  18  46  92  0
```

Рис. 1. Скрин ввода исходных данных

Далее создается исходная популяция (рис.2). Каждая строка это хромосома, последний элемент в сроке – значение фитнес-функции для данной особи. Исходная популяция сразу сортируется по возрастанию (ухудшению) значений фитнес-функции.

```
Initial population:
0   3   5   2   1   4   104
4   5   2   3   0   1   142
1   2   5   0   3   4   223
0   1   4   2   5   3   225
0   2   4   1   5   3   256
0   5   1   2   4   3   262
0   3   2   5   4   1   263
1   2   5   0   4   3   264
1   3   5   4   2   0   277
1   4   2   0   5   3   299
2   0   5   3   1   4   299
0   2   3   4   5   1   302
```

Рис. 2. Формирование исходной популяции

На рисунке 3 представлены результаты кроссинговера первой популяции со случайным выбором родителя. Видно, что по значению фитнес-функции дети в целом проигрывают родителям. Но самые удачные дети все же останутся в популяции и займут место худших из родителей. На рисунке 4 представлен результат естественного отбора (по значению фитнес-функции) из всех родителей и детей на данном этапе в результирующую популяцию с неизменившимся количеством особей *S*.

После отбора лучших из лучших в новую популяцию нужно с определенной долей вероятности произвести мутацию.

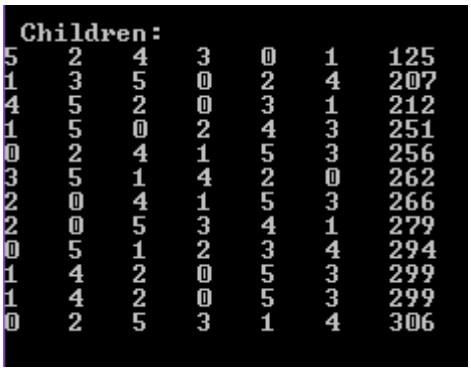

Рис. 3. Результаты скрещивания

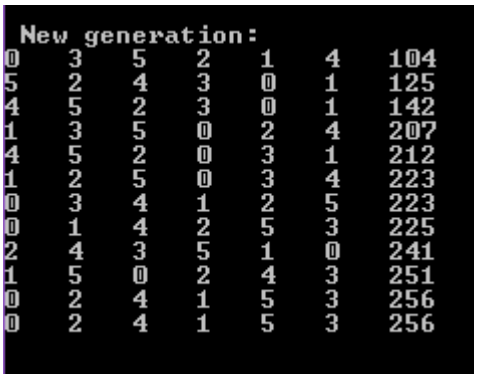

Рис. 4. Новая популяция

Необходимость мутации связана с тем, что алгоритм может уйти в одно из решений и упустить, что есть еще варианты. На одном из интернет-форумов в обсуждении генетического алгоритма прозвучала мысль, что грамотная выбранная мутация вполне может заменить кроссинговер. Учитывая, что генетический алгоритм вообще индивидуален для каждого типа задач, такое мнение вполне обосновано.

Оптимальная для данной задачи мутация подбиралась экспериментально.

Применялась мутация с «инверсией» двух случайных генов сначала в худшей, потом в случайно выбранной особи. Эксперимент показал, что выбор случайной или худшей особи не сильно повлиял на скорость поиска решения. Процесс мутации для случайной особи приведен на рисунках 5 и 6. Мутировавшая хромосома и результат мутации выделены цветом.

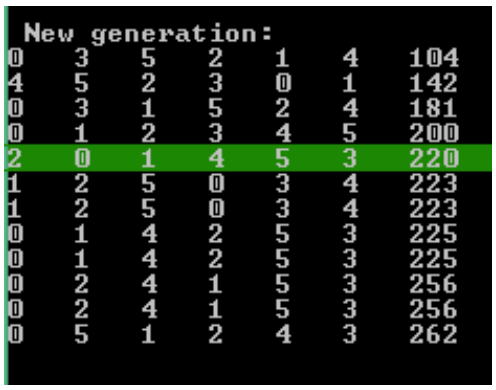

Рис. 5. Популяция перед мутацией «инверсией»

Рис. 6. Популяция после мутации «инверсией»

Также производилась мутация сначала слабой, а потом случайной особи с помощью случайной перестановки генов. Выбор особи не повлиял на скорость поисков.

