Решение задачи коммивояжера с помощью генетических алгоритмов

Гуляев Алексей
Департамент компьютерных наук
Институт Транспорта и Связи
Ломоносова 1-1, Рига, Латвия
Email: alexgul@fis.lv
22 мая 2001 г.

Аннотация

Данная работа посвящена применению генетических алгоритмов для решения комбинаторной задачи нахождения наикратчайшего пути (коммивояжера). Эта задача НП-полная (задача с нелинейной полиномиальной оценкой числа итераций). ГА используется для нахождения около-оптимального пути за линейное время. В любой момент времени коммивояжер может быть только в городе. Города могут посещаться только единожды.

Ключевые слова: Генетические алгоритмы; Коммивояжер; TSP.

Содержание:

1. Введение в генетические алгоритмы

Генетический алгоритм - это простая модель эволюции в природе, реализованная в виде компьютерной программы. В нем используются как аналог механизма генетического наследования, так и аналог естественного отбора. При этом сохраняется биологическая терминология в упрощенном виде.

Хромосома	Вектор (последовательность) генов.
Фенотип⇔Генотип	Набор хромосом = вариант решения задачи.
Кроссовер	Операция, при которой две хромосомы обмениваются своими частями.
Мутация	Случайное изменение одной или нескольких позиций в хромосоме.

Табл.1. Моделирование генетического наследования

Чтобы смоделировать эволюционный процесс, сгенерируем вначале случайную популяцию - несколько индивидуумов со случайным набором хромосом (числовых векторов). Генетический алгоритм имитирует эволюцию этой популяции как циклический процесс скрещивания индивидуумов и смены поколений.

Жизненный цикл популяции - это несколько случайных скрещиваний (посредством кроссовера) и мутаций, в результате которых к популяции добавляется какое-то количество новых индивидуумов. Отбор в генетическом алгоритме - это процесс формирования новой популяции из старой, после чего старая популяция погибает. После отбора к новой популяции опять применяются операции кроссовера и мутации, затем опять происходит отбор, и так далее.

Приспособленность индивидуума	Значение целевой функции на этом индивидууме.
Выживание наиболее приспособленных	Популяция следующего поколения формируется в соответствии с целевой функцией. Чем приспособленнее индивидуум, тем больше вероятность его участия в кроссовере, т.е. размножении

Табл.2. Отображение естественного отбора в природе к генетическому алгоритму

Таким образом, модель отбора определяет, каким образом следует строить популяцию следующего поколения. Как правило, вероятность участия индивидуума в скрещивании берется пропорциональной его приспособленности. Часто используется так называемая *стратегия элитизма*, при которой несколько лучших индивидуумов переходят в следующее поколение без изменений, не участвуя в кроссовере и отборе. В любом случае каждое следующее поколение будет в среднем лучше предыдущего. Когда приспособленность индивидуумов перестает заметно увеличиваться, процесс останавливают и в качестве решения задачи оптимизации берут наилучшего из найденных индивидуумов**.**

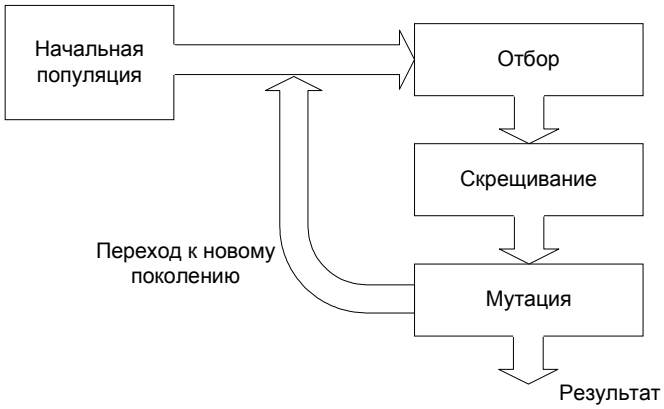

Рис.1. Блок-схема простого ГА

2. Введение в проблему коммивояжера

Суть задачи состоит в том, чтобы найти кратчайший замкнутый путь обхода нескольких городов, заданных своими координатами. Оказывается, что уже для *n* городов поиск оптимального пути представляет собой сложную задачу, побудившую развитие различных новых методов (в том числе нейросетей и генетических алгоритмов).

Каждый вариант решения (для *n* городов) - это числовая строка, где на j-ом месте стоит номер j-ого по порядку обхода города. Таким образом, в этой задаче *n* параметров, причем не все комбинации значений допустимы.

Эта задача НП-полная (задача с нелинейной полиномиальной оценкой числа итераций) и мультимодальная. ГА используется для нахождения около-оптимального пути за линейное время. В любой момент времени коммивояжер может быть только в городе. Города могут посещаться только единожды.

Значение функции приспособленности – расстояние, пройденное коммивояжером за весь маршрут. Оно должно быть минимальным.

2.1. Решение проблемы с помощью стандартного ГА

Для того чтобы иметь возможность использовать стандартный ГА необходимо решить следующие проблемы:

1. Поиск двоичного представления маршрутов, которые могут быть легко транслированы в хромосомы.
2. Определении функции приспособленности.

