



Nikita Starichkov @demist

Пользователь

21,2

карма

0,0

рейтинг



Профиль

5

Публикации

67

Комментарии

0

Избранное

5

Подписчики

24 ноября 2012 в 19:00

Разработка → Поиск гамильтонова цикла в большом графе (задача коммивояжера).Часть 1

из песочницы

Программирование*

1. Постановка задачи

Полный взвешенный граф из 500 вершин задан матрицей смежности.

Необходимо найти гамильтонов цикл в этом графе как можно меньшей суммарной стоимости.

2. Решение

1. Жадный алгоритм

Тут все просто, запускаемся из любой вершины, выбираем минимальную стоимость ребра из тех, по которым можно идти, переходим в вершину x , нашу вершину вычеркиваем, в вершине x делаем тоже самое и т.д.

Просто и быстро.

Однако ответ может быть очень и очень далеким от правды.

Жадина хорошо работает на случайном графе, однако тут мы ничего про граф не знаем.

Поэтому идем дальше.

2. Честный перебор

Задача коммивояжера, она же поиск гамильтонова цикла минимальной суммарной стоимости — NP-тяжелая, так что полный перебор для такого графа может продлиться вечность, поэтому данная идея не подходит.

3. Метод ветвей и границ

Вторая идея, пришедшая в голову — использовать улучшенный перебор, откидывая на каждом шаге алгоритма явно неоптимальные решения.

Один из наиболее эффективных алгоритмов такого рода является метод ветвей и границ («поиск с возвратом», «backtracking»), разработанный Литтлом, Мерти, Суини, Кэрелом в 1963 г.

Пусть $S(0)$ — множество всех допустимых замкнутых маршрутов задачи о коммивояжере с n городами и матрицей затрат c . Метод Литтла основан на разбиении множества на два непересекающихся подмножества и на вычислении оценок каждого из них. Далее подмножество с минимальной оценкой (стоимостью) разбивается на два подмножества и вычисляются их оценки. На каждом шаге выбирается подмножество с наименьшей из всех полученных на этом шаге оценок и производится его разбиение на два подмножества. В конце концов получаем подмножество, содержащее один цикл (замкнутый маршрут, удовлетворяющий наложенным ограничениям), стоимость которого минимальна.

Алгоритм состоит из двух этапов:

Первый этап

Приведение матрицы затрат и вычисление нижней оценки стоимости маршрута g .

1. Вычисляем наименьший элемент в каждой строке(константа приведения для строки)
2. Переходим к новой матрице затрат, вычитая из каждой строки ее константу приведения
3. Вычисляем наименьший элемент в каждом столбце(константа приведения для столбца)
4. Переходим к новой матрице затрат, вычитая из каждого столбца его константу приведения.

Как результат имеем матрицу затрат, в которой в каждой строчке и в каждом столбце имеется хотя бы один нулевой элемент.

5. Вычисляем g как сумму констант приведения для столбцов и строк.

Второй (основной) этап

- 1.Вычисление штрафа за неиспользование для каждого нулевого элемента приведенной матрицы затрат.

Штраф за неиспользование означает элемента с индексом (h,k) в матрице, означает, что это ребро не включается в наш маршрут, а значит минимальная стоимость «неиспользования» этого ребра равна сумме минимальных элементов в строке h и столбце k .

- а) Ищем все нулевые элементы в приведенной матрице
- б) Для каждого из них считаем его штраф за неиспользование.
- в) Выбираем элемент, которому соответствует максимальный штраф

Почему максимальный? Потому что это означает, что исключение из маршрута этого ребра приведет к максимальному увеличению стоимости оптимального маршрута, который мы как раз и ищем.

2. Теперь наше множество $S(0)$ разбиваем на два — Sw (содержащие ребро h,k) и Sw/o (не содержащие ребро h,k)

3. Вычисление оценок затрат для маршрутов, входящих в каждое из этих множеств.