На рисунках 7 и 8 представлен процесс мутации одной случайной особи с помощью случайной перестановки генов. Мутировавшая особь и результат также выделены цветом.

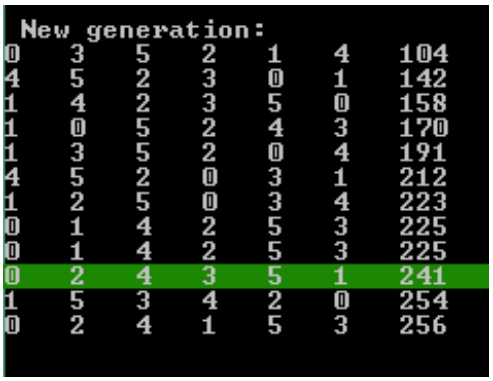

Рис. 7. Популяция перед мутацией с помощью перестановок

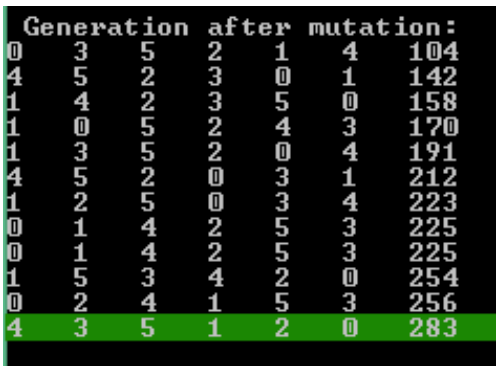

Рис. 8. Популяция после мутации с помощью перестановок

Как уже упоминалось ранее, при мутации перестановками наблюдалось сокращение количества поколений, которые приводили к нахождению всех оптимальных маршрутов, поэтому увеличили процент мутаций и мутировали уже 2 особи. При этом также не имело значения, выбирали мы худших или случайных особей. На рисунках 9 и 10 приведен пример мутации двух случайных особей с помощью перестановок.

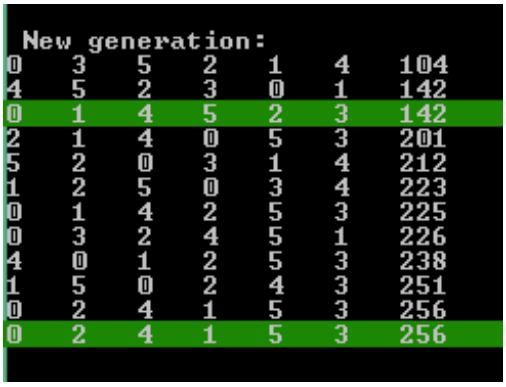

Рис. 9. Популяция с увеличенным процентом мутаций с помощью перестановок

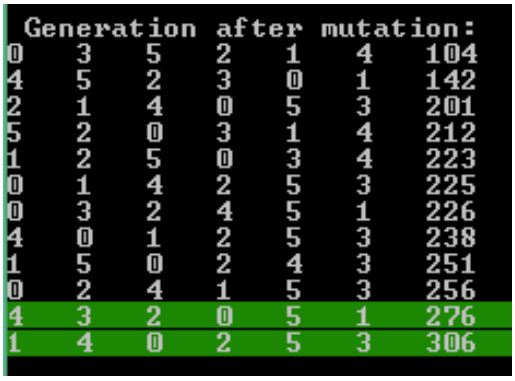

Рис. 10. Популяция с увеличенным процентом мутаций после мутации с помощью перестановок

Видно, что мутации зачастую производят довольно плохих особей, при этом мутация со случайной «инверсией» не самым лучшим образом влияла на работу алгоритма. А мутация со случайной перестановкой, хотя и производила часто особи с плохой фитнес-функцией, в отличие от мутации с «инверсией», не позволяла алгоритму свалиться в одно из решений, игнорируя остальные.

