Евристические методы и стратегии поиска для решения задач оптимизации

VASSILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Задача наполнения рюкзака

1.3 кг 2500$

0.25 кг 1100$

0.25 кг 1100$

1.6 кг 1900$

0.48 кг 1200$

0.04 кг 430$

0.04 кг 430$

Вместимость
рюкзака 1.8 кг

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Жадный алгоритм

- Давайте начнём брать сначала самые легкие вещи
  - Пример жадного алгоритма
- Для одной задачи может быть несколько типов жадных алгоритмов
  - Все они определяются одинаково, разница лишь в том, по какой метрике определять порядок

Vasiliy Zamolotov MSU
NOT FOR DISTRIBUTION

Задача наполнения рюкзака

0.04 кг 430$

0.04 кг 430$

0.25 кг 1100$

0.25 кг 1100$

0.48 кг 1200$

1.3 кг 2500$

1.6 кг 1900$

Вместимость рюкзака 1.8 кг

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Задача наполнения рюкзака

Общая стоимость 4260$

VASSILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

1.3 кг 2500$

1.6 кг 1900$

0.04 кг 430$

0.04 кг 430$

0.25 кг 1100$

0.25 кг 1100$

0.48 кг 1200$

Вместимость
рюкзака 1.8 кг

Жадный алгоритм

- Для первого подхода при выборе веса в качестве метрики
  - Стоимость 4260$
- Далее попробуем взять стоимость в качестве метрики

VASSILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Задача наполнения рюкзака

1.3 кг 2500$

1.6 кг 1900$

Вместимость рюкзака 1.8 кг

0.48 кг 1200$

0.04 кг 430$

0.04 кг 430$

0.25 кг 1100$

0.25 кг 1100$

VASILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Задача наполнения рюкзака

1.6 кг 1900$

1.3 кг 2500$

0.48 кг 1200$

0.04 кг 430$

0.04 кг 430$

0.25 кг 1100$

0.25 кг 1100$

Вместимость рюкзака 1.8 кг

VASSILY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Задача наполнения рюкзака

Общая стоимость 3700$

1.6 кг 1900$

0.04 кг 430$

0.04 кг 430$

0.25 кг 1100$

0.25 кг 1100$

1.3 кг 2500$

0.48 кг 1200$

Вместимость рюкзака 1.8 кг

VASSILY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Жадный алгоритм

- Для первого подхода при выборе веса в качестве метрики
  - Стоимость 4260$
- Для второго подхода при выборе стоимости в качестве метрики
  - Стоимость 3700$
- Далее попробуем расчётную метрику цена/вес

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Задача наполнения рюкзака

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

1.3 кг 2500$

0.25 кг 1100$

0.25 кг 1100$

1.6 кг 1900$

0.48 кг 1200$

0.04 кг 430$

0.04 кг 430$

Вместимость рюкзака 1.8 кг

Задача наполнения рюкзака

0.04 кг 430$

0.04 кг 430$

0.25 кг 1100$

0.25 кг 1100$

0.48 кг 1200$

1.3 кг 2500$

1.6 кг 1900$

Вместимость рюкзака 1.8 кг

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Задача наполнения рюкзака

Общая стоимость 4260$

1.3 кг 2500$

1.6 кг 1900$

0.04 кг 430$

0.04 кг 430$

0.25 кг 1100$

0.25 кг 1100$

0.48 кг 1200$

Вместимость рюкзака 1.8 кг

VASSILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Жадный алгоритм

- 4260$ лучшее решение?
  - оптимальное решение?

VASSILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Общая стоимость 4700$

1.3 кг 2500$

0.25 кг 1100$

0.25 кг 1100$

Жадный алгоритм

- Для одной задачи может быть несколько типов жадного алгоритма
  - Выдают различный результат
  - Зависят от входных данных

Преимущества:

- Очень быстро моделируются
- Очень быстро рассчитываются
- Найти допустимое решение должно быть легко

Недостатки:

- Не гарантирует качество решения
- Качество решения будет зависеть от входных данных

VASSILY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Цель курса

- Начнём с жадного алгоритма, чтобы понять, как можно его улучшить
- Затем перейдём к продвинутым техникам:
  - Программирование в ограничениях (Constraint programming)
  - Локальный поиск (Local Search)
  - Целочисленное программирование (Integer programming)
- Для того, чтобы:
  - Уметь находить допустимое решение
  - Уметь строить высококачественное решение вне зависимости от входных данных
  - В идеале, обеспечивать оптимальность решения

