Дискретная оптимизация
весна 2013

Александр Дайняк

http://www.dainiak.com

Организационные вопросы

- Вся информация на странице http://www.dainiak.com/ru/teaching/courses/current/discopt_s2013
- Две возможности получить оценку по курсу:
  - Сдавать домашние задания и подтвердить их аутентичность в конце семестра
  - Сдать курс Linear and Discrete Optimization на Coursera и подтвердить свои знания в конце семестра (курс начался 18 февраля и самое время записаться!)
- Запись на курс *обязательна* для получения оценки в конце семестра, записаться можно до 1 марта: http://goo.gl/keRxR
- По вопросам, связанным с курсом, можно писать по адресу dainiak+discopt@gmail.com, соблюдая правила, описанные здесь: http://www.dainiak.com/ru/teaching/misc/email

Организационные вопросы

- Курс будет содержать 8-10 лекций
- Полноценные слайды могут быть не ко всем лекциям
- Если сдавать курс Д.О.: экзамена нет, оценка выставляется по домашним заданиям.
  Потребуется знание Python 3.x и, возможно, LaTeX.
- Если сдавать L.&D.O. на Coursera: я не даю никаких дополнительных заданий, но в конце курса L.&D.O. (середина апреля) провожу опрос, чтобы Вы могли подтвердить авторство своих достижений.
  (При этом на Coursera есть свои тесты и домашние задания.)

Организационные вопросы

Плюсы в том, чтобы сдавать L.&D.O.:

- Повод подучить английский (математический).
- Если понравится, можно пройти ещё интересные курсы, например, курсы по алгоритмам Стэнфордского или Принстонского университетов. При этом опыт уже будет.
- Профиль на Coursera можно сделать общедоступным и указывать ссылку на него в своём резюме.

Минусы:

- Английский *нужно* знать на базовом уровне, достаточном, чтобы успевать за процессом.
- Основной упор в курсе L.&D.O. на линейную оптимизацию, поэтому не рассматриваются некоторые темы, вошедшие в курс Д.О.

Общая постановка задачи

Дано:

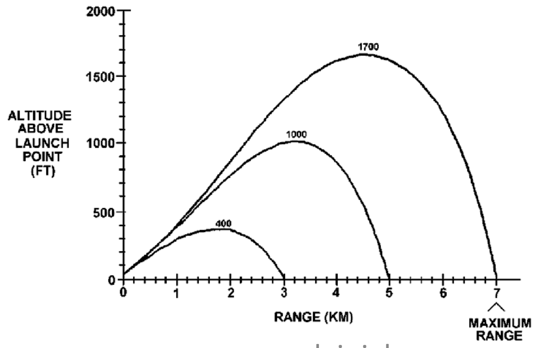
- S — заданное множество (область поиска)
- f — функция на множестве S

Задача:

- найти $x \in S$, такой, что $f(x) > f(y) \ \ \forall y \in S$

Непрерывная оптимизация

- Обычно:
  - S — числовое множество, метрическое пространство, …
  - f — выпуклая/непрерывная/гладкая функция
- Выгодные особенности:
  - S и f часто задаются явно
  - По множеству S можно «непрерывно перемещаться»



Дискретная оптимизация

- Особенности:
  - Множество S конечно или счётно, либо может быть сужено до конечного
  - Функция f, как правило, задана неявно
- Примеры:
  - Транспортные задачи
  - Задачи обхода и другие задачи на графах
  - Задачи теории расписаний

Транспортная задача

- Требуется перевезти T единиц товара с m складов в n магазинов
- Количество товара на i-м складе равно a_i
- Количество, требующееся в j-м магазине, равно b_j

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = T$$

- Стоимость перевозки с i-го склада в j-й магазин равна c_{ij}
- Задача: найти x_{ij} — количество, которое надо перевозить с i-го склада в j-й магазин, минимизировав при этом сумму

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \to min$$

- Пришли к задаче линейного программирования!