Решением данной проблемы может служить бинарная матрица V, где $V_{ct} = 1$ если город c посещен в момент времени t, в противном случае 0. Она гарантирует, что только одна “1” находится в каждой строке и столбце.

Например, пути ABEDC и CEDBA могут быть представлены в виде следующих матриц перестановок (строки-города, столбцы-время):

V_1

	1	2	3	4	5
A	1	0	0	0	0
B	0	1	0	0	0
C	0	0	0	0	1
D	0	0	0	1	0
E	0	0	1	0	0

Табл.3. Таблица пути ABEDC

V_2

	1	2	3	4	5
A	0	0	0	0	1
B	0	0	0	1	0
C	1	0	0	0	0
D	0	0	1	0	0
E	0	1	0	0	0

Табл.4. Таблица пути CEDBA

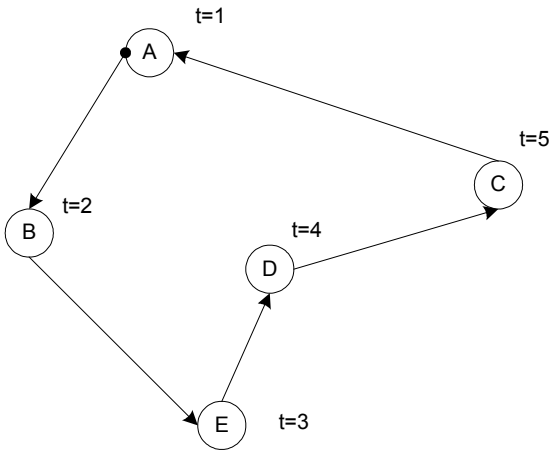

Рис.2. Путь ABEDC

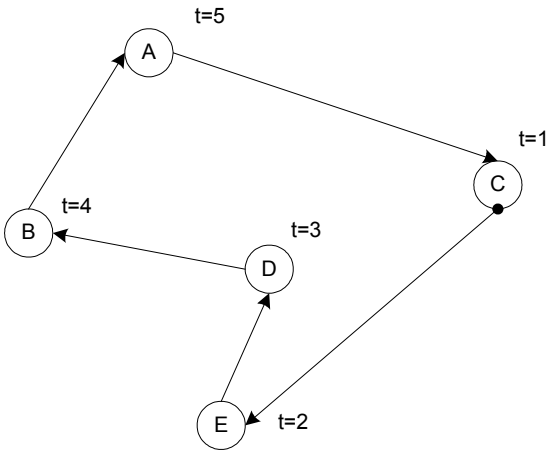

Рис.3. Путь CEDBA

Матрицы могут быть представлены в виде следующих хромосом:

$$V_1 = 1000001000000010001000100$$
$$V_2 = 0000100010100000010001000$$

Необходимо отметить, что города в одинаковой последовательности, но с разными стартовыми точками или с противоположным направлением кодируются разными матрицами, т.е. разными хромосомами. Например:

$V_3(BEDCA) = 0000110000000100010001000$, $V_1(ABEDC \equiv BEDCA)$.

ГА может создать некоторые хромосомы, не имеющие решения. Это может произойти при создании начальной популяции и при действии операторов ГА. Например, при одноточечном кроссовере может возникнуть следующая ситуация:

$$V_1 = 100000100000 \quad 0010001000100$$
$$V_2 = 000010001010 \quad 0000010001000$$

$$\Downarrow$$

$$V_4 = 100000100000 \quad 0000010001000$$

Рис.4. Пример скрещивания в результатекоторого, получилосьневерное решение

V_4					
	1	**2**	**3**	**4**	**5**
A	1	0	0	0	0
B	0	1	0	0	0
C	0	0	0	0	0
D	0	0	1	0	0
E	0	1	0	0	0

Табл.5. Таблица неверного решения

Очевидно, что такое решение неправильно, т.к. город C ни разу не посещен. Также неправильное решение появится в результате скрещивания V_1 и V_3.

Для избежания таких случаев необходимо использовать альтернативное представление хромосом.

2.2. Символьное представление

До этого момента использовалась схема представления, в которой каждый параметр решения (ген) ассоциирован с определенным набором битов хромосомы (локус), и значения этих битов использовались для оценивания значения функции приспособленности.

В природе функция гена сильно независима от позиции в ДНК, т.е. она будет продолжать создавать одинаковые аминокислоты вне зависимости от их позиций. Дополнительно, в природе существуют *операторы пересортировки*, такие как *инверсия*, которые решают задачу упорядочивания генов для заданного класса индивидов.

Для симуляции этой способности в ГА необходимо провести различия между локусами и аллелями, т.е. необходимо найти путь для представления значений параметров вне зависимости от их позиции в хромосомах. Это может быть сделано, например, с помощью добавления функционально-специфичной метки к каждому гену в хромосоме:

f1	f2	f3
ген	ген	ген

Поэтому пересортировка генов не изменит фенотип:

f1	f2	f3
ген	ген	ген

f2	f3	f1
ген	ген	ген

После того как этот тип представления хромосом определен, в ГА определяются *операторы пересортировки*, которые изменяют порядок генов в хромосоме.