Моделирование

- Как формализовать оптимизационную задачу в математическую модель?
- Важно, чтобы постановка задачи правильно и полностью вписывалась в математическую модель, иначе задача не будет решена в полной мере

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Задача наполнения рюкзака

- Возьмём I как множество предметов $i \in I$:
  - Где w_i
  - Где v_i
- Вместимость рюкзака K
- Найдем подмножество элементов из I, которые:
  - Будут суммарно выдавать максимальную стоимость
  - При этом не будут превышать вместимость рюкзака K

VASILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Оптимизационная модель

- Как моделировать оптимизационную задачу?
  - Определить множество переменных
    - Это то, что мы будем искать в качестве решения
  - Определить ограничения, выраженные через переменные
    - Это то, что будет определять множество допустимых решений
  - Определить целевую функцию
    - Это то, что выражает цель оптимизации и определяет качество решения
- В результате получается оптимизационная модель
  - Задекларированная формализация
  - Могут быть различные формулировки для одной оптимизационной модели

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Задача наполнения рюкзака

- Множество бинарных переменных x_i, которое определяет, берём ли мы в рюкзак предмет $i \in I$, при этом:
  - $x_i = 1$, если предмет будет взят в рюкзак
  - $x_i = 0$, если предмет взят не будет
- Ограничение задачи:
  - $\sum_{i \in I} w_i x_i \leq K$
- Целевая функция задачи:
  - $\sum_{i \in I} v_i x_i$

Maximize $\sum_{i \in I} v_i x_i$

При этом, $\sum_{i \in I} w_i x_i \leq K,\ x_i \in \{0,1\}, (i \in I)$

VASILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Экспоненциальный рост

- Как много может быть возможных конфигураций?
  - $(0, 0, 0, \ldots, 0), (0, 0, 0, \ldots, 1), \ldots, (1, 1, 1, \ldots, 1)$
- Не все из конфигураций попадают в допустимое множество
  - Из-за ограничения на вместимость рюкзака
- Сколько возможных комбинаций?
  - $2^{|I|}$
- Как много времени потребуется на расчёт?
  - Допустим на тестирование одной конфигурации уходит 1 миллисекунда
  - Если количество элементов равно 50, то на перебор всех вариантов уйдет **1.285.273.866 веков**

VASILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Динамическое программирование

- Одна из наиболее популярных техник для работы с оптимизационными задачами
  - Для некоторых типов задач подходит хорошо, но с некоторыми не работает совсем
- Базовый принцип
  - Разделяй и властвуй
  - Расчёт снизу вверх

VASILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Динамическое программирование

- Основные понятия и определения
  - Пусть есть множество элементов $I = \{1, 2, 3, \dots, n\}$
  - $O(k, j)$ определяет оптимальное решение для проблемы наполнения рюкзака с вместимостью рюкзака k и элементами $[1 \dots j]$

Maximize $\sum_{i \in 1 \dots j} v_i x_i$

При этом, $\sum_{i \in 1 \dots j} w_i x_i \leq k,\ x_i \in \{0,1\}, (i \in 1 \dots j)$

- Нам интересно найти наилучшее значение для $O(K, n)$

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Рекуррентные отношения

- Предположим, что мы знаем как найти
  - $O(k, j-1)$ для каждого k из $0 \dots K$
- Мы хотим найти $O(k, j)$
  - Что больше задачи $O(k, j-1)$ всего на 1 предмет j
- Если $w_j \le k$ у нас есть два случая
  - Мы не возьмём в рюкзак предмет j, потому что решение $O(k, j-1)$ лучше
  - Мы возьмём в рюкзак предмет j и лучшее решение будет значением $v_j + O(k - w_j, j-1)$
- В итоге
  - $O(k, j) = \max(O(k, j-1), v_j + O(k - w_j, j-1))$, если $w_j \le k$
  - $O(k, j) = O(k, j-1)$, если $w_j > k$
- Очевидно
  - $O(k, 0) = 0$ для любого k

VASILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Рекуррентные отношения

- Можем написать простейшую программу

```
def O(k,j):
    if j==0:
        return 0
    elif w[j]<=k:
        return max(O(k,j-1),v[j]+O(k-w[j],j-1))
    else:
        return O(k,j-1)
```

- Насколько эффективен этот алгоритм?

VASSILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Рекуррентные отношения

- Рассмотрим расчёт чисел Фибоначчи

```python
def fib(n):
    if n==0 or n==1:
        return 1
    else:
        return fib(n-2)+fib(n-1)
```

- Насколько эффективен этот алгоритм?
  - Для каждого n необходимо рассчитывать функцию n-1 раз
  - Что само по себе не очень эффективно

VASILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Динамическое программирование

- Рассчитываем рекурсивным методом снизу вверх
  - Рассчитываем для нуля предметов
  - Рассчитываем для одного предмета
  - …
  - Рассчитываем для всех предметов

Maximize $5x_1 + 4x_2 + 3x_3$

При этом,

$4x_1 + 5x_2 + 2x_3 \leq 9,\ \ x_i \in \{0,1\}, (i \in 1 \dots 3)$

Живой пример

VASSILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Динамическое программирование

Вместимость	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	3
3	0	0	0	3
4	0	5	5	5
5	0	5	6	6
6	0	5	6	8
7	0	5	6	9
8	0	5	6	9
9	0	5	11	**11**

Maximize $5x_1 + 4x_2 + 3x_3$

При этом,
$4x_1 + 5x_2 + 2x_3 \leq 9, \; x_i \in \{0,1\}, (i \in 1 \dots 3)$

VASILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Динамическое программирование

Вместимость	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	**44**

Maximize $16x_1 + 19x_2 + 23x_3 + 28x_4$

При этом,

$2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 7, \ x_i \in \{0,1\}, (i \in 1 \dots 4)$

Динамическое программирование

- В чём сложность алгоритма?
  - Время на заполнение таблицы
  - То есть время на поиск $O(k, j)$
- Время расчёта полиномиально?
  - Безусловно, нет
  - Нижняя оценка - log(K) бит для расчёта K
  - Является псевдо-полиномом – имеет полиномиальное время в случае, если K мало
  - Следовательно – подход эффективен для небольшого K

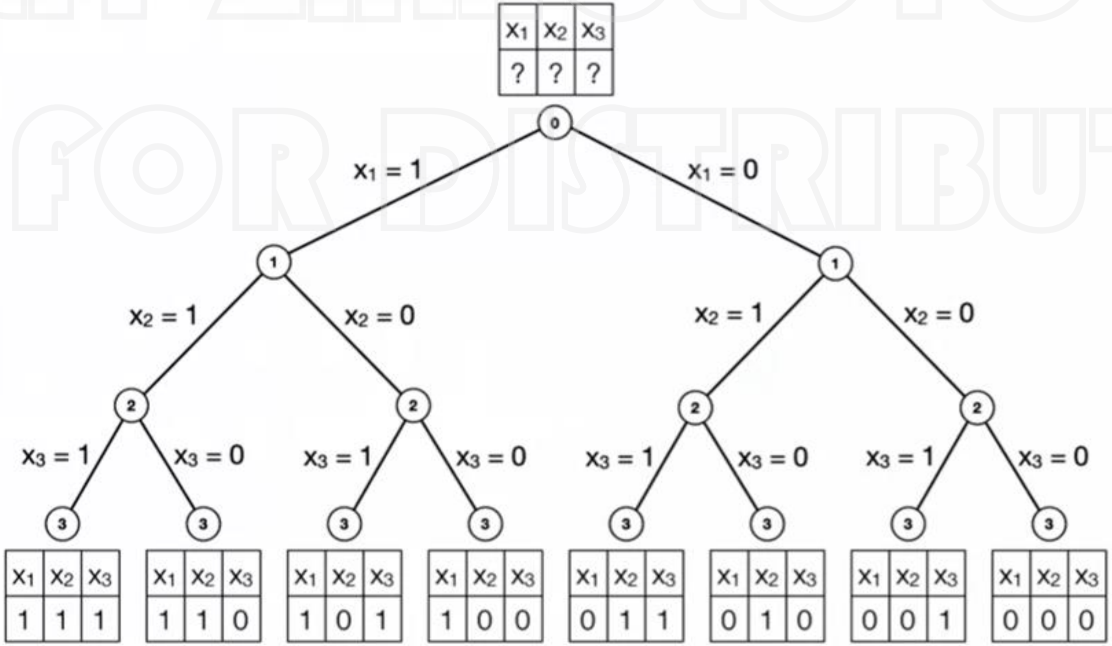
VASILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Метод ветвей и границ