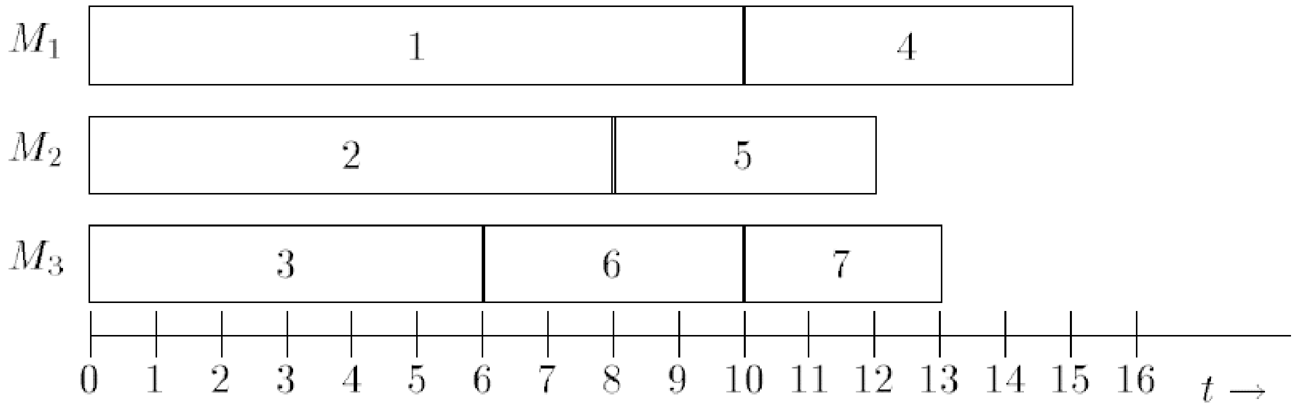
Транспортная задача

- $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = T$
- $\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \to min$

- Если a_i, b_j, c_{ij} целые, то оптимальное решение задачи тоже можно искать среди целочисленных, например, симплекс-методом. В данной задаче ограничение на целочисленность решения не создаёт дополнительных трудностей.

Задачи построения оптимальных расписаний

- Имеется несколько станков/рабочих/компьютеров M_j
- Имеется набор деталей/задач/программ, требующих обработки
- Требуется: распределить задачи по рабочим так, чтобы время окончания обработки последней задачи было минимальным



Задача о назначениях

- В страховой компании работают n агентов, продающих n разных типов услуг. Эффективность i-го агента при продаже услуг j-го типа равна c_{ij}. Сил каждого агента хватает только на продажу одного типа услуг.
- Продажу какого типа p_i нужно назначить i-му агенту, чтобы максимизировать суммарную эффективность $\sum_i c_{ip_i}$?

Задача о назначениях

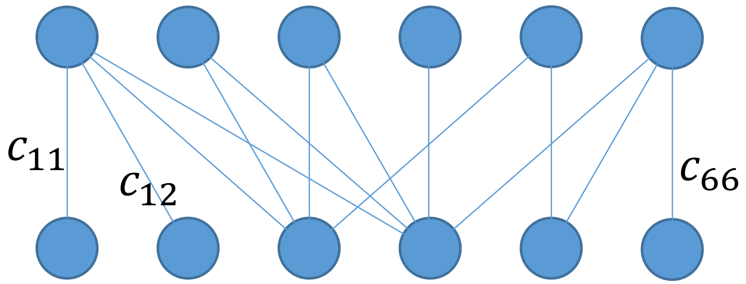
$$\sum_i c_{ip_i} \to max$$

- Можно переформулировать задачу в виде задачи ЛП:
  - x_{ij} — кодировка того, что кому назначено: $x_{ij} = 1$, если продажа j-го типа услуг назначена i-му агенту. В противном случае $x_{ij} = 0$.
  - Тогда надо подобрать x_{ij} так, чтобы
    $$\sum_{i,j} c_{ij} x_{ij} \to max$$
    при ограничении $\sum_i x_{ij} = \sum_j x_{ij} = 1$