Наиболее часто используется оператор *инверсии*. Он выбирает случайным образом 2 точки в хромосоме и переставляет гены (с ассоциированными метками) между этими точками.

Метки – строки символов. Эта форма представления называется *перестановочным кодированием* и лучше всего подходит для комбинаторных оптимизационных задач, таких как задача коммивояжера. В данном случае решение выглядит в виде массива символов или целых чисел, например ABEDC или 12543, каждый элемент которого представляет город, а последовательность – последовательность объезда этих городов.

Однако поскольку операторы скрещивания (см. рис.5) и мутации дают неправильные решения, то необходимо использовать *специализированные операторы*.

$$V_1 = 12 \;\; 543$$
$$V_2 = 35 \;\; 421$$
$$\Downarrow$$
$$V_3 = 12 \;\; 421$$

Рис.5. Неверная операция кроссовера

2.3. Символьные операторы ГА

Все символьные операторы , описанные в работе [2], являются перестановочно-относительными.

Все рассмотренные ниже кроссоверы могут создавать потомство, не имеющее решения (невалидные решения). Некоторые это делают реже других. Невалидные решения могут быть полезны для создания и внесения разнообразия в популяцию, однако этим они могут замедлить или даже предотвратить сходимость ГА. Одним из решений этой проблемы может стать идея *восстановления* (возможно, части) невалидных решений.

2.3.1. Частично отображаемый кроссовер

Частично отображаемый кроссовер берет сначала часть пути одного родителя и сохраняет последовательность и позиции как можно большего числа городов другого родителя. Точка кроссовера выбирается случайно (рис.6).

$$V_1 = 12 \;\; 543$$
$$V_2 = 35 \;\; 421$$

Рис.6. Точка сечения частично отображаемого кроссовера

Затем производится отображение замен первых частей хромосом $\{1 \Leftrightarrow 3, 2 \Leftrightarrow 5\}$, которое применяется поточечно к родителям для получения потомства. Проверяется каждый элемент первого родителя: если имеется для него замена, то она производится и затем копируется ($\Updownarrow$) в первого потомка (см. рис.10), в противном случае он просто копируется ($\Downarrow$) (см. рис.7). Подобная процедура может быть использована и для второго потомка.

$$V_1 = 1\,2\,5\,4\,3$$
$$\Updownarrow\Updownarrow\Updownarrow\Downarrow\Updownarrow$$
$$V_6 = 3\,5\,2\,4\,1$$

Рис.7. Частично отображаемый кроссовер

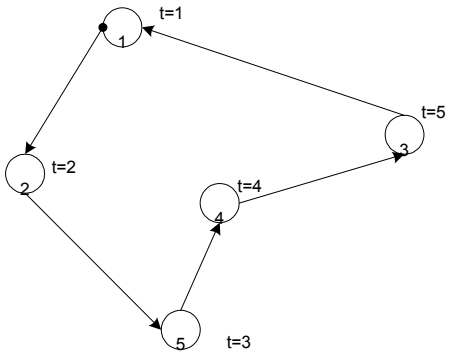

Рис.8. Родитель V_1

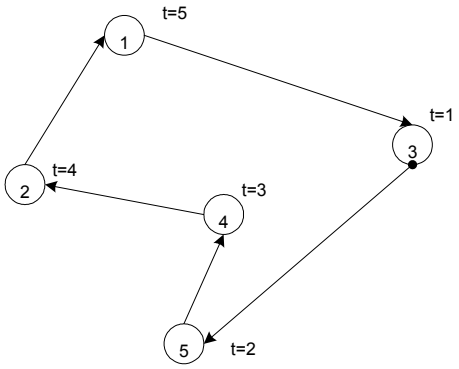

Рис.9. Родитель V_2

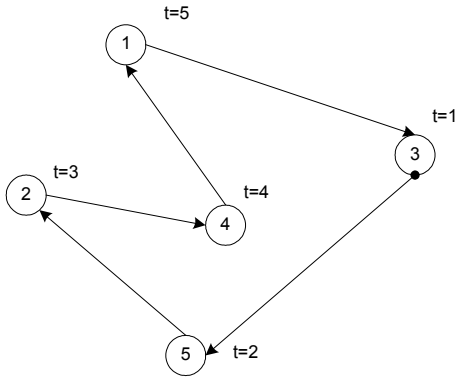

Рис.10. Потомок V_6

2.3.2. Упорядоченный кроссовер

Упорядоченный кроссовер берет часть пути одного родителя и сохраняет родственный порядок городов из другого родителя. Первые две точки кроссовера выбираются случайно (рис. 11). Каждый элемент центральной секции первого родителя копируется в потомка (рис.12).

$$V_1 = 12 \;\; 5 \;\; 43$$
$$V_2 = 35 \;\; 4 \;\; 21$$

Рис.11. Точки сечения в упорядоченном кроссовере

$$V_4 = 1\,\Updownarrow\Updownarrow\,4\,3$$
$$V_5 = 2\,5$$

Рис.12. Копирование точек в нового потомка

Затем элементы второго родителя собираются в список (рис.13), начиная со второй точки кроссовера.

