УДК 519.856

В.А. Моров

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА К ЗАДАЧАМ ОПТИМИЗАЦИИ. РЕАЛИЗАЦИЯ ГЕНЕТИЧЕСКОГО АЛГОРИТМА ДЛЯ ЗАДАЧИ КОММИВОЯЖЕРА

В статье рассматриваются применение генетического алгоритма к задачам оптимизации и реализация его для задачи коммивояжера. Описаны основные элементы алгоритма и построенная программа для достижения поставленной цели.

Application of genetic algorithm to the optimization problems and algorithm's realization for the Traveling Salesman Problem (TSP) is considered in this article. There are described common elements of the algorithm and constructed program of solution of the problem.

Введение

Задачей оптимизации в математике, информатике и исследовании операций называется задача нахождения экстремума (минимума или максимума) целевой функции в некоторой области конечномерного векторного пространства, ограниченной набором линейных и/или нелинейных равенств и/или неравенств.

В процессе проектирования обычно ставится цель определить в некотором смысле наилучшие структуру или значения параметров объектов. Такая задача называется оптимизационной. Если оптимизация связана с расчетом оптимальных значений параметров при заданной структуре объекта, то ее именуют параметрической оптимизацией. Задача выбора оптимальной структуры является структурной оптимизацией.

Стандартная математическая задача оптимизации формулируется таким образом. Среди элементов *x*, образующих множество *X*, найти такой элемент *x**, который доставляет минимальное значение *f(x*)* заданной функции *f(x)*. Чтобы корректно поставить задачу оптимизации, необходимо задать: допустимое множество *X*; Целевую функцию — отображение $f : X \to \mathrm{R}$; критерий поиска (max или min).

Если минимизируемая функция не является выпуклой, то часто ограничиваются поиском локальных минимумов и максимумов – точек x_0 таких, что всюду в некоторой их окрестности $f(x) \geq f(x_0)$ для минимума и $f(x) \leq f(x_0)$ – для максимума.

Если допустимое множество $X = \mathrm{R}^n$, то такая задача называется задачей безусловной оптимизации, в противном случае – задачей условной оптимизации.

В зависимости от природы множества X задачи математического программирования классифицируются как:

задачи дискретного программирования (или комбинаторной оптимизации), если X конечно или счетно;

задачи целочисленного программирования, если X – подмножество множества целых чисел;

задачи нелинейного программирования, если ограничения или целевая функция содержат нелинейные функции и X является подмножеством конечномерного векторного пространства.

Если же все ограничения и целевая функция содержат лишь линейные функции, то это задача линейного программирования.

Существует множество методов оптимизации, которые можно разделить на три группы: детерминированные; случайные (стохастические); комбинированные.

В частности, большой интерес представляют собой эволюционные методы, являющиеся стохастическими. Упоминания о применении генетических алгоритмов для решения задачи оптимизации относятся к концу 1960-х гг. Эволюционные методы основываются на примере работы эволюции и обучения, к таким методам относят нейронные сети, генетические алгоритмы[1].

Генетический алгоритм – это эвристический алгоритм поиска, используется для решения задач оптимизации и моделирования путем случайного подбора, комбинирования и вариации искомых параметров с применением механизмов, напоминающих биологическую эволюцию, является разновидностью эволюционных вычислений. Отличительная особенность генетического алгоритма – акцент на использовании оператора «скрещивания», производящего операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе[1]. Схематически алгоритм представлен на рис. 1.

Для применения алгоритма задачи приводятся к виду, при котором решение может быть представлено как набор более мелких составных частей (аналог генотипа и его составных частей – генов). Длина генотипа может быть как фиксированной, так и переменной.

Сам алгоритм состоит из нескольких шагов.

0. Подготовительный шаг – формирование начальной популяции (начального набора решений). Алгоритм для формирования может быть различным, но чаще всего используют случайный процесс с целью охватить большее разнообразие для поиска решений. Возможно применение других способов формирования, – например, с заранее известными свойствами, но следует иметь в виду, что это может повлиять на ход развития системы в дальнейшем.

1. Отбор – важный этап в алгоритме, отвечает за выбор направления развития популяций, чаще всего отбрасываются решения с низким значением функции приспособленности (fitness function), что способствует улучшение средней приспособленности всей популяции.