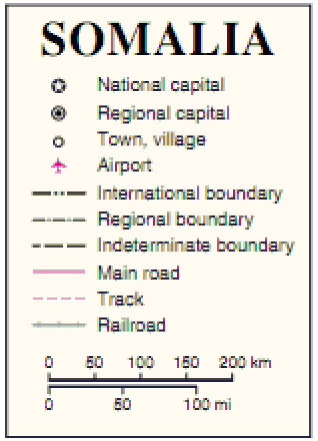
Задача о назначениях

$$\sum_i c_{ip_i} \to max$$

- Можно переформулировать в терминах теории графов: это задача о построении совершенного паросочетания максимального веса



Задача коммивояжёра (Travelling Salesman Problem, TSP)



Коммивояжёр Бендер продаёт в Сомали журнал «Пиратский вестник». Стартуя из Могадишо, ему требуется заехать в каждый населённый пункт и вернуться обратно в столицу, минимизировав суммарные издержки на перемещение.

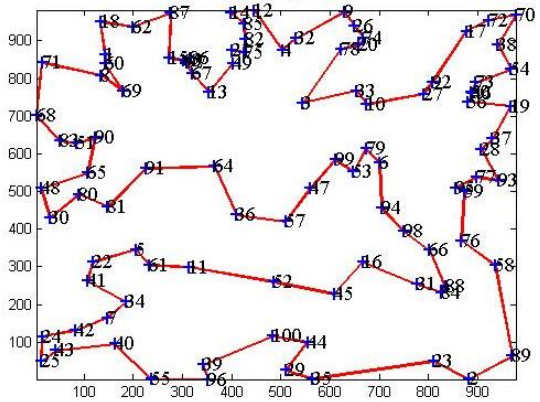
Поезд, самолёт, автобус: масса возможностей

Задача коммивояжёра

Более инженерное применение: построить траекторию сверла для фрезерного станка, высверливающего отверстия на плате:

Задача коммивояжёра

- В терминах теории графов: найти во взвешенном графе гамильтонов цикл минимального веса
- Можно предполагать, что граф полный (если ребра не было, добавим его, положив его вес равным ∞)

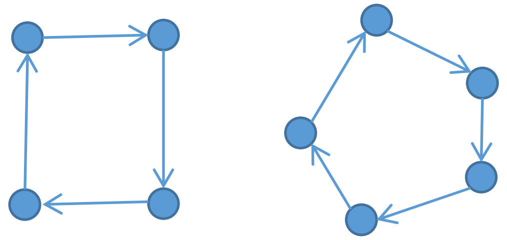


Задача коммивояжёра

- Попробуем переформулировать в терминах ЛП:
  - $0, \dots, n$ — номера вершин в графе,
  - c_{ij} — стоимость пути из i-й вершины в j-ю
  - $x_{ij} \in \{0,1\}$ — индикатор того, что есть дуга из i-й вершины в j-ю
  - Минимизируем $\sum_{ij} c_{ij} x_{ij}$ при ограничениях:
  - $\sum_i x_{ij} = \sum_j x_{ij} = 1$ — из каждой вершины выходит и в каждую вершину входит ровно одна дуга

Задача коммивояжёра

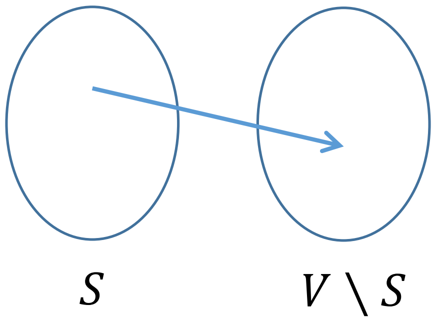
- Минимизируем $\sum_{ij} c_{ij} x_{ij}$ при ограничениях $\sum_i x_{ij} = \sum_j x_{ij} = 1$
- Проблема, как избежать такого:



- Формально условия регулярности соблюдены, но граф получился несвязным

Задача коммивояжёра

- Минимизируем $\sum_{ij} c_{ij} x_{ij}$ при ограничениях $\sum_i x_{ij} = \sum_j x_{ij} = 1$
- Проблема: несвязность.
- Плохой выход из положения: $\forall S \subset \{0, \ldots, n\}$ $\quad \sum_{i \in S, j \notin S} x_{ij} \geq 1$



- Экспоненциально много неравенств!