Наконец, города, уже представленные в потомках, удаляются (рис.14), и оставшиеся элементы копируются вместо пустых пробелов потомка, начиная со второй точки сечения кроссовера (рис. 15).

[2 1 3 5 4]

Рис.13. Список городов во втором родителе

[2 1 3 5 4] => [1 3 4]

Рис.14. Удаление дублируемых городов

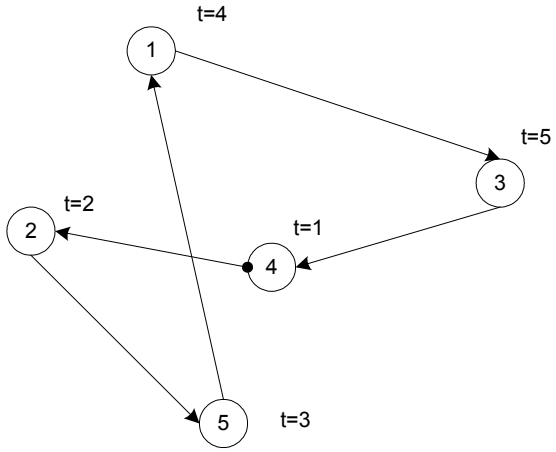

Рис.15. Потомок $V_7 = 42513$

2.3.3. Циклический кроссовер

В циклическом кроссовере каждый город и последовательность, в которой они посещаются, берется от одного родителя. Первый город потомка берется от первого родителя (рис.16). Второй город потомка берется у второго родителя из последней позиции (рис.17). В данном случае мы не можем взять его от второго родителя, так как этот город уже находится в хромосоме, поэтому мы его берем от первого родителя, и т.д. до тех пор, пока новая хромосома не будет создана.

$$V_1 = 1\,2\,5\,4\,3$$
$$V_6 = 3\,5\,4\,2\,1$$
$$\Downarrow$$
$$V_8 = 1$$

Рис.16. Копирование города из 1-го родителя

$$V_1 = 1\,2\,5\,4\,3$$
$$V_6 = 3\,5\,4\,2\,1$$
$$\Downarrow \quad \Downarrow$$
$$V_8 = 1 \quad 3$$

Рис.17. Копирование города из 2-го (1-го) родителя

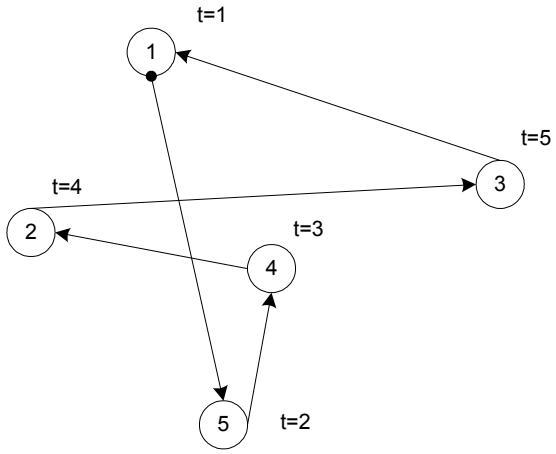

Рис. 18. Потомок $V_8 = 15423$

2.3.4. Кроссовер рекомбинации ребер

Кроссовер рекомбинации ребер делает потомство только с помощью ребер, представленных в обоих родителях. Сначала из двух родителей (рис.19) строится список ребер (табл.6). Города в потомках выбираются по одному, при этом выбирается город с наименьшим количеством ребер. Первым элементов в хромосоме является город с маленьким количеством связей (рис.20). После того как город выбран, он удаляется из таблицы и города, присоединенные к нему, рассматриваются как кандидаты при следующем выборе.

