AlexGl 1 июня 2011 в 03:29

Задача о вершинном покрытии

Алгоритмы

Из песочницы

Введение.

На данный момент не известно полиномиальных по времени алгоритмов точного решения NP-трудных задач. Более того, специалисты по теории сложности склоняются к варианту, что таких алгоритмов не существует. Однако, NP-трудные задачи часто встречаются в жизни. Одним из способов борьбы с NP-трудными задачами на практике является применение приближенных алгоритмов.

Рассмотрим лучший известный приближенный алгоритм решения **задачи о вершинном покрытии**.

Постановка задачи.

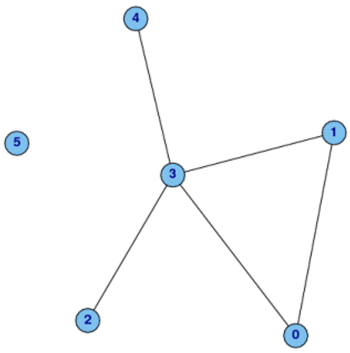
Вершинное покрытие – это такое множество **S** вершин графа, что у каждого ребра графа хотя бы один конец входит в **S**.

Задача заключается в том, чтобы выбрать в неориентированном графе минимальное (по количеству вершин) вершинное покрытие.

Цель.

Построим простой алгоритм решения этой задачи, ошибающийся не более чем в два раза. Это означает, что если есть оптимальное решение — множество вершин T, то полученное нами решение S удовлетворяет неравенству **|S| <= 2|T|**.

Пример.



В приведенном примере вершинным покрытием является, например, множество вершин **{0, 1, 2, 4}**. Все шесть вершин графа также образуют вершинное покрытие. Однако, минимальным вершинным покрытием в рассматриваемом графе будет множество, состоящее всего из двух вершин. Например, это могут быть вершины с номерами 1 и 3. Действительно, все ребра графа покрываются двумя этими вершинами.

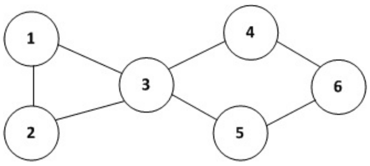
Алгоритм.

На каждом шаге алгоритма производим следующие действия:

- Выбираем случайное ребро графа **e=(u, v)**.
- Добавляем в решение S обе выбранные вершины **u** и **v**.
- Удаляем из графа все ребра, инцидентные вершинам **u** или **v**.

Повторяем этот шаг до тех пор, пока не удалим все ребра графа.

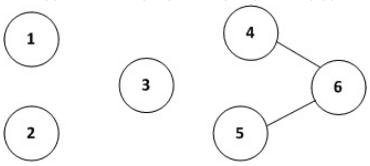
Пример работы алгоритма.



В приведенном примере минимальное вершинное покрытие содержит три вершины. Например, это могут быть вершины **1, 3** и **6**. Рассмотрим работу нашего алгоритма на приведенном примере.

Первая итерация:

- Выбираем случайное ребро. Например, ребро **(1, 3)**.
- Добавляем в решение **S** обе вершины выбранного ребра: **S={1, 3}**.
- Удаляем из графа все ребра, инцидентные вершинам **1** или **3**.



Вторая итерация:

- Выбираем случайное ребро. Пусть это будет ребро **(4, 6)**.
- Добавляем в решение **S** обе вершины выбранного ребра: **S={1, 3, 4, 6}**.
- Удаляем из графа все ребра, инцидентные вершинам **4** или **6**.

В графе не осталось ребер. Следовательно, результатом работы нашего алгоритма будет вершинное покрытие **S={1, 3, 4, 6}**.

Доказательство.

Докажем, что построенное множество является вершинным покрытием. Действительно, на каждом шаге мы удаляли лишь уже покрытые ребра. Мы повторяли эту итерацию, пока в графе еще оставались ребра. Таким образом, мы покрыли все ребра графа.

Докажем, что наш алгоритм ошибается не более, чем в 2 раза.

Все рассматриваемые алгоритмом ребра e не имеют общих вершин. Следовательно, из каждого из этих рёбер в оптимальное решение T должна входить хотя бы одна вершина. Это значит, что **2|T|>=|S|**.

Немного истории

В знаменитой работе Ричарда Карпа «Reducibility among combinatorial problems» под названием Node Cover рассматривается соответствующая Vertex Cover задача разрешимости и доказывается ее NP-полнота.
Рассмотренный нами алгоритм был независимо разработан Fanica Gavril и Mihalis Yannakakis в 1974 году.
Лучшая известная на сегодняшний день оценка приближенного алгоритма задачи Vertex Cover принадлежит George Karakostas. Оценка доказана в работе A better approximation ratio for the vertex cover problem.

Заключение.

На данный момент не известно полиномиальных приближенных алгоритмов для Vertex Cover значительно улучшающих полученную нами точность. То есть, не известно полиномиального по времени алгоритма, который бы ошибался не более, чем в 1.999 раза. Таким образом, мы получили простой полиномиальный по времени алгоритм с хорошей точностью для решения NP-трудной задачи.