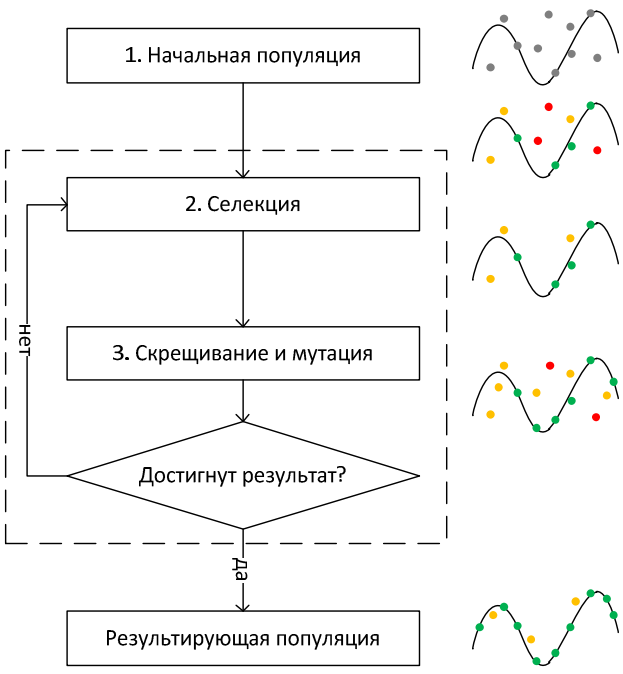

Рис. 1. Схема работы генетического алгоритма.

2. Скрещивание – этап, на котором происходит образование новых решений в популяции, прошедшей через отбор, для восстановления численности. Особенность его в том, что при использовании скрещивания берутся два или более существующих решений в популяции, а из них – составные части (гены) и соединяются в новом решении, которое остается в популяции.

3. Скрещивание не позволяет в полной мере охватить все возможные варианты сочетаний и значений генов, поэтому не менее важен процесс мутации. Он состоит в том, что в некоторых решениях из популяции происходят случайные изменения в генах. Этот процесс способствует увеличению

разнообразия в популяции.

4. Оценка решений и остановка алгоритма – в большинстве случаев; если для решения задачи необходимо применять генетический алгоритм, то нет критерия останова, основанного на самих решениях, вместо него применяется подход с числом вычислений (числом создаваемых популяций). Иногда останов можно производить заранее, если возможен случай вырождения популяций.

Основными шагами алгоритма являются шаги с 1 по 4, один проход по которым «создает новую популяцию».

Эвристические алгоритмы позволяют решать практически любые задачи оптимизации. Но их эффективность ниже, чем у локальных методов. Таким образом, фактически данные алгоритмы, хотя и могут использоваться для решения любых задач, чаще всего применяются к задачам, для которых не разработаны специальные локальные методы или решение такими методами является при заданных параметрах неэффективным [2].

К подобным задачам можно отнести очень популярную задачу оптимизации – задачу коммивояжера. При определенных условиях решение ее с помощью известных точных методов становится невозможным из-за большого числа вариантов. Рассмотрим эту задачу подробнее.

Задача коммивояжера – одна из самых известных задач комбинаторной оптимизации, заключающаяся в отыскании самого выгодного маршрута, проходящего через указанные города хотя бы по одному разу, с последующим возвратом в исходный город. В условиях задачи указываются критерий выгодности маршрута (кратчайший, самый дешевый, совокупный критерий и т.п.) и соответствующие матрицы расстояний, стоимости и т.д. Как правило, указывается, что маршрут должен проходить через каждый город только один раз – в таком случае выбор осуществляется среди гамильтоновых циклов[3].

Существует несколько частных случаев общей постановки задачи, в частности:

геометрическая задача коммивояжера (также называемая планарной или евклидовой, когда матрица расстояний отражает расстояния между точками на плоскости);

треугольная задача коммивояжера (когда на матрице стоимостей выполняется неравенство треугольника);

симметричная и асимметричная задачи коммивояжера.

Существует и так называемая обобщенная задача коммивояжера[3].

Решим данную задачу с помощью генетического алгоритма.

Постановка задачи коммивояжера

Рассмотрим метрическую задачу коммивояжера, когда расстояния между городами можно вычислить (аналог точек на плоскости). При данной постановке задачи мы имеем: число городов, координаты каждого города на плоскости:

$$\{g = (x, y)\},\ i = \overline{1, n}.$$

Таким образом, расстояние между городами можно находить как расстояние между двумя точками:

$$S(g_i, g_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Необходимо найти такой путь через города g_i, чтобы суммарное расстояние были минимальным.

Построение генетического алгоритма для задачи коммивояжера

Для применения генетического алгоритма необходимо определить основные структурные элементы: вид элемента популяции, процесс скрещивания, мутации и вид фитнесс-функции.

За элемент популяции (одно возможное решение) принимаем маршрут через все города. Каждый такой маршрут является возможным решением и не противоречит условиям задачи, хотя может быть совсем не оптимальным. Предполагаем, что все элементы популяции корректны, т.е. все они – потенциальные решения поставленной задачи и не являются противоречивыми.

