Исследование эвристического метода решения задачи коммивояжера

Борознов В.О. (bor_vlad@rambler.ru)

Астраханский Государственный Технический Университет

Введение

Многие практические задачи железнодорожного транспорта сводятся к задаче коммивояжера, задаче отличающейся простотой постановки и исключительной сложностью решения. Данная задача относится к классу NP-полных задач. Традиционные методы решения этой задачи имеют массу недостатков, в том числе – «ловушки» локальных оптимумов. В работе представлены результаты исследования нетрадиционного эвристического метода для решения задачи коммивояжера.

Постановка задачи

Коммивояжер, выходящий из какого-нибудь города, желает посетить N-1 других городов и вернуться к исходному пункту. Известны расстояния между всеми этими городами. Требуется установить в каком порядке он должен посещать города, чтобы общее пройденное расстояние было минимальным [1]. Решение этой задачи существует и его можно найти следующим образом: перебрать все маршруты и выбрать из них наименьший. Но все дело в том, что ни одна, даже самая быстродействующая вычислительная машина не в состоянии осуществить этот перебор при сколько-нибудь значительном числе.

Задача коммивояжера (ЗК) представляет собой задачу отыскания кратчайшего Гамильтонова пути в полном конечном графе с N вершинами. Все известные методы нахождения точного решения включают в себя поиск в пространстве решений, которое увеличивается экспоненциально от N. Так как пространство решений слишком большое (N!), то задача решается с помощью эвристических методов в тех случаях, когда N не является незначительно малым.

Описание метода

Предлагаемый метод, был выведен экспериментальным путём на основе сравнений точных решений с решениями, полученными эвристическими алгоритмами – «жадным», методом включения дальнего. Данный метод основан на анализе имеющегося эталонного маршрута и его оптимизации назовём его BV методом. Алгоритм условно состоит из трёх этапов: 1) получение начального эталонного решения, 2) построение BV матрицы, 3) оптимизация текущего решения. Начальное решение представляет собой лучшее решение из всех решений полученных на основе «жадного» метода. Для его реализации строиться **N** маршрутов, по количеству городов. В каждом из этих **N** маршрутов, начальный город равен номеру маршрута, а все последующие города выбираются по принципу жадного алгоритма (выбираем ближайший город). Например:

1; 7; 3; 4; 2; 5; 6;
2; 5; 6; 1; 3; 7; 4;
……
……
7; 4; 5; 2; 1; 3; 6;

Далее, как уже было сказано выше, среди них выбирается наилучший маршрут, то есть, с наименьшей длиной пути и он помечается как эталонный.

Алгоритм построения начального эталонного решения.
1.Инициализируется матрица начальных решений размерность $N * N$, p=1, q=1.
2.Город p записывается, в матрицу $[p,q]$ и становиться текущим.
3.Выбирается очередной ранее не посещённый город ближайший к текущему городу q. Записывается его матрицу$[p,q]$, после чего он становиться текущим.
4.Повторяем пункт 3 до тех пор, пока не обойдём все города.
5.Увеличиваем номер города на единицу $p++$ и повторяем пункты 2 – 4, пока номер не станет равным $p=N+1$.
6.Из полученых N маршрутов выбираем лучший и помечаем его как эталон.

Второй этап алгоритма – построение BV-матрицы. Для этого строиться матрица оценок размером $N \times N$, первоначально заполненная нулями. Далее просматривая N построенных «жадных» маршрутов, для каждой пары соседних городов p и q, соответствующий элемент матрицы оценок $a[p,q]$ увеличивается на единицу.

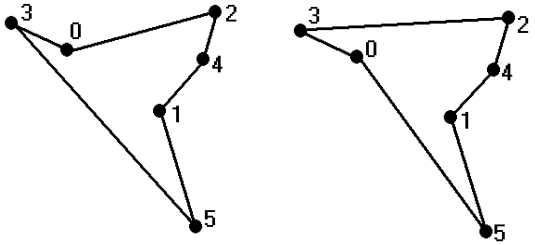
Алгоритм построения BV - матрицы.
1.Инициализируем BV – матрицу размерностью $N*N$ (заполняем нулями).
2.В цикле осуществляем проход по каждому из N маршрутов (матрица начальных решений).
3.Выбираем два соседних города из маршрута в матрице начальных решений и по значению этих городов сопоставляем им строку и столбец BV – матрицы.
4.Увеличиваем значение BV – матрицы [выбранная строка, выбранный столбец] на единицу.

