МатОптимизация для honeypot-размещения

Н. Н. Кузюрин С. А. Фомин

26 июля 2019 г.

Постановка

АС ВН : $\{h_1^{AC}, h_2^{AC}, \ldots, h_n^{AC}\}, \{t_1^{AC}, t_2^{AC}, \ldots, t_n^{AC}\}$

- $h_1^{AC} \in H$ — количество элементов АС i-го типа.
- n — количество типов (пример типов: Сервер БД, файловый сервер, почтовый сервер, прокси сервер, ПЭВМ оператора, ПЭВМ руководителя и т.д.),
- $t_i^{AC} \in T^{AC}$ — время сканирования i-го типа АС.

ЛИС: ложная информационная система

Характеризуется тремя уровнями:

- K_1 — полностью ложная сеть разворачивается до реальной АС на входе из внешней сети
- K_2 — ложные узлы развернуты вперемешку с реальными узлами АС.
- K_3 — ложные сетевые сервисы, поднятые на реальных серверах и ПЭВМ вперемешку с работающими сервисами

Уровни ЛИС характеризуются

K_1 — $(L_{par}, \quad x_1^{K_1}, x_2^{K_1}, \ldots, x_n^{K_1}, \quad c_1^{K_1}, c_2^{K_1}, \ldots, c_n^{K_1}, \quad t_1^{K_1}, t_2^{K_1}, \ldots, t_n^{K_1},)$

L_{par} — длина пароля нестандартной сетевой аутентификации.

$x_i^{K_1} \in K_1$ — количество элементов ЛИС i-го типа n — количество типов.

$c_i^{K_1} \in C^{K_1}$ — затраты на создание i-го элемента ЛИС (пример: объем оперативной памяти, выделяемой для создания ложного узла i-го типа, в виртуальной среде)

$t_i^{K_1} \in T^{K_1}$ — время, необходимое для сетевого сканирования ложного узла i-го типа.

K_2 — $(x_1^{K_2}, x_2^{K_2}, \ldots, x_n^{K_2}, \quad c_1^{K_2}, c_2^{K_2}, \ldots, c_n^{K_2}, \quad t_1^{K_2}, t_2^{K_2}, \ldots, t_n^{K_2})$

$x_i^{K_2} \in K_2$ — количество элементов ЛИС i-го типа.

$c_i^{K_2} \in C^{K_2}$ — затраты на создание i-го элемента ЛИС

$t_i^{K_2} \in T^{K_2}$ — время, необходимое для сетевого сканирования ложного узла i-го типа.

K_3 — $(x_1^{K_3}, x_2^{K_3}, \ldots, x_m^{K_3}, \quad c_1^{K_3}, c_2^{K_3}, \ldots, c_m^{K_3}, \quad t_1^{K_3}, t_2^{K_3}, \ldots, t_m^{K_3},)$

m — количество сетевых протоколов.

$x_i^{K_3} \in (0, 1)$ — бинарный индикатор использования того или иного ложного сетевого протокола на том или ином порту ЛИС уровня приложения.

$c_i^{K_3} \in C^{K_3}$ — затраты на создание i-го сетевого протокола ЛИС (пример: выделяется объем оперативной памяти для создания ложных сетевых сервисов)

- $t_i^{K_3} \in T^{K_3}$ — время, необходимое для сетевого сканирования $x_i^{K_3}, i \in 1, \ldots m.$

Найти

Требуется найти: $T^{K_1} + T^{K_2} + T^{K_3}$

Ограничение: затраты на создание многоуровневой ЛИС не должны превышать заданных на каждом уровне: $C^{K_1}, C^{K_2}, C^{K_3}$

Оптимизация

- Требуется найти: $T^{K_1} + T^{K_2} + T^{K_3}$
- $\Rightarrow T^{K_1} + T^{K_2} + T^{K_3} \rightarrow MAX$

Учитывая отдельность ограничений, все распадается на три независимых оптимизационных задачи:

$$\begin{aligned} \forall k &\in (1,2,3) \\ T^{K_k} = \sum_i t_i^{K_k} \times x_i^{K_k} &\rightarrow MAX \\ \sum_i c_i^{K_k} \times x_i^{K_k} &\leq C^{K_k} \\ 0 \leq x_i^{K_k} &\leq b_i^{K_k} \leftarrow \text{Целые или индикаторы} \end{aligned}$$

Выводы

- ЦЛП — Целочисленное линейное программирование
- ПЦЛП — Положительное целочисленное линейное программирование
- Одно ограничение!
- Задача о рюкзаке (k=3), или мультирюкзаке
- Есть FPTAS-алгоритм — полиномиальный алгоритм с гарантированной точностью
- Точное решение при весьма приближенной модели и не нужно
- С мультирюкзаком надо еще посмотреть, но что-то было.