Теги: vertex cover, approximation algorithm, NP-hard

Комментарии 27

Longedok 1 июня 2011 в 03:37 0

Однако, минимальным вершинным покрытием в рассматриваемом графе будет множество, состоящее всего из одной вершины: вершины с номером 3.
А как же ребро, соединяющее вершины 0 и 1? Оно ведь ни одним концом не входит в 3.

AlexGl 1 июня 2011 в 03:42 0

Да, конечно. Большое спасибо! Исправил.

alexeygrigorev 1 июня 2011 в 08:22 0

А вот скажите — а где задачи о вершинном покрытии, клике и независимом множестве применяются? Мне интересно, какие прикладные задачи с их помощью решаются. Я в свое время когда читал о них, даже не смог в контестерах задачи такие найти, чтобы проверить правильность реализации алгоритмов.

AlexGl 1 июня 2011 в 11:25 +3

Возможно, это связано с тем, что на контестах обычно требуются точные детерминированные решения. А для NP-трудных задач на данный момент известны только экспоненциальные по времени решения, что не нравится контестам. Применений у NP-трудных задач действительно много. Поэтому люди и ищут все более и более быстрые и изощрённые способы решения NP-трудных задач.
Например, Вы хотите расставить как можно меньше вышек мобильной связи, но хотите покрыть какой-то набор дорог. Это и есть задача о вершинном покрытии.
Если Вы хотите рассадить в своем зоопарке по клеткам бегемотов, то Вам необходимо решить задачу о независимом множестве. Ведь известно, что бегемот из первой клетки не даст спокойно жить бегемотам из нулевой и третьей клеток (т.к. в графе есть ребра 1-0 и 1-3) и.т.д. Так, Вам нужно выбрать максимальное количество клеток, попарно не соединенных ребрами. Это и есть задача о независимом множестве.

alexeygrigorev 1 июня 2011 в 11:28 0

Хорошие примеры, спасибо.

AlexGl 1 июня 2011 в 12:21 +1

Пример для клики не заметил. Если Вы хотите организовать самую большую вечеринку для «своих» (то есть, чтобы на вечеринке каждый знал каждого), то Вам необходимо решить задачу о максимальной клике. Обозначьте друзей вершинами, проведите ребро между двумя вершинами, если два этих друга знакомы и решите задачу о максимальной клике.

alexeygrigorev 1 июня 2011 в 08:24 0

Кстати, на английском это не NP-hard, а NP-complete

alexeygrigorev 1 июня 2011 в 08:27 0

Однако, извиняюсь — в википедии есть статья NP-hard. Как и NP-complete. Это вроде бы даже разные вещи? Не понял пока. Не расскажите?

mrShadow 1 июня 2011 в 09:24 +1

По-моему, NP-complete — это NP-hard, для которой известно, что к ней сводятся другие NP-задачи. То есть, доказательство того, что для некоторой NP-complete задачи есть полиномиальный алгоритм, приводит к P = NP.

AlexGl 1 июня 2011 в 10:38 +2

Извините, я тут не очень пока ориентируюсь. Ответил вот тут: habrahabr.ru/blogs/algorithm/120328/#comment_3945684

AlexGl 1 июня 2011 в 10:37 +8

alexeygrigorev: Да, конечно расскажу.
mrShadow: Немного уточню.

NP-hard (NP-трудная, NP-сложная) задача — это задача, мгновенно получая ответы на которую, мы смогли бы за полиномиальное время решить любую задачу из класса NP.
Или, что тоже самое, смогли бы решить какую-нибудь NP-полную задачу.
Неформально это значит, что NP-трудная задача не проще, чем все задачи из класса NP. Например, если бы NP-трудная задача решилась за полиномиальное время, то получилось бы, что какая-то NP-полная задача тоже решается за полиномиальное время, то есть P=NP.

NP-complete (NP-полная) задача — это NP-трудная задача, которая сама к тому же принадлежит классу NP.
В классе NP лежат задачи разрешимости (decision problem). Это задачи, на которые есть ответ да/нет. Например, к NP (и NP-полным) задачам относится следующая версия задачи о вершинном покрытии: «Существует ли в графе вершинное покрытие мощности не более k?». Это вопрос на да/нет.
Также бывают задачи оптимизации (NP optimization problem). Это NP-задачи, в которых требуется найти лучшее решение. Это уже не задачи на да/нет. Следовательно, они не лежат в классе NP. Но многие из них являются NP-трудными. Например, NP-трудной является задача из топика: «Найти минимальное вершинное покрытие».

Умея решать NP-сложную задачу (например, задачу из топика) можно быстро решить NP-полную задачу (например, задачу о существовании в графе покрытия <=k). Действительно, пусть мы нашли минимальное вершинное покрытие. Его размер m. Если m<=k, то в графе существует вершинное покрытие мощности <=k, иначе — не существует.

