NP-полные задачи

Классы задач

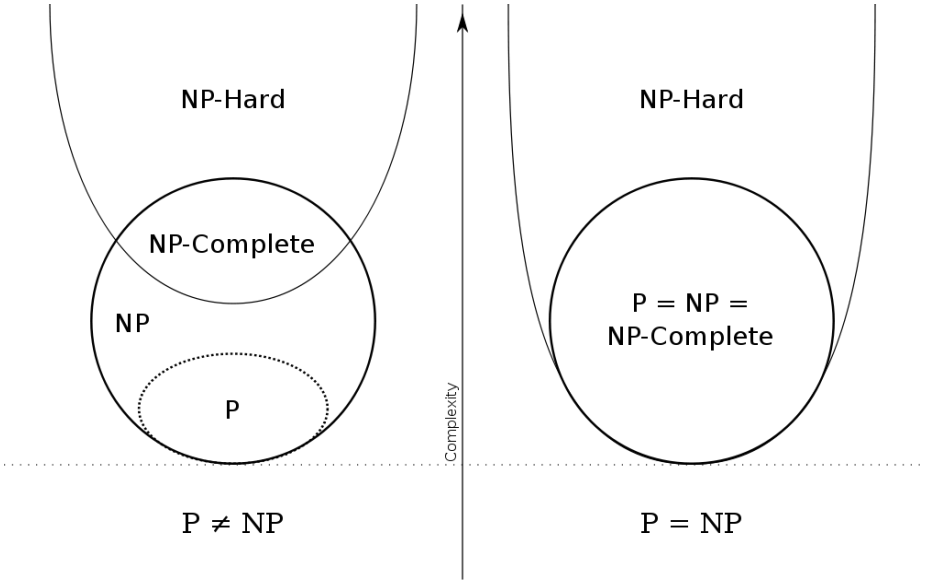
Класс P - задачи, разрешимые в течение полиномиального времени $O(n^k)$.

Класс NP - задачи, проверяемые в течение полиномиального времени: если дано решение, его корректность можно проверить за полиномиальное время. Могут быть решены за полиномиальное время на недетерминированной вычислительной машине. Все P-задачи также являются NP-задачами.

Класс NP-полных задач (NP-complete) - задача принадлежит классу NP-полных задач, если она принадлежит классу NP и является столь же “сложной”, как и самая сложная задача из NP. Все задачи из NP сводятся к любой NP-полной задаче (за полиномиальное время).

Для NP-полных задач пока не известно эффективных алгоритмов решения за полиномиальное время на детерминированной вычислительной машине. Если удастся найти алгорим решения одной NP-полной задачи за полиномиальное время на детерминированной машине, то и все NP-задачи также можно будет решить за полиномиальное время на детерминированной машине (т.е. P=NP, что пока не доказано).

Класс NP-трудных задач (NP-hard) - задача является NP-трудной, если к ней сводится некоторая NP-полная задача (т.е. NP-трудная задача не менее трудна, чем любая NP-полная задача). NP-трудная задача не обязательно входит в класс NP (т.е. может не решаться за полиномиальное время даже на недетерминированной вычислительной машине).



Задачи принятия решения и оптимизации

Задача оптимизации - требуется найти самое лучшее (оптимальное, по некоторому критерию) решение из множества допустимых решений.
Задача принятия решения (распознавания свойств) - может иметь ответом только “да” или “нет”.

Теория NP-полных задач строится только для задач принятия решения (т.е. при доказательстве NP-полноты ограничиваются только задачами принятия решения). Тем не менее, каждую задачу оптимизации можно свести к задаче принятия решения путем наложения ограничения на решение: например, если поиск кратчайшего пути в графе между заданной парой вершин - задача оптимизации, то соответствующей задачей принятия решения будет “есть ли путь в графе между заданной парой вершин, состоящий не более чем из k ребер (k - заданная константа)? ”. Если задача оптимизации “простая” (решается за полиномиальное время), то “простой” будет и задача принятия решения, аналогично, если задача принятия решения “сложная” (не решается за полиномиальное время), то “сложной” будет и соответствующая ей задача оптимизации. Таким образом, теория NP-полноты имеет следствия и для задач оптимизации

Доказательство NP-полноты
Пусть A и B - задачи принятия решения. Если есть алгоритм, который за полиномиальное время преобразует входные данные задачи A во входные данные задачи B, причем ответом на задачу A будет “да” тогда и только тогда, когда “да” будет ответом на задачу B, то назовем его алгоритмом приведения с полиномиальным временем.