Задача о рюкзаке

Задача

«0–1 Рюкзак (Knapsack)»

Даны:

$c_1, \ldots, c_n, c_j \in N$ — *«стоимости» предметов;*

$a_1, \ldots, a_n, a_j \in N$ — *«размеры» или «веса»;*

$B \in N$ — *«размер рюкзака».*

Найти максимальное значение f^* *целевой функции*

$$f \equiv \sum_{i=1}^{n} c_i x_i \to \max$$

с ограничением на размер «рюкзака»:

$$\sum_{i=1}^{n} a_i x_i \leq B, \quad x_i \in \{0, 1\}.$$

Определение

Алгоритм с мультипликативной ошибкой не более $(1+\varepsilon)$*, где* $\varepsilon > 0$*, называется* ε***-оптимальным.***

Определение

Алгоритм с мультипликативной ошибкой не более $(1+\varepsilon)$, где $\varepsilon > 0$, называется ***ε-оптимальным.***

Определение

Полностью полиномиальной аппроксимационной схемой *(FPTAS) называется приближенный алгоритм, в котором уровень точности ε выступает в качестве нового параметра, и алгоритм находит ε-оптимальное решение за время, ограниченное полиномом от длины входа и величины ε^{-1}.*

«Рюкзак»: отбор легких решений

```
def knapsack_dynprog_lightest(items, B):
    T = {0: ItemSet()} # Цена -> самый легкий набор

    for item in items: # Цикл по всем предметам
        newT = []
        for sol in T.values():      # по всем частичным
            test = sol + item       # формируем новый набор
            if test.weight <= B and (test.cost not in T
                    or test.weight < T[test.cost].weight):
                newT.append(test) # подходит!


        for sol in newT:                # регистрируем
            T[sol.cost] = sol           # новые решения

    return T[max(T.keys())]     # возвращаем самое дорогое
```

«Рюкзак»: отбор легких решений

Предметы ($\frac{\text{стоимость}}{\text{вес}}$): $[\frac{6}{3}, \frac{3}{4}, \frac{2}{5}, \frac{5}{6}, \frac{5}{7}, \frac{1}{8}]$, $B = 9$

T	item	newT
0: $\frac{0}{0}$	$\frac{6}{3}$	$[\frac{6}{3}]$
0: $\frac{0}{0}$, 6: $\frac{6}{3}$	$\frac{3}{4}$	$[\frac{3}{4}, \frac{9}{7}]$
0: $\frac{0}{0}$, 9: $\frac{9}{7}$, 3: $\frac{3}{4}$, 6: $\frac{6}{3}$	$\frac{2}{5}$	$[\frac{2}{5}, \frac{5}{9}, \frac{8}{8}]$
0: $\frac{0}{0}$, 2: $\frac{2}{5}$, 3: $\frac{3}{4}$, 5: $\frac{5}{9}$, 6: $\frac{6}{3}$, 8: $\frac{8}{8}$, 9: $\frac{9}{7}$	$\frac{5}{6}$	$[\frac{5}{6}, \frac{11}{9}]$
0: $\frac{0}{0}$, 2: $\frac{2}{5}$, 3: $\frac{3}{4}$, 5: $\frac{5}{6}$, 6: $\frac{6}{3}$, 8: $\frac{8}{8}$, 9: $\frac{9}{7}$, 11: $\frac{11}{9}$	$\frac{5}{7}$	[]
0: $\frac{0}{0}$, 2: $\frac{2}{5}$, 3: $\frac{3}{4}$, 5: $\frac{5}{6}$, 6: $\frac{6}{3}$, 8: $\frac{8}{8}$, 9: $\frac{9}{7}$, 11: $\frac{11}{9}$	$\frac{1}{8}$	$[\frac{1}{8}]$

Оптимальное решение: $\frac{11}{9}$

Лемма

Сложность алгоритма с отбором «легких» решений — $O(nf^*)$.