Третий этап алгоритма, состоит в преобразовании эталонного маршрута с помощью последовательного применения BV-модификаторов к каждой паре городов, которым соответствует ненулевой элемент матрицы оценок. Если преобразованный маршрут имеет меньшую длину пути, то он становиться эталонным и в матрице оценок соответствующий элемент заменяется на 0. То есть, каждый BV – модификатор в качестве параметра вызова получает номера (*позиции*) двух городов, которым в BV – матрице сопоставлено любое ненулевое число. Результатом вызова является модифицированная целевая функция. При каждом вызове, каждый BV – модификатор хранит в себе наилучшее значение модифицированной целевой функции и пары городов. После прохода по всей BV – матрице, из всех BV – модификаторов выбирается наилучшее значение модифицированной целевой функции, и сравнивается с эталоном. Если модифицированный маршрут имеет превосходство, то он становится эталоном. А в BV – матрице для данной пары городов присваивается значение 0. Проход по BV – матрице повторяется до тех пор, пока модифицированный маршрут имеет превосходство над эталоном. Отсутствие превосходства модифицированного маршрута над эталонном, говорит о том, что полученное решение является конечным для данного метода решения.

Рассмотрим каждый из BV – модификаторов. Но для начала, скажем, что каждый BV – модификатор предназначен для определённой ситуации. Экспериментальным путем, было, выделено четыре модификатора.

Алгоритм BV – модификатора «Сглаживания».
1.В качестве параметра вызова получает позиции двух городов.
2.Меняем местами города (их позиции).
3.Вычисляем целевую функцию и возвращаем её как результат вызова.

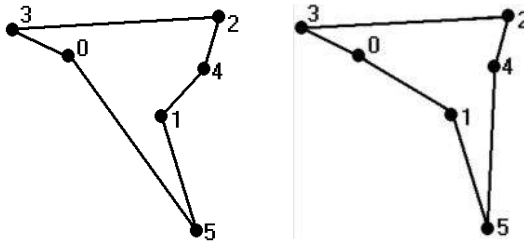
Результат применения данного модификатора показан на рисунке 1. Если построить расположение городов по матрице расстояний, то результатом применения модификатора будет сглаживание «углов» в маршруте.



1 4 2 **0** **3** 5 1 1 4 2 **3** **0** 5 1

Рисунок 1. Результат применения BV – модификатора «Сглаживания»

Алгоритм BV – модификатора «Перенос».
1.В качестве параметра вызова получает позиции двух городов.
2.Сохраняем значение первой позиции.
3.Осуществляем сдвиг городов влево на 1, от первой до второй позиции.
4.Второй позиции присваиваем значение ранее сохранённого города из первой позиции.
5.Вычисляем целевую функцию и возвращаем её как результат вызова.
Результат применения данного модификатора показан на рисунке 2.

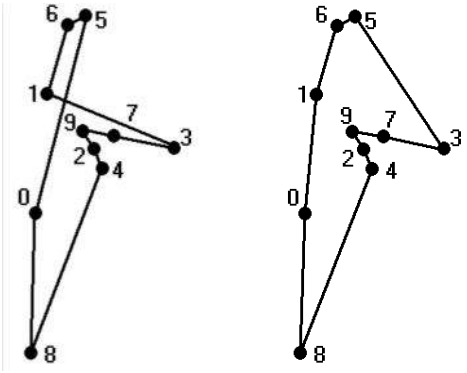


1 4 2 3 **0** 5 1 4 2 3 **0** **1** 5 1

Рисунок 2. Результат применения BV – модификатора «Перенос»

Алгоритм BV – модификатора «Инверсный переход».
1.В качестве параметра вызова получает позиции двух городов.
2.Копируем последовательность городов от первой +1 до второй –1 позиции в массив.
3.Осуществляем инверсную (*в обратном порядке*) запись из массива в исходный маршрут.
4.Вычисляем целевую функцию и возвращаем её как результат вызова.

Результат применения данного модификатора показан на рисунке 3. Если построить расположение городов, то в исходном варианте участок (1 - 2), подвергаемый преобразованию, будет содержать пересечение маршрута. После преобразование пересечения в маршруте не будет.

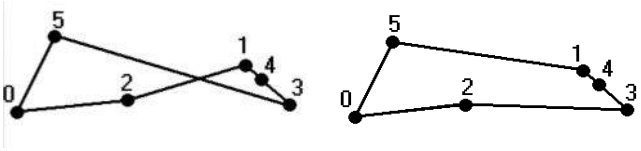


7 **3** 1 6 5 **0** 8 4 2 9 7 7 **3** 5 6 1 **0** 8 4 2 9 7

Рисунок 3. Результат применения BV – модификатора «Инверсный переход»

Алгоритм BV – модификатора «Инверсный перенос».
1.В качестве параметра вызова получает позиции двух городов (по матрице решений).
2.Копируем последовательность городов от первой до второй позиции в массив.
3.Осуществляем инверсную (*в обратном порядке*) запись из массива в исходный маршрут.
4.Вычисляем целевую функцию и возвращаем её как результат вызова.