$$V_1 = 1\,2\,5\,4\,3$$

$$V_2 = 3\,5\,4\,2\,1$$

Рис.19. Исходные родители

Город	Соединен с		
1	2	3	
2	1	5	4
3	4	1	5
4	5	3	2
5	2	4	3

Табл.6. Список ребер

$$V_9 = 1\_\_\_\_$$

Рис.20. Хромосома на первом шаге алгоритма

Город	Соединен с		
2	5	4	
3	4	5	
4	5	3	2
5	2	4	3

Табл.7. Список ребер после первого шага

$$V_9 = 1\,3\_\_\_$$

Рис.21. Хромосома на втором шаге

Город	Соединен с	
2	5	4
4	5	2
5	2	4

Табл.8. Список ребер после второго шага

$$V_9 = 1\,3\,5\_\_$$

Рис.22. Хромосома на третьем шаге

Город	Соединен с
2	4
4	2

Табл..9. Список ребер после третьего шага

$$V_9 = 1\,3\,5\,2\_$$

Рис.23. Хромосома на четвертом шаге

Город	Соединен с
4	

Табл.10. Список ребер после четвертого шага

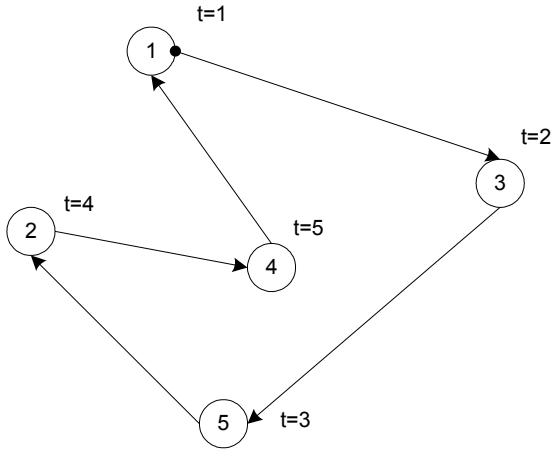

Рис.24. Полученный потомок $V_9 = 13524$

Этот тип кроссовера создает хромосомы, более чем в 95% случаев имеющие решения.

2.3.5. Улучшенный кроссовер рекомбинации ребер

Улучшенная версия кроссовера рекомбинации ребер состоит в сохранении записи ребер, представленных в обоих родителях и способствующих их выбору.

$$V_1 = 1\,2\,5\,4\,3$$

$$V_2 = 3\,5\,4\,2\,1$$

Рис.25. Исходные родители

Город	Соединен с		
1	2	3	
2	1	5	4
3	4	1	5
4	5	3	2
5	2	4	3

Табл.11. Список ребер

$$V_{10} = 1\_\_\_\_$$

Рис.26. Хромосома на первом шаге алгоритма

Город	Соединен с		
2	5	4	
3	4	5	
4	5	3	2
5	2	4	3

Табл.12. Список ребер после первого шага

$$V_{10} = 13\_\_\_$$

Рис.27. Хромосома на втором шаге

Город	Соединен с	
2	5	4
4	5	2
5	2	4

Табл.13. Список ребер после второго шага

$$V_{10} = 135\_\_$$

Рис.28. Хромосома на третьем шаге

Город	Соединен с
2	4
4	2

Табл.14. Список ребер после третьего шага

$$V_{10} = 1354\_$$

Рис.29. Хромосома на четвертом шаге

Город	Соединен с
4	

Табл.15. Список ребер после четвертого шага

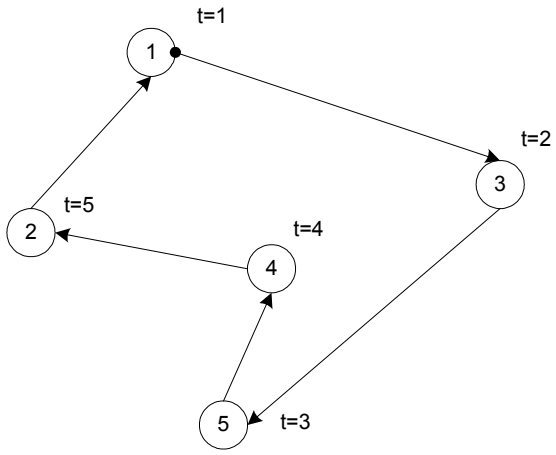

Рис.30. Полученный потомок $V_{10} = 13542$

2.3.6. Измененный кроссовер

Работа этого кроссовера может быть описана следующим образом. Выбирается точка сечения (рис.31), затем первая часть одного родителя копируется в первого потомка (рис.32). Во вторую часть потомка копируются гены второго родителя (рис.33). Если такие гены уже встречаются в потомке, то они пропускаются, а оставшуюся часть потомка дополняют гены первого родителя. Аналогичный алгоритм и для второго потомка.

$$V_1 = 12\ \ 543$$
$$V_2 = 35\ \ 421$$

Рис.31. Точка сечения кроссовера

$$V_{11} = 12\_\_\_$$

Рис.32. Потомок V_{11}

$$V_{11} = 124\_\_$$

Рис.33. Потомок V_{11}

$$V_{11} = 12453$$

Рис.34. Потомок V_{11}

2.3.7. Мутация

Мутация работает по следующему принципу. Выбирается случайным образом два города в хромосоме, и меняются местами.

$$V_1 = 12543$$
$$V_1' = 13542$$

Рис.35. Мутация хромосомы V_1 (2 и 5 элемент меняются местами)

2.3.8. *“Жадная мутация”*

Жадная мутация (*greedy reconnection*) состоит в упорядочивании последовательности городов. Ее применение помогает хорошо искать локальные оптимумы.