Maximize $45x_1 + 48x_2 + 35x_3$

При этом,

$5x_1 + 8x_2 + 3x_3 \leq 10,\ x_i \in \{0,1\}, (i \in 1 \dots 3)$



VASSILY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Метод ветвей и границ

- Итерация двух шагов
  - Ветвление
  - Ограничение
- Ветвление
  - Разбиение задачи на несколько подзадач
- Ограничение
  - Поиск оптимистичной оценки для лучшего решения на основе подзадач
  - Максимизация нижней границы и минимизация верхней границы

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Метод ветвей и границ

- Как получить оптимистичную оценку?
  - Ослабить условия задачи

Maximize $45x_1 + 48x_2 + 35x_3$

При этом,

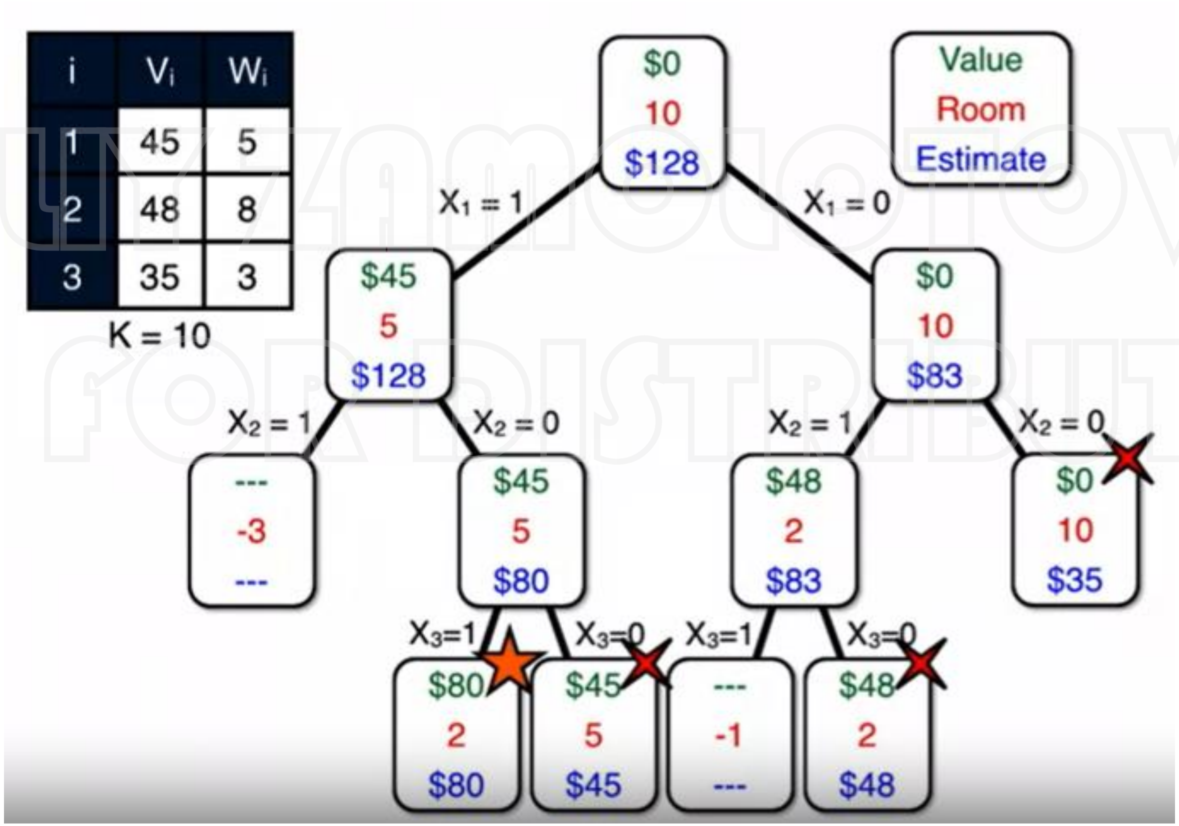
$5x_1 + 8x_2 + 3x_3 \leq 10, \; x_i \in \{0,1\}, (i \in 1 \ldots 3)$