В качестве фитнесс-функции мы принимаем функцию вида:

$$S = \sum_{i,j \in P} S(g_i, g_j),$$

где P – множество всех связей в маршруте.

Данная функция определяет минимизируемый параметр и позволяет оценивать получаемые решения.

Метод мутации реализуется следующим образом: выбираем случайный город; находим связи соответствующие этому городу; соединяем соседние города прямой связью (исключаем выбранный город из этой связи); вставляем город в случайное место.

Пример.

Предположим, что решение до мутации имеет вид, представленный на рис. 2.

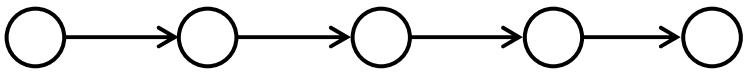

Рис. 2. Решение до мутации.

Применяя к нему алгоритм мутации, можем получить решение в виде представленном на рис. 3.

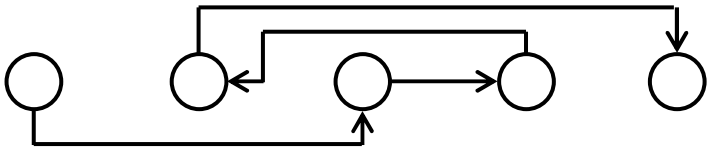

Рис. 3. Решение, но после мутации.

Метод скрещивания реализуется сложнее, так как требует дополнительных проверок на корректность получаемого маршрута.

Алгоритм метода скрещивания:

выбираем два маршрута из популяции, которые будут выступать в роли родителей;

в образуемые маршруты переносим связи, которые существуют в обоих маршрутах-родителях;

при несовпадающих связях предпочтение отдается связи в первом родителе для городов с четными номерами; если ее применить невозможно, то берем связи из второго родителя (предпочитаются связи из второго маршрута для городов с нечетными номерами), если это также невозможно, то берем из первого маршрута.

При таком задании скрещивания для получения двух решений можно применять данный метод дважды, меняя маршруты-родители местами. Тогда второй потомок будет получен при условии, что при несовпадении связей предпочтение будет отдаваться второму маршруту.

Пример. На рис. 4 представлен пример применения данного алгоритма скрещивания к двум решениям:

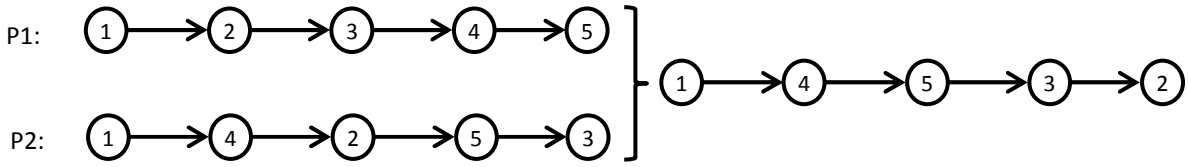

Рис. 4. Пример «скрещивания».

Анализ полученных результатов

Применяя данный алгоритм при различных начальных условиях (размер популяций, процент мутаций, вес ближайших городов), получаем различные результаты.

1) 15 городов.

Размер популяции 1000, 15% мутации, вес ближайших городов 0%.

Результаты:

число итераций до вырождения: 1200,

время выполнения: 0.321 с,

погрешность от оптимального (если возможно посчитать): <5%.

2) 40 городов.

Размер популяции 10000, 15% мутации, вес ближайших городов 0%.

Результаты:

число итераций до вырождения: 143000 / 375000,

время выполнения: 4.3 с / 9 с.

3) 40 городов.

Размер популяции 10000, 5% мутации, вес ближайших городов 50%.

Результаты:

число итераций до вырождения: 101000,

время выполнения: 3.7 с.

Последние два примера: первое минимально найденное значение с погрешностью между ними в 3% было найдено менее чем за 5 сек. Последующие решения отличаются погрешностью меньше 1%. При правильном задании размеров начальной популяции и процента мутации можно ускорить работу алгоритма.

1. Гладков, Л.А. Генетические алгоритмы: Учебное пособие / Л.А. Гладков, В.В. Курейчик, В.М. Курейчик. – Изд. 2-е. – М.: Физматлит, 2006. – 320 с.

2. Курейчик, В.М. Поисковая адаптация: теория и практика / В.М. Курейчик, Б.К. Лебедев, О.К. Лебедев. – М.: Физматлит, 2006. – С. 272.

3. Johnson, D.S., McGeoch, L.A. The traveling salesman problem: a case study. Local search in combinatorial optimization / D.S. Johnson, L.A. McGeoch. – Chichester: Wiley. – P. 215-310.