Сначала случайным образом выбираются две точки в хромосоме (рис.35) так, чтобы между ними было по крайне мере 2 точки. Затем внутренняя последовательность точек упорядочивается: высчитывается Эвклидово расстояние между двумя соседними точками, и они переставляются в зависимости от близости друг к другу. В нашем случае берется вторая точка (город №2) и сравнивается с соседними точками (город №1 и город №5). Пусть город №5 ближе к городу №1, чем город №2. Тогда переставляем города №2 и №5 местами (рис.36). Сравниваем расстояния городов между собой у третьей точки (города №5, №2, №4). Пусть город №4 ближе к городу №5, чем город №2. Меняем местами города №4 и №2 (рис.37). В результате имеем модифицируемую хромосому (рис.38).

$V_1 = 12543$

$V_1' = 1\_\_\_3$

Рис.35. Шаг 1

$V_{tmp} = 15243$

$V_1' = 15\_\_3$

Рис.36. Шаг 2

$V_{tmp} = 15423$

$V_1' = 154\_3$

Рис.37. Шаг 3

$V_1' = 15423$

Рис.38. Конечный потомок полученный при мутировании V_1

2.4. Функция приспособленности

Значение функции приспособленности – расстояние, которое проходит коммивояжер согласно пути (хромосоме). Поскольку все координаты городов задаются в двумерной системе координат, то расстояние пути у j-ой хромосомы высчитывается по формуле Эвклидова расстояния:

$$d_j = \sqrt{\sum_{i=0}^{n-1} [(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2]},$$

где x, y - координаты городов.

Поскольку это значение должно быть минимально, то конечная формула функции приспособленности выглядит следующим образом:

$$fitness_j = d_{\max} \cdot 1{,}1 - d_j,$$

где $d_{\max}$ - максимальное евклидово расстояние среди всей текущей популяции.

Значение этой функции, чем больше, тем лучше. Поскольку эта формула дает значение относительно текущей популяции, то при увеличении числа итераций это значение будет убывать.

2.5. Аналитические методы решения

Проблема коммивояжера может быть успешно решена аналитическим (не с помощью ГА) способом для не очень большого количества городов. Существенным недостатком этих методов является то, что при увеличении числа городов вычислительная сложность решения этой проблемы существенно возрастает и при некоторых условиях она уже не может быть решена за линейное время или решена, но не оптимально.

Самые простые из аналитических методов решений: полный перебор, кратчайший незамкнутый путь и метод ветвей и границ.

2.5.1. *Комбинаторное решение*

Комбинаторный способ решения состоит в переборе всех возможных вариантов решения и выборе лучшего. Этот способ наиболее неэффективный с точки зрения вычислительной сложности. Количество всех просматриваемых решений равно $n!$, где n - количество городов.

Преимущество этого метода в том, что полученное с его помощью решение является однозначно оптимальным.

2.5.2. *Кратчайший незамкнутый путь*

Один из вариантов аналитического решения этой проблемы – использование методов теории графов. Методы теории графов основаны на хорошо разработанных теоремах и аксиомах теории графов.

Строится кратчайший незамкнутый путь (КНП) следующим образом. Соединяются ребром две ближайшие точки, затем отыскивается точка, ближайшая к любой из уже рассмотренных точек, и соединяется с ней и т.д. до исчерпания всех точек.

Главный недостаток этого метода состоит в том, что этот метод одиночной связи приводит к “серпантинным” или “цепным” решениям.

2.5.3. Метод ветвей и границ

Метод ветвей и границ состоит в следующем. Для множества решений строится некоторая оценка, удовлетворяющая определенным свойствам. Разбиваем множество решений на несколько непересекающихся множеств, называемых концевыми множествами первого шага, и для каждого строим оценку. Выбираем концевое множество первого шага с минимальной оценкой (для задачи минимизации), разбиваем его на несколько непересекающихся концевых множеств второго шага. Теперь можем продолжить разбиения 2-мя методами:

1. **Метод полного ветвления**. Выбираем из концевых множеств текущего шага и концевых множеств предыдущего шага множество с наименьшей оценкой и разбиваем его. Продолжаем этот процесс, пока не получим одноэлементное множество, которое и будет оптимальным планом задачи
2. **Метод одностороннего ветвления.** Из концевых множеств текущего шага выбираем множество с наименьшей оценкой и разбиваем его. Продолжаем до тех пор, пока не получим одноэлементное множество. Запоминаем его оценку (первый рекорд). Отбрасываем все множества с оценкой, большей первого рекорда, из оставшихся выбираем произвольное и разбиваем его, пока не получим одноэлементное множество с оценкой, меньшей первого рекорда, запоминаем оценку (второй рекорд) и отбрасываем множества с оценкой большей, чем второй рекорд, из оставшихся выбираем произвольное и т.д., либо прекращаем разбиение множества, пока не получим на каком-нибудь шаге множество с оценкой, превышающей текущий рекорд, тогда выбираем любое из оставшихся и т.д.

3. Описание реализации

В ходе данной работы была создана программная среда для тестирования проблемы коммивояжера с помощью ГА. Особенностью данной реализации является то, что коммивояжер не возвращается в исходную точку. Ему задается начальная и конечная точка, и, используя все промежуточные города, он ищет оптимальный путь.