- Как мы можем ослабить задачу наполнения рюкзака?
  - Ослабить условие вместимости рюкзака

Живой пример

VASSILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Метод ветвей и границ



VASSILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Метод ветвей и границ

- Подход не самый лучший, ограничили только несколько ветвей

Maximize $45x_1 + 48x_2 + 35x_3$

При этом,

$5x_1 + 8x_2 + 3x_3 \leq 10, \; x_i \in \{0,1\}, (i \in 1 \dots 3)$

- Как мы ещё можем ослабить задачу наполнения рюкзака?
  - Разбить элементы на части (по примеру плитки шоколада)

VASSILY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Линейная релаксация

- Переменные могут принимать дробные значения от 0 до 1

Maximize $45x_1 + 48x_2 + 35x_3$

При этом,

$5x_1 + 8x_2 + 3x_3 \leq 10, \; 0 \leq x_i \leq 1, (i \in 1 \dots 3)$

- Это называется линейной релаксацией
  - Будет более подробно рассказано об этом позже в курсе

VASSILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Линейная релаксация

- Как мы можем решить новую задачу?
  - Расположить предметы по убыванию показателя v_j/w_j
  - Получится своего рода удельный показатель на 1 кг
- Наполняем рюкзак, пока не закончится в нём место
  - При этом последний предмет, который попадёт в рюкзак, вероятно, не будет целым

Maximize $45x_1 + 48x_2 + 35x_3$

При этом,
$5x_1 + 8x_2 + 3x_3 \leq 10,\ 0 \leq x_i \leq 1, (i \in 1 \dots 3)$

- В конкретном примере:
  - $v_1/w_1 = 9$, $v_2/w_2 = 6$, $v_3/w_3 = 11.7$
  - Предметы 3 и 1 поместятся в рюкзак полностью
  - Предмет 2 поместится только на ¼
  - Оценка 92

VASILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Линейная релаксация

- Почему это верно?
  - Пусть $x_i = y_i / v_i$

Maximize $\sum_{i \in 1 \dots j} y_i$

При этом, $\sum_{i \in 1 \dots j} w_i y_i / v_i \le k,\ 0 \le y_i \le 1, (i \in 1 \dots j)$