Задача коммивояжёра

- Минимизируем $\sum_{ij} c_{ij} x_{ij}$ при ограничениях $\sum_i x_{ij} = \sum_j x_{ij} = 1$
- Проблема: несвязность.
- Хороший выход из положения — условия Таккера:
  - $\forall i, j = 1, \dots, n, \quad i \neq j, \quad u_i - u_j + n x_{ij} \leq n - 1$
  - (дополнительно n новых переменных и $n(n-1)$ неравенств)

Задача коммивояжёра

- Минимизируем $\sum_{ij} c_{ij} x_{ij}$ при ограничениях $\sum_i x_{ij} = \sum_j x_{ij} = 1$
- $\forall i, j = 1, \dots, n, \quad i \neq j, \qquad u_i - u_j + n x_{ij} \leq n - 1$

- Если x_{ij} задают ГЦ, то условия Таккера выполнены:

  Считаем, что начало маршрута в вершине с номером 0. Полагаем $u_i = p$, если вершина с номером i посещалась на p-м шаге. Тогда если $x_{ij} = 1$, то $u_i - u_j = -1$.

Задача коммивояжёра

- Минимизируем $\sum_{ij} c_{ij} x_{ij}$ при ограничениях $\sum_i x_{ij} = \sum_j x_{ij} = 1$
- $\forall i, j = 1, \dots, n, \quad i \neq j, \qquad u_i - u_j + n x_{ij} \leq n - 1$
- Допустим теперь, что x_{ij} задают объединение нескольких циклов. В нём есть цикл $(i_1, i_2, \dots, i_t, i_1)$, не проходящий через вершину 0. Возьмём неравенства:
  - $u_{i_1} - u_{i_2} + n x_{i_1 i_2} \leq n - 1$
  - $u_{i_2} - u_{i_3} + n x_{i_2 i_3} \leq n - 1$
  - …
  - $u_{i_t} - u_{i_1} + n x_{i_t i_1} \leq n - 1$

Задача коммивояжёра

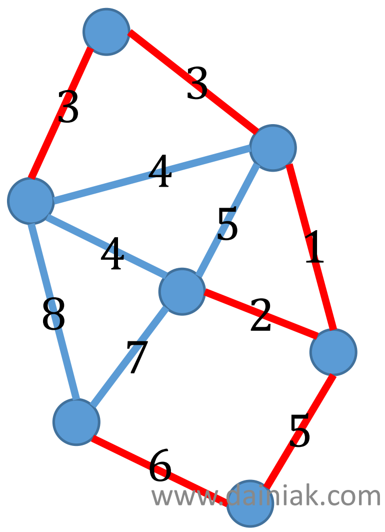
- Минимизируем $\sum_{ij} c_{ij} x_{ij}$ при ограничениях $\sum_i x_{ij} = \sum_j x_{ij} = 1$
- $\forall i, j = 1, \dots, n, \quad i \neq j, \quad u_i - u_j + n x_{ij} \leq n - 1$
- Складываем неравенства:
  - $u_{i_1} - u_{i_2} + n x_{i_1 i_2} \leq n - 1$
  - $u_{i_2} - u_{i_3} + n x_{i_2 i_3} \leq n - 1$
  - …
  - $u_{i_t} - u_{i_1} + n x_{i_t i_1} \leq n - 1$
- Получаем: $tn \leq t(n - 1)$ — противоречие!

Задача коммивояжёра

- Задача коммивояжёра сложная: в общем случае полиномиальных алгоритмов решения не известно.
- Есть различные варианты задачи:
  - Метрическая ЗК (Metric TSP) — веса рёбер графа удовлетворяют неравенству треугольника
  - Евклидова ЗК (Euclidean TSP) — вершины графа являются точками евклидова пространства, веса рёбер определяются евклидовыми расстояниями

Минимальное остовное дерево (Minimal Spanning Tree)

- Дан граф с весами на рёбрах
- Требуется выбрать дерево,
  покрывающее все вершины
  и имеющее как можно меньший вес
- В отличие от ЗК, решается очень просто: жадным алгоритмом!