Реализованные операторы: кроссовер (см. пункт 2.3.6), мутация (см. пункт 2.3.7), жадная мутация (см. пункт 2.3.8) и селекция (колесо рулетки с пропорциональными значению приспособленности секторами).

В качестве значений по умолчанию используется: вероятность использования оператора кроссовера = 0.9, вероятность мутации = 0.3, вероятность жадной мутации = 0.95, размер популяции = 80 особей.

3.1. Примеры решения задач

3.1.1. Расположенные по кругу города

Демонстрация возможностей ГА нагляднее всего может быть проведена при решении задачи коммивояжера на городах образующих кругообразные фигуры, как-то круг, восьмерка, спираль. В данном случае – это круг. Очевидно, что кратчайший путь – это путь, описывающий этот круг.

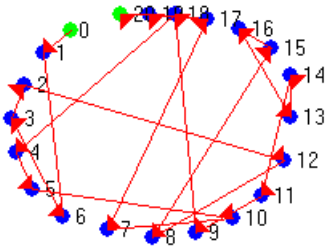

Рис.39. Поколение №1.
Минимальное расстояние = 1552

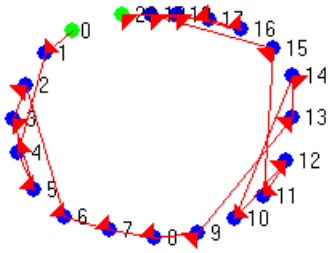

Рис.40. Поколение №35.
Минимальное расстояние = 942

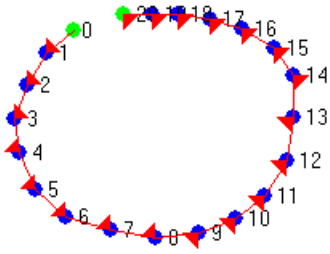

Рис.41. Поколение №64.
Минимальное расстояние = 470

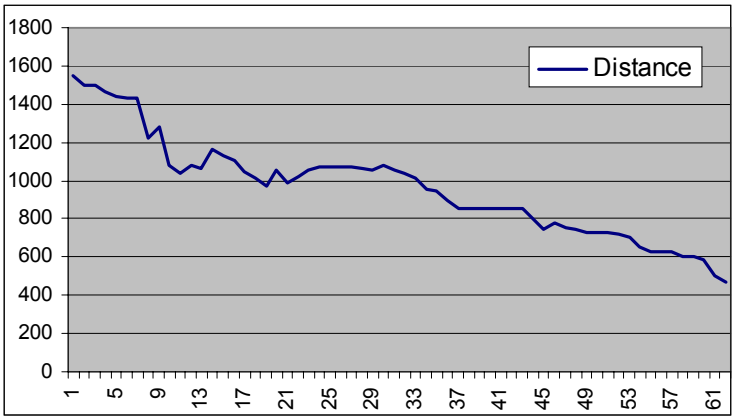

Рис.42. График зависимости минимальных расстояний (ось y) от поколений (ось x)

Лучшая особь для 20-ти городов была получена на 64-ом поколении. Расстояние, которое пройдет коммивояжер согласно этому маршруту = 470 у.е. Найденное решение является наилучшим.

3.1.2. Стохастически расположенные города

Рассмотрим задачу коммивояжера на 30-ти стохастически созданных городах.

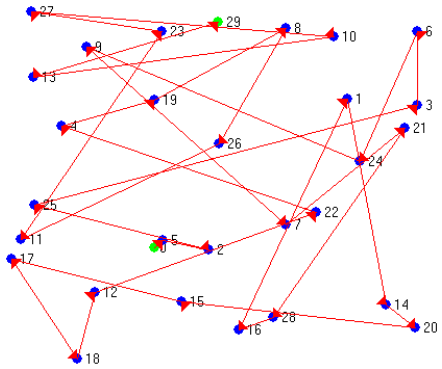

Рис.43. Поколение №1.
Минимальное расстояние = 5008

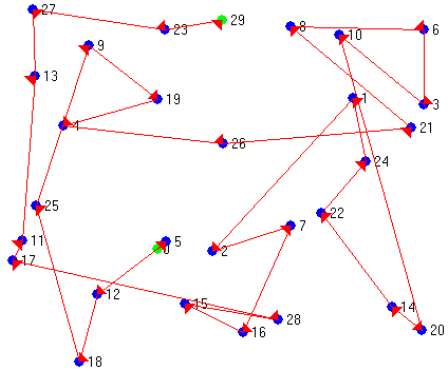

Рис.44. Поколение №56.
Минимальное расстояние = 3205

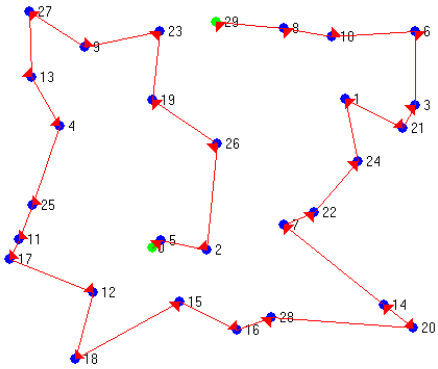

Рис.45. Поколение №600.
Минимальное расстояние = 1854

Лучший маршрут получен на 600-ой итерации. Дальнейшие итерации улучшений не давали.