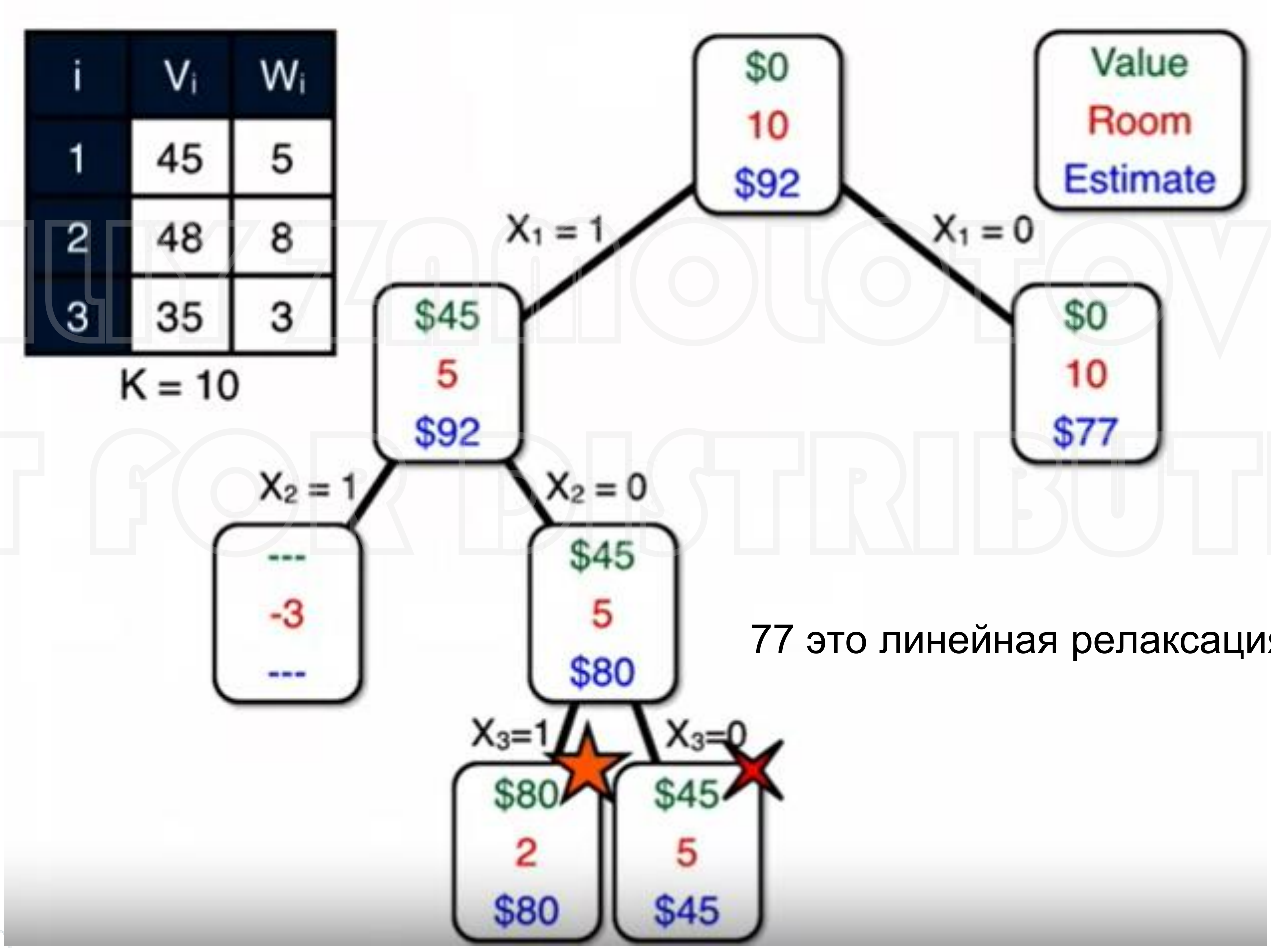
VASSILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Почему это не будет решением задачи?

Живой пример

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Использование линейной релаксации



77 это линейная релаксация на 2 и 3 предметы

VASSILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Стратегии поиска в методе ветвей и границ

- Первый по глубине (depth-first)
  - Обрезаем те ветви, оценка которых хуже уже найденного решения
- Первый лучший (best first)
  - Выбираем ту ветвь, которая имеет лучшую оценку
- Наименьшее расхождение (least discrepancy)
  - Доверяем жадному алгоритму

VASILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Первый по глубине

- Идём вглубь
- Когда обрезаем ветви?
  - Когда находим новую ветвь с оптимистичной оценкой ниже уже найденного решения
- За найденной ветвью обрезаем все нижестоящие
- Насколько это эффективно с точки зрения использования памяти?
  - Довольно неэффективно, потому что приходится хранить большое количество информации

VASSILY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Первый лучший

- Идём к лучшему
- Когда обрезаем ветви?
  - Когда все ветви дают решение хуже найденного
- За найденной ветвью обрезаем все нижестоящие
- Насколько это эффективно с точки зрения использования памяти?
  - Так же неэффективно