Если для задачи B известен алгоритм решения за полиномиальное время, то задачу A также можно решить за полиномиальное время путем приведения ее входных данных (за полиномиальное время) к входным данным задачи B и решения задачи B. Аналогично,

если для задачи A нет алгоритма решения за полиномиальное время, то чтобы показать его отсутствие и для задачи B, нужно найти алгоритм приведения с полиномиальным временем задачи A к задаче B.

Таким образом, чтобы доказать NP-полноту задачи, нужно показать, что она принадлежит классу NP и что к ней сводится (приводится) некоторая известная NP-полная задача. В роли такой “изначальной” NP-полной задачи обычно выступает SAT или 3-CNF-SAT, NP-полнота которых была доказана.

SAT

Задача SAT (Boolean Satisfiability Problem) - задача о выполнимости булевой формулы. Пусть дана булева формула от n переменных $f(x_1, x_2, \ldots, x_n)$, т.е. выражение, построенное из вхождений n переменных, операций отрицания (НЕ), коньюнкции (И), дизъюнкции (ИЛИ) и скобок. Требуется проверить, является ли формула выполнимой, т.е. существует ли такой набор значений переменных $y_1, y_2, \ldots, y_n$, что $f(y_1, y_2, \ldots, y_n) = \text{true}$.

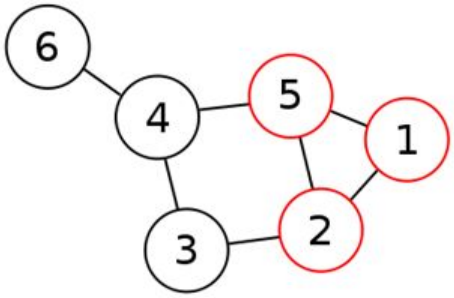
Задача SAT для произвольной формулы в конъюнктивной нормальной форме (CNF) является NP-полной (время решения - $O(n*2^n)$). Тем не менее, есть виды формул (например, 2-CNF), для которых задача SAT не является NP-полной.

Задача 3-CNF-SAT - задача SAT, когда формула находится в 3-коньюнктивной нормальной форме: каждая дизъюнкция содержит ровно 3 литерала - переменную или ее отрицание. Также является NP-полной.

Примеры NP-полных задач

Задача о клике

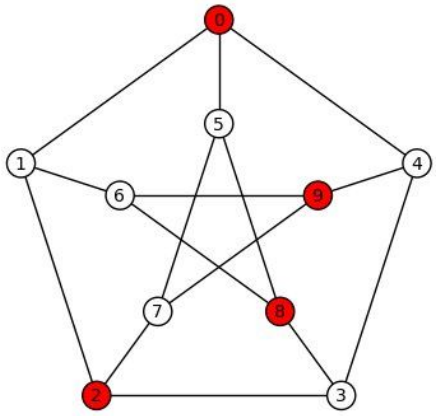
Клика в неориентированном графе (clique) - полный подграф графа (т.е. любая пара вершин которого соединена ребром). Нужно найти клику максимального размера (maximum clique; клику с максимальным числом вершин).



Задача о независимом множестве

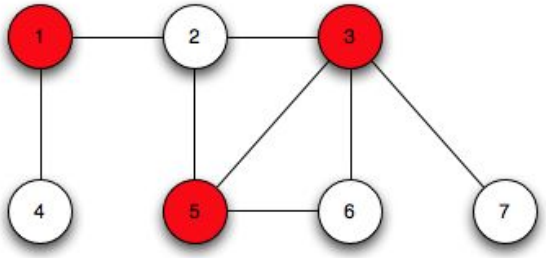
Независимое множество (independent set) - подграф графа, каждая пара вершин которого **не** соединена ребром. Нужно найти независимое множество максимального размера.