Результат применения данного модификатора показан на рисунке 3. Аналогичен «Инверсному переходу» с одним отличием, города ограничивающие участок меняются местами.



5 0 **2** 1 4 3 5 **2** 0 **5** 1 4 3 2

Рисунок 4. Результат применения BV – модификатора «Инверсный перенос».

Следует отметить, что при последовательном вызове происходит вызов не всех модификаторов, а только некоторых из них. Поскольку каждый модификатор имеет свои ограничения (расстояние между позициями) перед каждым вызовом BV – модификатора осуществляется проверка условия вызова. Маршрут рассматривается как кольцевой список.

Результаты исследования

Оценим результативность данного метода, сравнив его с алгоритмом полного перебора (АПП). Для каждой задачи *N* размерности производилась по 33 испытания, что соответствует доверительной вероятности 0.75 и ошибки 0.1. Случайным образом строилась матрица расстояния заданной размерности *N*, после чего задача решалась обоими методами. Результаты испытаний приведены в таблице 1.

Таблица 1. Результаты сравнения BV – метода с АПП

N	Среднее время полного перебора, мс.	Среднее время BV – метода, мс.	Средняя погрешность в %
3	11,42	19,84	0,0193
4	21,86	21,40	0
5	37,24	21,84	0
6	71,32	21,02	0,1122
7	100,96	20,22	0,0356
8	136,00	19,82	0,0766
9	444,46	15,22	0,0949
10	3984,56	20,82	0,0582
11	39577,9	20,40	0,2525

Следует отметить, что данный метод не позволяет получить точного решения, а даёт лишь приблизительное решение. Как правило, это обусловлено неполнотой BV – матрицы, а так же наличием одинаковых расстояний между городами. На рисунке 5 показана экспериментально полученная зависимость абсолютного отклонения от размерности задачи.

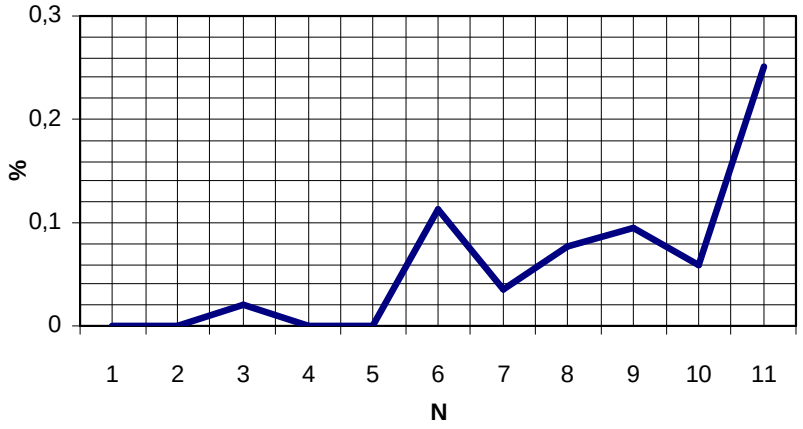


Рисунок 5. Абсолютное отклонение BV – метода от размерности задачи.

Большим недостатком метода полного перебора является временная зависимость от размерности задачи, что делает данный метод не пригодным для решения практических задач. По этому для решения данной задачи используют приближенные методы решения. Самым оптимальным (результат/время) среди них является Генетические Алгоритмы (ГА).

Так же для сравнения с разработанным методом, предлагается эвристический метод включения дальнего (МВД). Результат сравнения представлены в таблице 2. Для каждой задачи *N* размерности производилась по 33 испытания, что соответствует доверительной вероятности 0.75 и ошибки 0.1. Случайным образом строилась матрица расстояния заданной размерности *N*, после чего задача решалась тремя методами. ГА были взяты с параметрами: вероятность кроссенговера 0.85, вероятность мутация – 0.1, критерий окончания эволюции: за последние 25 поколений целевая функция не улучшилась. Среднее отклонение считалось относительно BV - метода.

Таблица 2. Результаты сравнения BV - метода с ГА и МВД

N	Среднее время BV метода, мс.	Среднее время метода включения дальнего, мс	Среднее время ГА, мс	Отклонение МВД от BV, %	Отклонение ГА от BV, %
20	25,78	7,57	245,18	7,99	2,85
40	84,66	10,93	740,78	11,56	3,81
60	401,12	20,36	1954,75	13,18	3,54
80	918,3	57,96	4130,72	13,89	5,27
100	2856,96	110,93	7139,06	14,04	5,05
120	5504,69	213,24	12625,78	13,96	6,34
140	9337,42	462,18	21799,60	15,33	6,86
160	16187,54	622,96	31461,36	15,18	6,71

Для того, что бы оценить эффективность разработанного метода, построим зависимости времени (рисунок 6) и относительной погрешности (рисунок 7) от размерности задачи.