Округлим $c_i \leftarrow \lfloor c_i / scale \rfloor \cdot scale$, т. е. выполнено $c_i \equiv 0 \pmod{scale}$:

- c_i можно поделить на $scale \Rightarrow$ это не изменит оптимального набора.
- Время работы $\Rightarrow O(\frac{nf^*}{scale})$.
- Веса a_i не меняли $\Rightarrow$ любое допустимое решение «округленной» допустимо для исходной.
- Потери «округления» $\Rightarrow$ оптимум получившейся задачи будет меньше исходной.

Стоит ли игра свеч?

«округленная» задача

$\tilde{c}_i$ — Стоимости, $\tilde{c}_i = \lfloor c_i / scale \rfloor \cdot scale$;

$\tilde{x}_i$ — Включение предмета в оптимальный набор, $\tilde{x}_i \in \{0, 1\}$;

$\tilde{f}$ — Оптимум «округленной» задачи, $\tilde{f} = \sum_{i=1}^{n} \tilde{c}_i \tilde{x}_i$.

«округленная» задача

$\tilde{c}_i$ — Стоимости, $\tilde{c}_i = \lfloor c_i / scale \rfloor \cdot scale$;

$\tilde{x}_i$ — Включение предмета в оптимальный набор, $\tilde{x}_i \in \{0, 1\}$;

$\tilde{f}$ — Оптимум «округленной» задачи, $\tilde{f} = \sum_{i=1}^{n} \tilde{c}_i \tilde{x}_i$.

«Округление» только одного j-го предмета.

«округленная» задача

$\tilde{c}_i$ — Стоимости, $\tilde{c}_i = \lfloor c_i / scale \rfloor \cdot scale$;

$\tilde{x}_i$ — Включение предмета в оптимальный набор, $\tilde{x}_i \in \{0, 1\}$;

$\tilde{f}$ — Оптимум «округленной» задачи, $\tilde{f} = \sum_{i=1}^{n} \tilde{c}_i \tilde{x}_i$.

«Округление» только одного j-го предмета.

$\tilde{x}_j = 1$: j-й и в оптимальном наборе. На нем «теряем» $c_j - \tilde{c}_j \leq scale$.

«округленная» задача

$\tilde{c}_i$ — Стоимости, $\tilde{c}_i = \lfloor c_i / scale \rfloor \cdot scale$;

$\tilde{x}_i$ — Включение предмета в оптимальный набор, $\tilde{x}_i \in \{0, 1\}$;

$\tilde{f}$ — Оптимум «округленной» задачи, $\tilde{f} = \sum_{i=1}^{n} \tilde{c}_i \tilde{x}_i$.

«Округление» только одного j-го предмета.

$\tilde{x}_j = 1$: j-й и в оптимальном наборе. На нем «теряем» $c_j - \tilde{c}_j \leq scale$.

$\tilde{x}_j = 0$: Вместо j-го взяли что-то подороже $\tilde{c}_j \Longrightarrow$ теряем $\leq c_j - \tilde{c}_j \leq scale$.

«округленная» задача

$\tilde{c}_i$ — Стоимости, $\tilde{c}_i = \lfloor c_i / scale \rfloor \cdot scale$;

$\tilde{x}_i$ — Включение предмета в оптимальный набор, $\tilde{x}_i \in \{0, 1\}$;

$\tilde{f}$ — Оптимум «округленной» задачи, $\tilde{f} = \sum_{i=1}^{n} \tilde{c}_i \tilde{x}_i$.

«Округление» только одного j-го предмета.

$\tilde{x}_j = 1$: j-й и в оптимальном наборе. На нем «теряем» $c_j - \tilde{c}_j \leq scale$.

$\tilde{x}_j = 0$: Вместо j-го взяли что-то подороже $\tilde{c}_j \Longrightarrow$ теряем $\leq c_j - \tilde{c}_j \leq scale$.

«округлять» все предметы $\Rightarrow f^* - \tilde{f} \leq n \cdot scale$.