3.2. Программная реализация

Программная реализация (рис.43) состоит из карты, на которой расположены города. При старте автоматически создается карта из 20-ти стохастически расположенных городов. Возможно рисование своей карты.

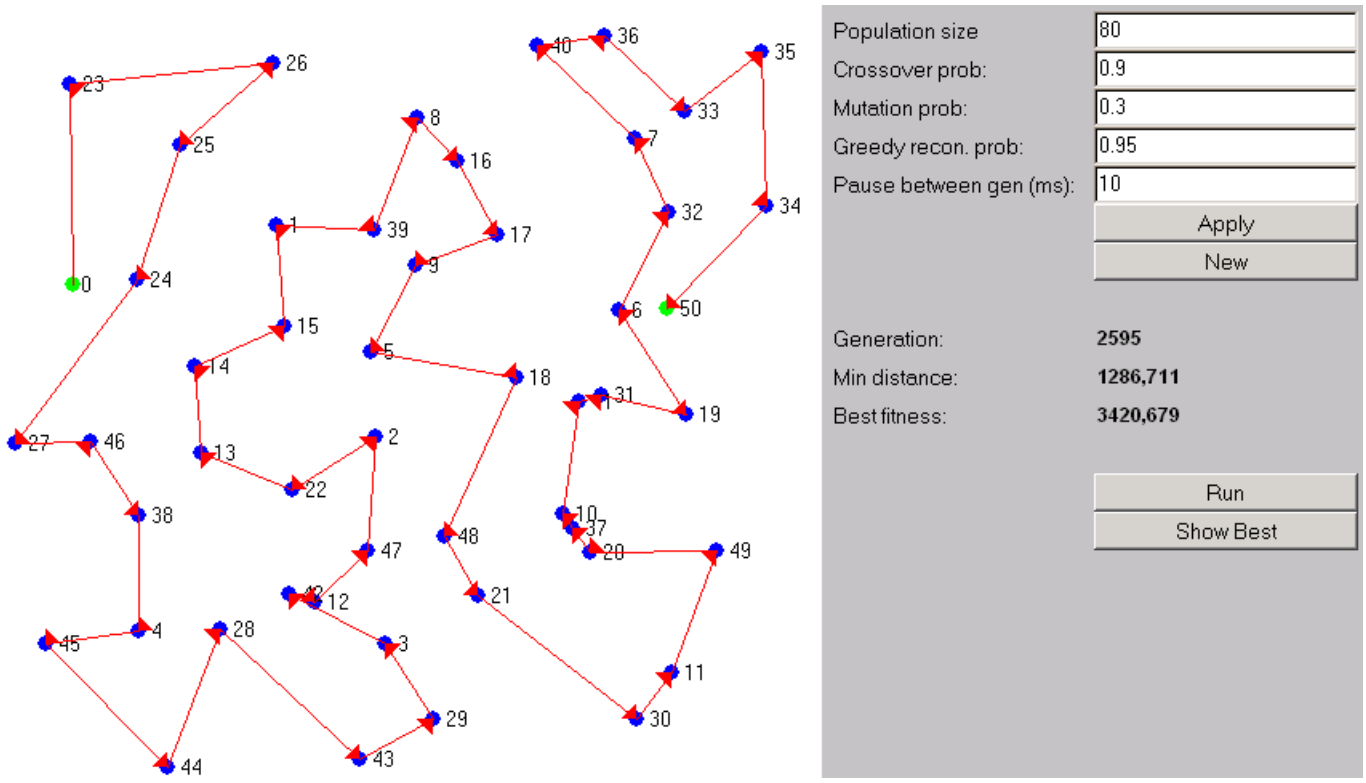

Рис.46. Графический вид программы

4. Заключение

Задача нахождения кратчайшего пути среди 20-ти городов требует перебора около $2.4 \cdot 10^{24}$ всех возможных вариантов путей (см. пункт 2.5.1). При переборе 500 миллионов путей в секунду потребовалось бы 150 лет для вычисления лучшего пути. При применении ГА около-оптимальное, или даже иногда оптимальное, решение может быть найдено всего за несколько минут.

В данной работе было наглядно продемонстрировано применение теории генетических алгоритмов в контексте транспортной задачи. Показано преимущество использования символьного представления хромосом, по сравнению с обычным (двоичным). Описана работа шести типов оператора кроссовера и двух типов оператора мутации для символьного представления. Приведены также возможные аналитические методы решения данной задачи.

5. Используемая литература

1. Holland, J. H. "*Adaptation in Natural and Artificial Systems*". Ann Arbor: The University of Michigan Press, 1975.
2. R. Poli. "*Introduction to Evolutionary Computation*". Lectures notes. School of Computer Science, The University of Birmingham, 1996. http://www.cs.bham.ac.uk/~rmp/slide_book/