а) Для множества Sw/o все просто: раз мы не берем ребро (h,k) , то для него оценка затрат равна оценке затрат множества $S(0)$ + штраф за неиспользование ребра (h,k)

б) При вычислении затрат для множества Sw примем во внимание, что раз ребро (h,k) входит в маршрут, то значит ребро (k,h) в маршрут входить не может, поэтому в матрице затрат пишем $c(k,h)=\text{infinity}$, а так как из пункта h мы «уже ушли», а в пункт k мы «уже пришли», то ни одно ребро, выходящее из h , и ни одно ребро, приходящее в k , уже использоваться не могут, поэтому вычеркиваем из матрицы затрат строку h и столбец k . После этого приводим матрицу, и тогда оценка затрат для Sw равна сумме оценки затрат для $S(0)$ и $r(h,k)$, где $r(h,k)$ — сумма констант приведения для измененной матрицы затрат.

4. Из множеств Sw и Sw/o выбирается то, которое имеет наибольшую оценку.

Так продолжаем, пока в матрице затрат не останется одна невычеркнутая строка и один невычеркнутый столбец.

Теперь результаты.

Если на каждом шаге выбирать только из двух множеств (Sw и Sw/o) то алгоритм работает вполне вменяемое время, однако точность просто ужасна (проигрывает ОЧЕНЬ много жадине из п.1)

Если же на каждом шаге выбирать лучшее множество из всех, полученных к этому шагу и не использовавшихся до этого, то для маленьких графов (порядка 40-50) вершин, получается хорошая точность, но 500 вершин дождаться не удалось.

Поэтому в голову приходит следующая идея:

4. Метод ветвей и границ с эвристикой

Да, правда, почему бы нам не ввести эвристику? Ведь в алгоритме ветвей и границ мы фактически строим дерево, в узлах которого решаем брать ребро (x,y) или нет, и вешаем двух детей — $Sw(x,y)$ и $Sw/o(x,y)$. Но лучший вариант для следующей итерации выбираем только по оценке. Так давайте выбирать лучший не только по оценке, но и по глубине в дереве, т.к. чем глубже выбранный элемент, тем ближе он к концу подсчета. Тем самым мы сможем наконец дождаться ответа.

Первый и самый простой вариант эвристики $\text{cost} = \text{mark} - k * \text{depth}$. k подбираем в зависимости от среднего веса ребра так, чтобы все-таки глубина не играла определяющей роли. Так же добавим, например, что depth равно минимум b , т.е. если наша вершина находится на расстоянии q от корня, и $q < b$, то $\text{depth} = b$, иначе $\text{depth} = q$. В качестве b выбираем такое число, что до такой глубины честный алгоритм ветвей и границ (без эвристики) доходит за адекватное время. В моем случае было $b = 30$.

Запускаем, ждем.

За ночь работы получаем ответ, который выигрывает у жадного алгоритма ~2%.

Мало, очень мало, учитывая, что жадный алгоритм работает пару секунд и пишется за 5 минут.

И теперь победитель:

Жадный алгоритм с локальным методом границ и ветвей

Алгоритм такой:

1. Запускаем жадный алгоритм, получаем некоторый маршрут.
2. Делим маршрут на несколько частей.
3. В каждой части делаем фиктивное ребро — из последней вершины маршрута в первую, которое трогать запрещается
4. На каждой из этих частей запускаем метод ветвей и границ, без эвристики.
5. Объединяем части, оптимизированные методом ветвей и границ, размыкая фиктивные ребра и соединяя последнюю вершину $n-1$ части с первой n части.

Как легко понять, этот алгоритм имеет несколько плюсов.

- честный метод ветвей и границ, без использования эвристики;
- легко параллелится, выигрываем во времени работы;

14.11.2016Поиск гамильтонова цикла в большом графе (задача коммивояжера).Часть 1 / Хабрахабр

— жадный алгоритм говорит нам лишь части разбиений, после объединения мы используем только NUMBER_OF_PARTS-1 ребер, данных нам жадным алгоритмом, и не оптимизированных методом ветвей и границ.