Метаэвристики

- Задачи дискретной оптимизации очень разнообразны
- Быстрых алгоритмов поиска точного решения часто не известно
- Выход: алгоритмы, выдающие не гарантированно оптимальное, но «близкое к оптимальному» решение
- Эвристики: «выглядящие разумно» подходы к решению конкретной задачи
- Метаэвристики: подходы к построению эвристик

Метаэвристики

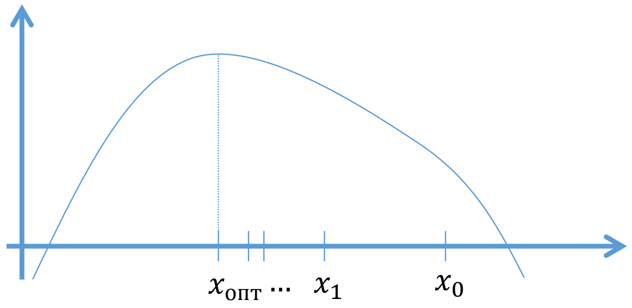
- Локальный поиск
- Жадные алгоритмы
- Эволюционные алгоритмы
- …

Локальный поиск

- Ищем не глобальный, а локальный оптимум (надеемся, что он окажется глобальным)
- Отправляемся из произвольной стартовой точки, помалу сдвигаясь туда, где «теплее» (точка, в которую двигаемся, берётся из небольшой окрестности текущей точки; отсюда *локальность*)

Локальный поиск

- Отправляемся из произвольной стартовой точки, помалу сдвигаясь туда, где «теплее»
- Отлично работает в выпуклой оптимизации:



Локальный поиск

- Отправляемся из произвольной стартовой точки, помалу сдвигаясь туда, где «теплее»
- В дискретной оптимизации всё сложнее:
  - Не можем сдвигаться на «сколь угодно малое ε»
  - Не всегда очевидно, как определять окрестность точки
  - Функции задаются сложно (обычно как суммы специального вида). Неясно, что такое «выпуклая функция».

Локальный поиск: общий алгоритм

- Минимизируем целевую функцию f на множестве S.
- Считаем, что задана *окрестностная функция:*

$$N: \quad S \to 2^S$$

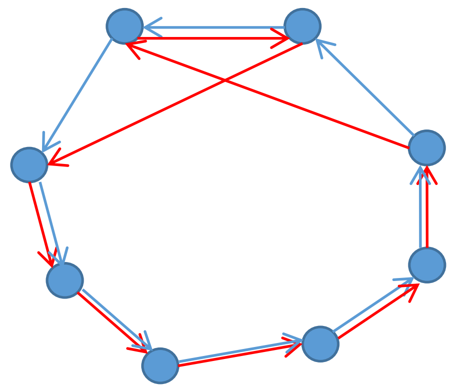
- Алгоритм локального поиска:
  1. x := random(S)
  2. if $\exists y \in N(x)$: $f(y) < f(x)$ then x := y, goto 2
  3. Output(x)

Локальный поиск

- Минимизируем целевую функцию f на множестве S.
- Окрестностная функция N: $\quad S \to 2^S$
  - *полная*, если
    $$\forall x', x'' \in S \;\; \exists x_1, x_2, \dots, x_k: \;\; x_{i+1} \in N(x_i), x_1 = x', x_k = x'',$$
    то есть из любой точки S можно попасть в любую другую, перемещаясь по окрестностям
  - *корректная*, если из любого начального приближения алгоритм локального поиска находит глобальный оптимум
  - *эффективная*, если $|N(s)|$ «невелико» для любого $s \in S$