Задача связана с задачей о клике, т.к. максимальное независимое множество графа будет максимальной кликой в дополнении этого графа и наоборот.



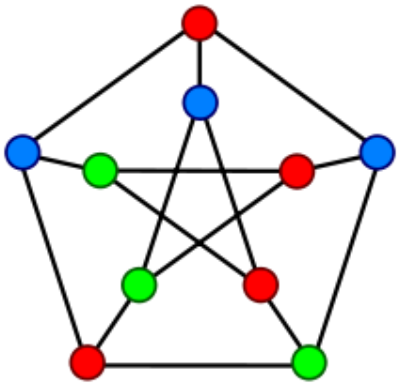
Задача о вершинном покрытии

Вершинное покрытие графа (vertex cover) - подножество вершин графа, каждая вершина которого инцидентна минимум одному ребру графа (нет ребер, не инцидентных хоть одной вершине из покрытия). Требуется найти вершинное покрытие минимального размера. Дополнение минимального вершинного покрытия - максимальное независимое множество.



Задача о раскраске графа

Раскраска графа (graph coloring) - присвоение вершинам графа меток (цветов), чтобы для каждого ребра его вершины были окрашены в разные цвета. Требуется найти раскраску, использующую минимальное число цветов (хроматическое число графа).



Задача о коммивояжере

Дан полный взвешенный граф (соответствует транспортной сети с известной “стоимостью” пути между каждой парой городов). Гамильтонов цикл - путь в графе, начинающийся и заканчивающийся в одной вершине и проходящий через остальные вершины ровно по одному разу. Требуется найти гамильтонов цикл с минимальным суммарным весом.

Задача о рюкзаке

Дано N разных предметов, каждый предмет имеет вес/объем и стоимость. Также дан рюкзак, имеющий верхний предел по максимальному весу/объему входящих в него предметов. Требуется найти такое подмножество предметов, чтобы оно входило в рюкзак и суммарная стоимость предметов была максимальной. Есть различные виды задач о рюкзаке, отличающиеся ограничениями на выбор предметов.

Алгоритмы решения NP-полных задач

Алгоритм точного решения NP-полной задачи (полный перебор) имеет экспоненциальную временную сложность и не эффективен для больших размеров задачи.

Методы решения задач, использующие полный перебор в той или иной мере:

- поиск с возвратом
- метод ветвей и границ
- альфа-бета отсечение

Если точное (оптимальное) решение не требуется, можно использовать различные приближенные алгоритмы, позволяющие найти субоптимальное (приближенное, т.е. возможно хуже оптимального, иногда в известное число раз) решение, но уже за полиномиальное время.

В число приближенных алгоритмов и методов для их построения входят:

- жадные алгоритмы
- динамическое программирование
- линейное программирование

- алгоритмы локального поиска
- эвристические алгоритмы
- рандомизированные алгоритмы
- генетические алгоритмы
- и др.

Метод ветвей и границ

Пусть X - множество возможных решений задачи (пространство поиска), f(x) - оценочная функция; нужно найти решение $y \in X$, $f(y) = \min \{f(x) \mid x \in X\}$.

Считаем, что есть эффективая процедура определения нижней оценки для подножества решений S: $bound(S) \le f(x) \ \forall \ x \in S$. Тогда для поиска оптимального решения методом ветвей и границ множество X рекурсивно разбивается на подмножества (стадия ветвления) для построения дерева поиска. Если для какой-либо ветви дерева S bound(S) > upper, где upper - текущая верхняя оценка оптимального решения, то такая ветвь может быть исключена из рассмотрения, т.к. не сможет дасть решения лучше чем текущее (стадия отсечения). После анализа всего дерева будет получено единственное оптимальное решение.