Результат.
Время работы на 25 частях — 3 минуты, выигрываем у жадины ~7%
10 частей — 4 часа, выигрыш ~15%





продолжение


алгоритм, c++, гамильтонов цикл

↑ +25 ↓

43,5k

★ 175





Nikita Starichkov @demist

карма рейтинг

21,20,0

Похожие публикации

- +30

Домашний алгоритм разбиения на слова (с картинками)

7,1k

★ 76

💬 12
- +35

Решаем головоломки шаманов в World of Warcraft генетическим алгоритмом

28,5k

★ 90


💬 26
- +14

Поиск гамильтонова цикла в большом графе (задача коммивояжера).Часть 2

17,6k


★ 89

💬 3



ФРИЛАНСИМ

Территория IT-фриланса



Спецпроект

Самое читаемое				Разработка
Сейчас	Сутки	Неделя	Месяц	
+13	Одна простенькая задачка. Быстро, красиво или чисто?			
9,6k	★ 59	💬 27		
+19	Парадокс Rimworld: захватывающая сюжетом «песочница»			
2,3k	★ 18	💬 8		
+11	Объясняем бабушке: Как зашифроваться за час			
4,3k	★ 60	💬 21		
+25	Log in или Log on? Front-end или Frontend? Продолжаем разбираться			
2,3k	★ 25	💬 16		

+10 Как я добавил 30000 человек в первый круг контактов, а эту соцсеть заблокируют в РФ

👁 2k ⭐ 11 💬 6

Комментарии (20)



datacompboy 24 ноября 2012 в 19:05 #

0 ↑ ↓

И всё это не факт что будет лучше для другого такого графа.
Или граф константа, и цель — найти оптимальный только в нём и только один раз?



demist 24 ноября 2012 в 19:13 # h ↑

0 ↑ ↓

задача стояла конкретно для одного графа.
понятно, что на разных графах будут разные результаты, разный подбор эвристик, констант в алгоритмах и т.д.
в любом случае ни один из этих алгоритмов не дает верное решение. только близкое к нему. и на другом графе будут, возможно, другие результаты. однако жадина+метод ветвей и границ наиболее оптимален, т.к. никогда не будет худшим. а у простой жадина он точно всегда выиграет. единственное — когда-то может повезти, и чистый МВиГ посчитается быстро и выиграет.



datacompboy 24 ноября 2012 в 22:08 # h ↑

0 ↑ ↓

для одного графа можно генетику запустить, и ждать до получения идеального варианта...



demist 24 ноября 2012 в 22:10 (комментарий был изменён) # h ↑

0 ↑ ↓

можно, да, но дэдлайн был близок и время было ограничено. лично мне еще понравился муравьиный алгоритм, но кодить его тоже времени не было



pasha_golub 24 ноября 2012 в 20:10 #

+2 ↑ ↓

Все хорошо! Но надо бы вначале напоминать, что есть Гамильтонов цикл. А то создается впечатление...



yeputons 24 ноября 2012 в 20:21 # h ↑

+1 ↑ ↓

Думал аналогично, пока не увидел, что граф полный.
Возможно, было бы лучше переименовать в «Задача коммивояжера».



demist 24 ноября 2012 в 20:23 # h ↑

+1 ↑ ↓

учел вашу критику =)



mephistopheies 24 ноября 2012 в 22:32 #

+1 ↑ ↓

во время учебы как то занимался анализом алгоритмов для решения задач комбинаторной оптимизации, как раз на примере задачи коммивояжера. лет 5 назад у меня в топ-3 были: 1 — муравьиный алгоритм, 2 — генетический алгоритм, 3 — нейросеть Хопфилда. сейчас на сколько я помню лидирует генетический алгоритм.