НЛО прилетело и опубликовало эту надпись здесь

НЛО прилетело и опубликовало эту надпись здесь

AlexGl 1 июня 2011 в 12:10 +1

Да, это синонимы.
Эта задача как раз не из NP. Классу NP принадлежат задачи разрешимости. Это задача оптимизации.
Кроме того NP-трудная отличается от NP тем, что она не легче любой задачи из NP. То есть, она трудная задача. Например, проверить, является ли одно число на 1 больше другого — это тоже задача из NP. Но она не NP-трудная.

НЛО прилетело и опубликовало эту надпись здесь

AlexGl 1 июня 2011 в 14:33 0

Рад, что Вы разобрались.
Но то, что эта задача не из P и не из NP понятно, т.к. в этих классах лежат только задачи разрешимости (т.е. задачи на да/нет).

НЛО прилетело и опубликовало эту надпись здесь

AlexGl 1 июня 2011 в 18:00 0

Любая оптимизационная задача из NP сводится не к некоторому языку из NP.
Тут, видимо, я не очень понятно объяснил именно формальное отличие. То есть, P и NP — это классы (множества) языков. Язык — это просто набор слов (строк, чисел, графов, пар...). Так вот язык, который состоит из графов, имеющих вершинное покрытие не больше k — NP-полный язык. А задача определения вот такого вот k по графу — это не язык. Это задача, которая также трудна, как и все языки из NP. Поэтому задача и называется NP-трудной.

palebluedot 1 июня 2011 в 13:22 0

Спасибо за статью.

AlexTikhonov 1 июня 2011 в 13:55 0

По поводу алгоритма: «Выбираем случайное ребро графа e=(u, v).». А не должно ли случайно быть доволнительно условие, что u и v не должны входить в S?

AlexGl 1 июня 2011 в 14:36 **0**

Возможно, я не очень подробно это описал. Но на каждой итерации алгоритма мы удаляем все ребра, инцидентные выбранным вершинам. Это значит, что мы никогда больше не сможем выбрать ребро, инцидентное вершинам из S.

AlexTikhonov 1 июня 2011 в 15:08 **+1**

гм. Теперь ясно. Просто меня немного смутили термины «ребро графа» и «множества ребер» — показалось что изменяется некое множество ребер (изначально состоящее из ребер графа), а граф оставался как есть.

Хотя в примерах итераций действительно третий пункт говорит уже не о каком-то множестве, а о графе.

В остальном же вполне понятный алгоритм и хорошая статья :)

Разве что я бы в конце еще добавил информацию о том, кто впервые описал (если не врет вики и я не ошибаюсь, то Fanica Gavril и Mihalis Yannakakis). Ну и про «1.999» — если не ошибаюсь, так там идет речь об $2-\Theta(\frac{1}{\sqrt{log_2 n}})$ — так выйдет чуток поточнее. *(фух, давно я в латехном формате не набирал формулы :))*

AlexGl 1 июня 2011 в 19:53 **0**

Спасибо за замечания! Вроде бы, все исправил и дополнил.

alex103 30 июля 2014 в 09:06 **0**

А вы используете алгоритм в какой то практической задаче, или просто ради научного интереса??

Что нибудь можете сказать по поводу habrahabr.ru/post/225831/?

AlexGl 30 июля 2014 в 10:48 **0**

> А вы используете алгоритм в какой то практической задаче, или просто ради научного интереса??

Нет, я не использую этот алгоритм на практике, но уверен, что он имеет множество практических и теоретических применений.

> Что нибудь можете сказать по поводу habrahabr.ru/post/225831/?

Полиномиальное решение задачи о покрытии множества влекло бы равенство P и NP. К сожалению, утверждение теоремы 2 не верно в общем случае. В теореме один Вы описываете некоторые из минимальных покрывающих множеств, но не все. В теореме два Вы говорите, что если выбрать наименьшее из описанных Вами минимальных множеств, то Вы получите наименьшее покрывающее множество. Это было бы верно, если бы Вы рассматривали множество всех минимальных покрытий, однако Вы рассматриваете лишь его подмножество.
То есть, в теореме 1 каждое Ваше решение соответствует минимального множеству, но обратное утверждение не верно. В теореме 2 Вы как раз пользуетесь обратным утверждением: Вы подразумеваете, что каждое минимальное покрытие описывается в теореме 1.

alex103 30 июля 2014 в 12:47 **0**

Имеет ли смысл прислать вам доказательства теорем?
Ещё раз подчеркну, что это не мой труд, а моего, в прошлом, руководителя(к сожалению уже ушедшего от нас).
Сам я не могу обнаружить ошибки в доказательствах(вернее не ошибки, а скорее «неполность»).
Если у вас, конечно, есть на это время.

AlexGl 30 июля 2014 в 15:03 **0**

Пришлите, если Вам это интересно, да.

САМОЕ ЧИТАЕМОЕ

Сутки Неделя Месяц