Решение «округленной» — аппроксимация исходной

f^*: оптимум исходной задачи;

f': стоимость аппроксимации,
$f' = \sum_{i=1}^{n} c_i \tilde{x}_i \geq \sum_{i=1}^{n} \tilde{c}_i \tilde{x}_i = \tilde{f}.$

Решение «округленной» — аппроксимация исходной

f^*: оптимум исходной задачи;

f': стоимость аппроксимации,
$f' = \sum_{i=1}^{n} c_i \tilde{x}_i \geq \sum_{i=1}^{n} \tilde{c}_i \tilde{x}_i = \tilde{f}$.

Абсолютная погрешность $f^* - f' \leq f^* - \tilde{f} \leq n \cdot scale$.

Решение «округленной» — аппроксимация исходной

f^*: оптимум исходной задачи;

f': стоимость аппроксимации,
$f' = \sum_{i=1}^{n} c_i \tilde{x}_i \geq \sum_{i=1}^{n} \tilde{c}_i \tilde{x}_i = \tilde{f}$.

Абсолютная погрешность $f^* - f' \leq f^* - \tilde{f} \leq n \cdot scale$.

Погрешность $\leq \frac{\varepsilon}{1+\varepsilon} f^*$

Решение «округленной» — аппроксимация исходной

f^*: оптимум исходной задачи;

f': стоимость аппроксимации,
$f' = \sum_{i=1}^{n} c_i \tilde{x}_i \geq \sum_{i=1}^{n} \tilde{c}_i \tilde{x}_i = \tilde{f}$.

Абсолютная погрешность $f^* - f' \leq f^* - \tilde{f} \leq n \cdot scale$.

Погрешность $\leq \frac{\varepsilon}{1+\varepsilon} f^*$
$\Rightarrow$ решение — ε-приближенное:

$$f' \geq f^* - \frac{\varepsilon}{1+\varepsilon} f^* = \frac{f^*}{(1+\varepsilon)}.$$

Решение «округленной» — аппроксимация исходной

f^*: оптимум исходной задачи;

f': стоимость аппроксимации,
$f' = \sum_{i=1}^{n} c_i \tilde{x}_i \geq \sum_{i=1}^{n} \tilde{c}_i \tilde{x}_i = \tilde{f}$.

Абсолютная погрешность $f^* - f' \leq f^* - \tilde{f} \leq n \cdot scale$.

Погрешность $\leq \frac{\varepsilon}{1+\varepsilon} f^*$
$\Rightarrow$ решение — ε-приближенное:

$$f' \geq f^* - \frac{\varepsilon}{1+\varepsilon} f^* = \frac{f^*}{(1+\varepsilon)}.$$

$$scale \to \max$$

$$scale \leq \frac{\varepsilon f^*}{n(1+\varepsilon)}$$

Решение «округленной» — аппроксимация исходной

f^*: оптимум исходной задачи;

f': стоимость аппроксимации,
$f' = \sum_{i=1}^{n} c_i \tilde{x}_i \geq \sum_{i=1}^{n} \tilde{c}_i \tilde{x}_i = \tilde{f}$.

Абсолютная погрешность $f^* - f' \leq f^* - \tilde{f} \leq n \cdot scale$.

Погрешность $\leq \frac{\varepsilon}{1+\varepsilon} f^*$
$\Rightarrow$ решение — ε-приближенное:

$$f' \geq f^* - \frac{\varepsilon}{1+\varepsilon} f^* = \frac{f^*}{(1+\varepsilon)}.$$

$$scale \to \max$$

$$scale \leq \frac{\varepsilon f^*}{n(1+\varepsilon)}$$

Нижняя оценка оптимума $f_{lb} \leq f^* \Rightarrow scale = \max\left\{1, \frac{\varepsilon f_{lb}}{n(1+\varepsilon)}\right\}$.

PTAS для рюкзака

```
def knapsack_fptas(items, B, epsilon, lower_bound):
    # Вычисляем нижнюю оценку стоимости
    F_lb = lower_bound(items, B)




    # параметр округления scale
    scale = epsilon * F_lb / len(items) / (1 + epsilon)



    # Набор с округленными стоимостями
    Ds = [Item(item.cost/scale, item.weight) for item in items]

    knapset, indices = knapsack_dynprog_lightest(Ds, B)
    ApproxCost = sum(items[i].cost for i in indices)
```

Выбор f_{lb}:«*MaxItemCost*»

Тривиальная нижняя оценка — стоимость самого дорогого предмета:

$$f_{lb} \equiv c_{\max} = \max_i c_i.$$

Выбор f_{lb}:«*MaxItemCost*»