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

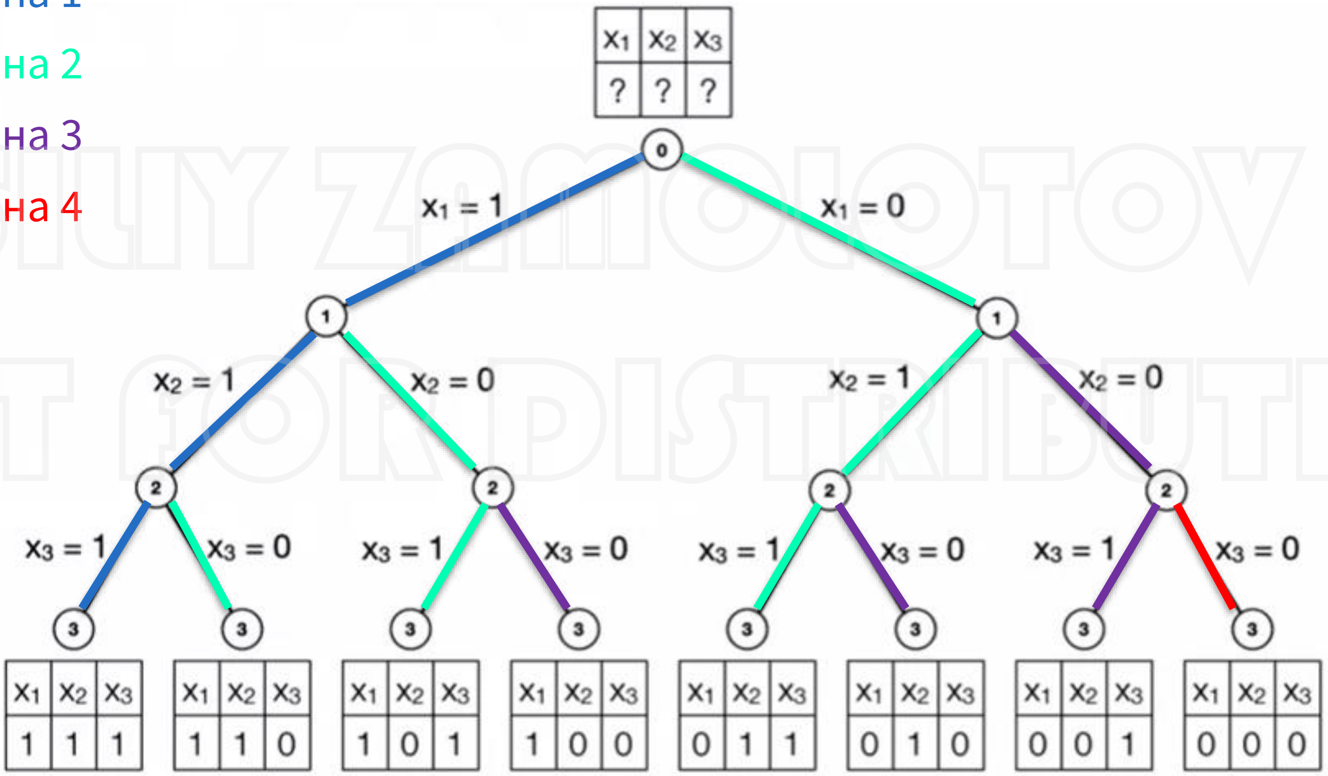
Наименьшее расхождение

- Предположим, что у нас есть отличная эвристика
  - Делает очень мало ошибок
  - Дерево поиска бинарное
  - Эвристика указывает на верные значения в левой стороне
  - Ветви с правой стороны считаются неверными
- Ограниченный поиск расхождений (Limited Discrepancy Search (LDS))
  - Избегает отклонения от эвристики
  - Исследует дерево решений увеличивая порядок отклонений
  - Доверяет эвристике меньше и меньше
- Порядок отклонений увеличивается волнами
  - Без отклонений от эвристики
  - С одним отклонением
  - С двумя отклонениями

VASSILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Наименьшее расхождение

- Волна 1
- Волна 2
- Волна 3
- Волна 4



VASILIY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Наименьшее расхождение

- Доверяем эвристике
- Когда обрезаем ветви?
  - Когда вся следующая волна ниже предыдущей по оценке
- Насколько это эффективно с точки зрения использования памяти?
  - Зависит от реализации и выбранной эвристики

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Заключение

- Релаксация и стратегия поиска – важнейшая часть в решении задач оптимизации
- Важно пробовать различные подходы для ускорения процесса расчёта и использования памяти:
  - Исследовать природу рассматриваемой задачи
  - Подобрать максимально эффективную эвристику там, где это возможно
  - Использовать релаксацию там, где это может быть эффективно
  - Выбрать наиболее подходящий метод поиска