Поэтому для данной конкретной задачи применялась усиленная мутация со случайной перестановкой.

На рисунке 11 представлен результат мутации на текущей популяции, т.е. та популяция, которая станет родительской в следующем поколении.

Рис. 11. Новая популяция после мутации

Как видно из рисунка 12 уже на 4 итерации (число итераций было задано изначально) алгоритм находит самых приспособленных особей и они занимают 8 позиций из 12 в популяции.

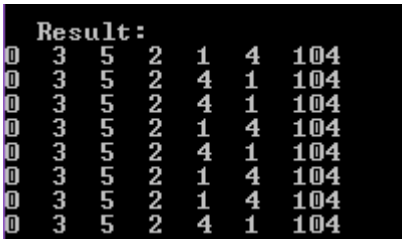

Рис. 12. Результат поиска самой приспособленной особи

Очевидно, что раз оба лидера найдены, продолжать итерации не имеет смысла.

В результате мы получили два оптимальных маршрута:

$$0\ \ 3\ \ 5\ \ 2\ \ 1\ \ 4,$$

$$0\ \ 3\ \ 5\ \ 2\ \ 4\ \ 1.$$

При этом длина (вес) оптимального маршрута находится по формуле (1)

$$Fit\_function(1) = matr\_v[0,3] + matr\_v[3,5] +$$

$$+ matr\_v[5,2] + matr\_v[2,1] + matr\_v[1,4] +$$

$$+ matr\_v[4,0] = 104.$$

И, соответственно, для второго маршрута

$$Fit\_function(2) = matr\_v[0,3] + matr\_v[3,5] +$$

$$+ matr\_v[5,2] + matr\_v[2,4] + matr\_v[4,1] +$$

$$+ matr\_v[1,0] = 104.$$

Интересен тот факт, что в данном примере должно получиться два варианта оптимального маршрута с одинаковой фитнес-функцией. При этом первый маршрут появлялся уже в первых поколениях, а второй гораздо позднее. При использовании не оптимальных для данной задачи способов мутации и отбора, если к финишу приходили одновременно 10 особей из популяции, то в лучшем случае 1-2 из них кодировали второй маршрут, остальные кодировали первый.

Как видно на рисунке 12, если хорошо подобрать мутацию, селекцию и кроссинговер, то вероятность того, что на финише получится одинаковое количество оптимальных маршрутов каждого типа гораздо выше, что говорит о лучшем отслеживании экстремумов.

4. Заключение

Таким образом, неудачная мутация и неверно проведенный кроссинговер и отбор приводят к вырождению популяции. При этом алгоритм либо вообще уходит от правильного решения, либо, в лучшем случае, сваливается в один из локальных экстремумов, теряя при этом все остальные оптимальные маршруты.

Напротив, если использовать различные мутации и скрещивания и наблюдать при этом за положительной и отрицательной динамикой поиска результатов, можно подобрать те способы реализации алгоритма, которые быстро найдут все оптимальные решения. При этом можно значительно сократить численность популяции и количество поколений. Что в свою очередь уменьшит вычислительную сложность (время выполнения) алгоритма.

Список используемых источников

1. Гладков Л.А., Курейчик В.В., Курейчик В.М. “Генетические алгоритмы”. ФИЗМАТЛИТ, Москва, 2006.
2. Рутковская Д., Пилиньский М., Рутковский Л. “Нейронные сети, генетические алгоритмы и нечеткие системы”. Горячая линия - Телеком, Москва, 2006.
3. Домнин Л.Н. “Элементы теории графов”. ПГУ, Пенза, 2007