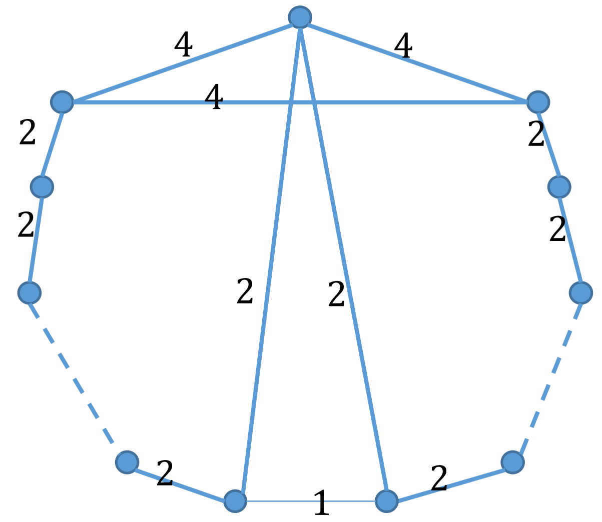
Локальный поиск в задаче коммивояжёра

- Рассмотрим задачу коммивояжёра:
  - S — множество всех гамильтоновых циклов графа
  - $f(H) = \sum_{e \in H} w(e)$
- Как определить окрестностную функцию?
  - Гамильтоновы циклы «близки», если у них много общих рёбер.
  - Самые близкие циклы:
    Удаляем из синего цикла два ребра, добавляем два новых — получаем красный
  - k-окрестность $N_k(H)$ цикла H — множество всех циклов, получаемых из H удалением-добавлением k рёбер



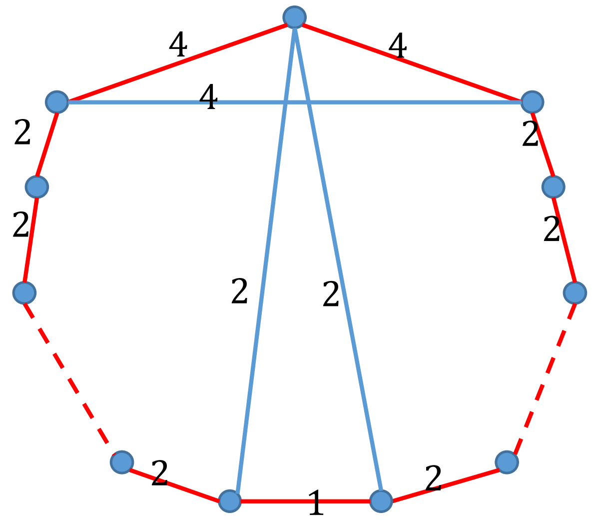
Локальный поиск

- Найдёт ли алгоритм локального поиска оптимальный гамильтонов цикл при использовании функции N_2 ?
- Нет! (Не при любом начальном приближении.)
- Рассмотрим n-вершинный граф:
  (вес неотмеченных рёбер — 5)



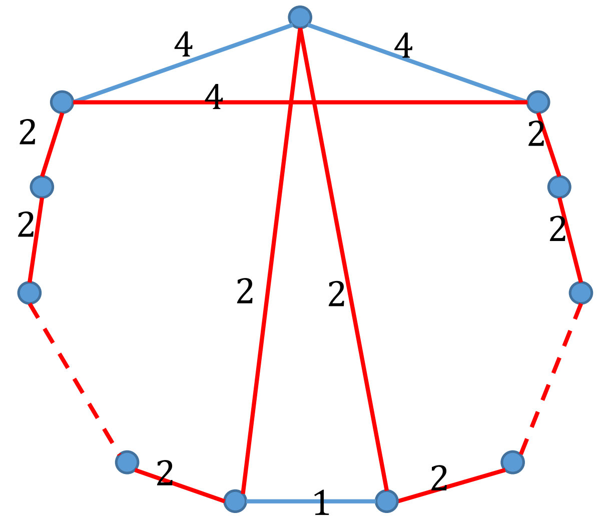
Локальный поиск

- Из указанного начального приближения локальный поиск никуда не сдвинется: вес любого цикла из 2-окрестности больше (веса неотмеченных рёбер равны 5)
- Вес начального приближения равен $2(n-3)+9=2n+3$



Локальный поиск

- В то же время в графе есть цикл, вес которого равен
$$2(n-1)+4=2n+2$$



Локальный поиск: pro et contra

- Вывод: в сложных задачах не стоит надеяться на локальный поиск.
- Замечание: в задаче TSP даже использование $(n-3)$-окрестности не помогает (*упражнение*).
- Очевидное достоинство локального поиска — идейная простота реализации.
- По сути вся сложность организации хорошего локального поиска в задании хорошей окрестностной функции.