demist 24 ноября 2012 в 22:52 # h ↑

0 ↑ ↓

я недавно находил, что все-таки муравьиная колония выигрывает сейчас



mephistopheies 24 ноября 2012 в 23:18 # h ↑

0 ↑ ↓

возможно ситуация опять поменялась, я сейчас не слежу за этой задачей



halyavin 24 ноября 2012 в 22:57 #

+2 ↑ ↓

А литературу вы читали? Наиболее эффективный алгоритм решения TSP — через линейное программирование. Есть готовые программы, которые подобные алгоритмы реализуют: [concorde](#). Современный рекорд порядка 100 000 вершин. Думаю с 500 вершинами проблем не будет.



demist 24 ноября 2012 в 23:13 # h ↑

0 ↑ ↓

задача была не получить ответ, а находить алгоритм. я думаю, за три дня я бы не находит то, что установило рекорд



halyavin 25 ноября 2012 в 09:24 # h ↑

+1 ↑ ↓

А чем чужой код для вас хуже? Как минимум его можно использовать, чтобы получить точное решение, с которым можно сравнивать ваши результаты.




demist 25 ноября 2012 в 20:37 (комментарий был изменён) # h ↑

0 ↑ ↓

говорю же-задача стояла накопить.
для оценки результатов использовался результат жадина.
идея использовать оптимальное решение, данное линейным алгоритмом хороша, но имеет, на мой взгляд один минус- мои алгоритмы сильно зависят от графа, и поэтому на разных графах будут показывать разные результаты относительно оптимального

решения, но вот жадина-то тоже зависит от графа, поэтому относительно жадина мне показалось рассматривать как-то логичнее, проще анализировать

 **Diversus** 25 ноября 2012 в 00:27 #

0 ↑ ↓

Я когда то решал похожую задачу [генетическим алгоритмом](#).

Было бы неплохо также его упомянуть. Показывает достойный результат на больших графах. Единственный его минус — это то, что может быть получено не оптимальное решение, а близкое к оптимальному.

Вот [пример](#) моего дипломника по этой теме.

Векторный редактор с картой города и поиск в транспортной сети. Тема очень близкая к данной.

Распакуйте архив в векторном редакторе, выберите уже готовую схему города с точками поиска и нажмите на кнопку «Поиск».

 **demist** 25 ноября 2012 в 00:36 # h ↑

0 ↑ ↓

круто, посмотрю утром, когда будет время

 **voidSoul** 26 июня 2015 в 19:25 # h ↑

0 ↑ ↓

А подскажите каким образом Вы кодировали/декодировали в хромосому путь обхода городов? Как из ноликов и единичек получить путь?

 **dima_mendeleev** 25 ноября 2012 в 00:59 #

+2 ↑ ↓

> Жадина хорошо работает на случайном графе, однако тут мы ничего про граф не знаем
Нематематик от такого впадет в ступор :-D


 **lany** 25 ноября 2012 в 16:43 #

+1 ↑ ↓

Задача коммивояжера, она же поиск гамильтонова цикла минимальной суммарной стоимости — NP-полная

Вот интересно, понимаете ли вы вообще, что такое NP-полная задача, когда употребляете этот термин? Вы, возможно, думаете, что это просто задача, решение которой недостижимо за полиномиальное время от некоторого параметра задачи? Спешу вас уверить, вы ошибаетесь. И задача коммивояжера вовсе не NP-полная, а NP-трудная. Ознакомьтесь, например:

www.nomachetejuggling.com/2012/09/14/traveling-salesman-the-most-misunderstood-problem/

 **demist** 25 ноября 2012 в 17:01 # h ↑

+2 ↑ ↓

да, согласен, в первоначальном варианте ошибка, верно — NP-тяжелая или NP-трудная. спасибо за указание

Только зарегистрированные пользователи могут оставлять комментарии. [Войдите](#), пожалуйста.

Интересные публикации



 **Новые-старые форматы: HD-винил и DIY-пластинки** 1

 **Делаем стартап просто и технологично. Маячки Eddystone** 0

 **Производительность межпроцессного обмена сообщениями в node.js** 0

 **SMART TV – будущее телевидения** 5

 **Выпуск Rust 1.13** 4