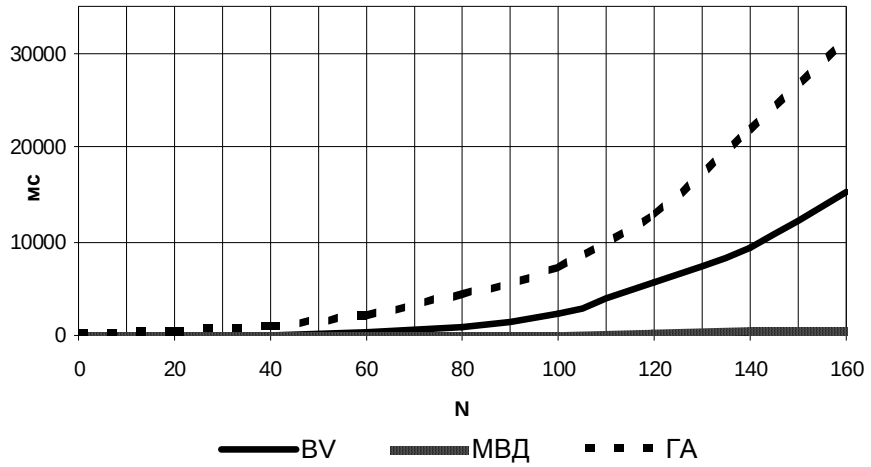


Рисунок 6. Временная зависимость от размерности задачи

Как видно из рисунка 6 описанный метод по времени затраченному на вычисления задачи оказался эффективней ГА. Это более заметно при увеличении размерности задачи. Быстрое получение результата BV – методом обеспечивается как за счёт простоты самого метода, так и самих BV – модификаторов.

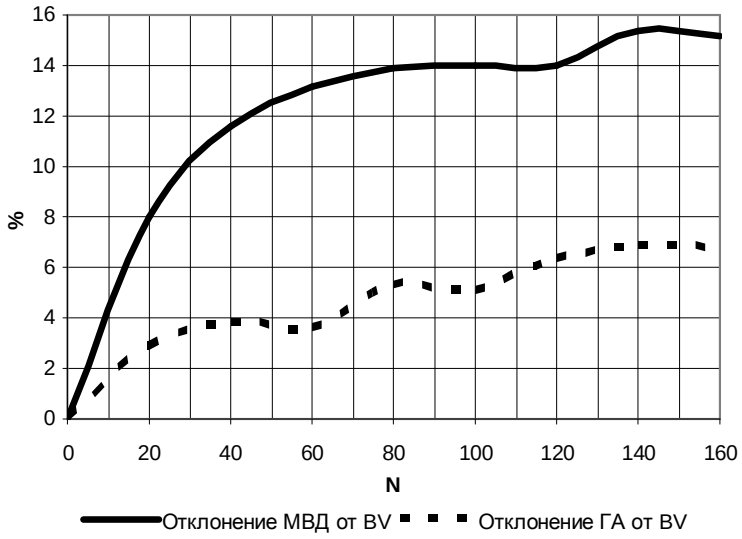


Рисунок 7. Относительная погрешность от размерности задачи

Относительная результативность нового метода отчётливо показана на рисунке 7. Из данного графика видно, что данный метод имеет превосходство в полученных результатах, как со стороны ГА, так и метода включения дальнего. Причём с ростом размерности задачи превосходство становиться более значимым. Что свидетельствует о целесообразности изучения и развития данного метода.

Заключение

Результаты показали, предлагаемый в работе эвристический метод решения ЗК не позволяет получить точного решения, но на малых размерностях (N<12) его погрешность очень мала <0,16%.

Экспериментальные данные сравнения с самым оптимальным эвристическим методом (результат/время) ГА – показали, что рассматриваемый метод имеет превосходство как в качестве найденного решения, так и в скорости (причём с ростом размерности задачи превосходство растёт).

Рассматриваемый в статье метод решения ЗК может иметь широкий спектр применения, так как он не ограничен ни размерностью задачи (решает задачу за разумное время) ни качеством решения.

Литература

1. В.М. Курейчик. Применение генетических алгоритмов для решения комбинаторно-логических задач оптимизации. Интеллектуальные САПР. Междуведомственный тематический научный сборник. Выпуск 5, Таганрог, 1995 г.

2. Дж. Макклониелл. Основы современных алгоритмов. 2-е дополненное издание. Москва. 2004 г.