VASILIY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Задание для воркшопа

- Решить задачу наполнения рюкзака для различного количества предметов

Maximize $\sum_{i \in I} v_i x_i$

При этом, $\sum_{i \in I} w_i x_i \leq K,\ x_i \in \{0,1\}, (i \in 1..n)$

Вход

n	K
v_1	w_1
v_2	w_2
…	…
v_n	w_n

Выход

Целевая функция	Оптимальность

x1	x2	x3	…	xn

VASSILY ZAMOLOTOV MSU NOT FOR DISTRIBUTION

Задание для воркшопа

- Решить задачу наполнения рюкзака для различного количества предметов

Maximize $\sum_{i \in I} v_i x_i$

При этом, $\sum_{i \in I} w_i x_i \leq K,\ x_i \in \{0,1\}, (i \in 1..n)$

VASSILY ZAMOLOTOV MSU
NOT FOR DISTRIBUTION

Вход

n	K
v_1	w_1
v_2	w_2
…	…
v_n	w_n

4	11
8	4
10	5
15	8
4	3

Выход

Целевая функция	Оптимальность
19	0

x1	x2	x3	x4
0	0	1	1

Солверы

- Constraint Programming:
  - CHOCO - java library, open source
  - Gecode - c++, free
  - FICO Express - binary, free with academic license
  - JACOP - java , open source
  - CPLEX – binary, free with academic license
  - MiniZinc / G12 - binary, free for students
  - or-tools - C++ , open source, APIs - Java, Python, and .NET
  - SAS OR – binary, free with academic license
- Local Search:
  - Local Solver - binary, free with academic license
  - OptaPlanner - java, open source
  - CPLEX – binary, free with academic license
  - SAS OR – binary, free with academic license
  - …
- Mixed Integer Programming:
  - BCP - c++, open source
  - CBC - c++, open source
  - CPlex - binary, free with academic license
  - GLPK - c, open source
  - gurobi - binary, free with academic license
  - LPSolve - c, open source
  - MIP – binary, open source
  - SCIP - binary, free for academic use
  - SAS OR - binary, free with academic license
- Non-Linear Optimization:
  - Artelis Knitro – binary, free with academic license
  - CPLEX – binary, free with academic license
  - SAS OR – binary, free with academic license
  - …

Пакеты для Python, часть 1

MINLP+MIQP+MILP +NLP+IP+LP	
Package	Link
gekko	Official
knitro	Official
lindo	Official
midaco	Official
naginterfaces	Official
octeract	Official
pydrake	Official
pygmo	Official
pyomo	Official
pyscipopt	Official
xpress	Official

MIQP+MILP+IP +LP	
Package	Link
copt	Official
cplex	Official
docplex	Official
gurobipy	Official
highs	Official
localsolver	Official
mosek	Official
optlang	Official
sasoptpy	Official

MILP+IP+LP	
Package	Link
cvxopt	Official
cvxpy	Official
cylp	Official
flowty	Official
lpsolve55	Official
Mindoptpy	Official
Mip	Official
Ortools	Official
picos	Official
pulp	Official
pymprog	Official
swiglpk	Official

NLP+LP	
Package	Link
iminuit	Official
nlopt	Official
openmdao	Official
pyopt	Official
scipy	Official
worhp	Official
cyipopt	Official

CP	
Package	Link
cplex	Official
cpmpy	Official
gecode-python	Official
kalis	Official
minizinc	Official
optapy	Official
ortools	Official
z3-solver	Official

Пакеты для Python, часть 2

GPP

Package	Link
arm-mango	Official
ax	Official
bayesian-optimization	Official
bayesianevolution	Official
bayeso	Official
bayesopt	Official
bolbib	Official
cma	Official
cmaes	Official
cuopt	Official
deap	Official
dfoalgos	Official
dfogn	Official
dlib	Official
evolopy	Official
freelunch	Official
gaft	Official
geneticalgorithm	Official
gyopt	Official
hebo	Official
heuristic_optimization	Official
hpbandster	Official
hyperopt	Official
inspyred	Official
mealpy	Official
mipego	Official
mystic	Official
nevergrad	Official
niapy	Official
oasis	Official
optuna	Official
optuner	Official
opytimizer	Official
pagmo	Official
pdfo	Official
platypus	Official
proxmin	Official
py-bobyqa	Official
pydogs	Official
pygpgo	Official
pymoo	Official
pyopus	Official
pypesto	Official
pyriad	Official
pysmac	Official
pysot	Official
pyswarms	Official
rapids-NeurIPS	Official
ray	Official
rbfopt	Official
scikit-optimize	Official
simanneal	Official
simple	Official
solidpy	Official
spearmint	Official
spotpy	Official
ssb-optimize	Official
swarmlib	Official
swarmpackagepy	Official
tgo	Official
turbo-NeurIPS	Official
turbo	Official
ultraopt	GitHub
yabox	GitHub
zoopt	GitHub