Преимущество в скорости метода ветвей и границ зависит от эффективности вычисления нижней оценки bound(S) для ветви дерева без посещения всех узлов этой ветви. При посещении всех узлов ветви метод сводится к полному перебору.

```
BranchAndBound(X):
      upper = Infinity
      solution = None
      BranchAndBound(X, upper, solution)
      return solution

BranchAndBound(x, upper, solution):
      if x is single element:
            if f(x) < upper:
                  upper = f(x) // update global upper
                  solution = x // update global solution
      else:
            subsets = subdivide(x)
            for each s in subsets:
                  if bound(s) <= upper: // branch isn't discarded
                        BranchAndBound(s, upper, solution)
```

Алгоритм Брона-Кербоша

Алгоритм Брона-Кербоша - метод ветвей и границ для поиска всех максимальных клик в графе (т.н. maximal clique - т.е. таких клик, которые не входят в другие клики большего размера; не путать с maximum clique!). Алгоритм оперирует с тремя множествами вершин: R - возможная клика, P - вершины-кандидаты на включение в клику, X - уже рассмотренные вершины (используется для отсечения повторяющихся или плохих вариантов).

```
BronKerbosch(G):
      BronKerbosch({}, G.V, {})

BronnKerbosch(R, P, X):
      if P and X is empty:
            R is a maximal clique, report it
      else:
            for each v in P:
                  BronKerbosch(R ∪ {v}, P ∩ Neigh(v), X ∩ Neigh(v))
                  P = P \ {v}
                  X = X ∪ {v}

Neigh(v):
      return all vertices adjacent with vertex v
```

Домашнее задание

Для указанной задачи разработать (либо изучить имеющиеся и запрограммировать) два алгоритма решения: находящий точное решение (с экспоненциальным временем работы) и находящий приближенное решение (с полиномиальным временем работы). Сравнить время работы алгоритмов и полученные решения на наборах входных данных как небольшого, так и большого размера (входные данные сгенерировать или достать самостоятельно). Если не хватает времени дождаться получения точного решения, привести приблизительную оценку возможного времени работы и указать, что решение не получено. Язык программирования по выбору.

Варианты заданий

Номер варианта для выполнения = свой номер в списке группы % количество вариантов

1. Задача о клике (maximum clique)
2. Задача о независимом множестве (maximum independent set)

3. Задача о вершинном покрытии (minimum vertex cover)
4. Задача о раскраске графа (chromatic number)
5. Задача о коммивояжере (travelling salesman)
6. Задача о рюкзаке 0-1 (0-1 knapsack - каждый предмет может быть выбран максимум 1 раз)
7. Задача о покрытии множества (set cover)
8. Задача о сумме подмножества (subset sum)
9. Ограниченная задача о рюкзаке (bounded knapsack - каждый предмет может быть выбран не больше фиксированного числа раз)
10. Взвешенная задача о покрытии множества (weighted set cover)
11. Задача об упаковке в контейнеры (linear bin packing - упаковка в одномерные одинаковые контейнеры)
12. Неограниченная задача о рюкзаке (unbounded knapsack - каждый предмет может быть выбран неограниченное число раз)

Контрольные вопросы

1. Классы задач. NP-полные задачи.
2. Задачи принятия решения и оптимизации. Примеры.
3. Приводимость задач и доказательство NP-полноты.
4. Задача SAT.
5. Задача о клике.
6. Задача о независимом множестве.
7. Задача о вершинном покрытии.
8. Задача о раскраске графа.
9. Задача о коммивояжере.
10. Задача о рюкзаке.
11. Приближенные алгоритмы решения NP-полных задач.
12. Метод ветвей и границ.
13. Алгоритм Брона-Кербоша.

Литература

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: Построение и анализ, 3-е издание:
    - Глава 34, NP-полнота
    - Глава 35, Приближенные алгоритмы
2. Ахо А., Хопкрофт Д., Ульман Д. Структуры данных и алгоритмы:
    - Глава 10, Методы разработки алгоритмов
3. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи
    - Главы 1-7