Тривиальная нижняя оценка — стоимость самого дорогого предмета:

$$f_{lb} \equiv c_{\max} = \max_i c_i.$$

Сложность «$KnapsackFPTAS_{MaxItemCost}$»:

$$O\left(\frac{nf'}{scale}\right) \leq O\left(\frac{n \cdot nc_{max}}{scale}\right) =$$

$$= O\left(\frac{n \cdot nc_{max}}{\frac{c_{max}\varepsilon}{n(1+\varepsilon)}}\right) = O\left(\frac{n^3(1+\varepsilon)}{\varepsilon}\right) = O\left(\frac{n^3}{\varepsilon}\right).$$

«Жадный-2» для «Рюкзака»

```
def knapsack_greedy(T, B):
    T.sort(key=profitableness)
    Cmax = Cg = Ag = 0
    for c, a in T:
        if a <= B:
            Cmax = max(c, Cmax)
        # если лезет в рюкзак
        if Ag + a <= B:
            # берем предмет (c,a)
            Ag = Ag + a
            Cg = Cg + c
    #выбираем, что больше
    return max(Cg, Cmax)
```

```
Вес рюкзака B= 10 кг
Входной массив  T <= [(3, 6), (4, 3), (5, 2), (6, 5), (7, 5), (8, 1)]
Отсортированный T => [(8, 1), (5, 2), (7, 5), (4, 3), (6, 5), (3, 6)]
  Берем предмет: <= ($8 , 1 кг)
  Берем предмет: <= ($5 , 2 кг)
  Берем предмет: <= ($7 , 5 кг)
Cg=$20 или Cmax=$8 ?
Набран рюкзак стоимостью  $20

Вес рюкзака B= 100 кг
Входной массив  T <= [(10, 1), (170, 100), (50, 40), (40, 20)]
Отсортированный T => [(10, 1), (40, 20), (170, 100), (50, 40)]
  Берем предмет: <= ($10 , 1 кг)
  Берем предмет: <= ($40 , 20 кг)
  Берем предмет: <= ($50 , 40 кг)
Cg=$100 или Cmax=$170 ?
Набран рюкзак стоимостью  $170
```

Выбор f_{lb}:«*KnapsackGreedy*»

Теорема

Алгоритм «KnapsackFPTAS$_{KnapsackGreedy}$» имеет сложность $O\left(\frac{n^2}{\varepsilon}\right)$.

Выбор f_{lb}:«*KnapsackGreedy*»

Теорема

Алгоритм «KnapsackFPTAS$_{KnapsackGreedy}$» имеет сложность $O\left(\frac{n^2}{\varepsilon}\right)$.

Доказательство.

Используя $f' \leq f^* \leq 2f_G$:

$$O\left(\frac{nf'}{scale}\right) = O\left(\frac{n \cdot f'}{\frac{\varepsilon \cdot f_G}{n(1+\varepsilon)}}\right) \leq O\left(\frac{2n^2(1+\varepsilon)}{\varepsilon}\right) = O\left(\frac{n^2}{\varepsilon}\right).$$

□

Трассировка алгоритма «KnapsackFPTAS»

Используются нижние оценки «MaxItemCost» и «KnapsackGreedy».

D = [$\frac{134}{16}$, $\frac{789}{250}$, $\frac{56}{43}$, $\frac{345}{333}$, $\frac{4567}{857}$, $\frac{555}{47}$] B = 1000

Optimal Knapsack: $\frac{5312}{963}$ costs 5312

---

lower_bound = `max_item_cost` $\Rightarrow f_{lb}$= 4567 ε = 0.1 $\Rightarrow$ scale = 69.196969697

Ds = [$\frac{1}{16}$, $\frac{11}{250}$, $\frac{0}{43}$, $\frac{4}{333}$, $\frac{66}{857}$, $\frac{8}{47}$]

Approx. knapsack: $\frac{75}{920}$ costs 5256

---

lower_bound = `knapsack_greedy` $\Rightarrow f_{lb}$= 5312 ε = 0.1 $\Rightarrow$ scale = 80.4848484848

Ds = [$\frac{6}{47}$, $\frac{1}{16}$, $\frac{56}{857}$, $\frac{9}{250}$, $\frac{0}{43}$, $\frac{4}{333}$]

Approx. knapsack: $\frac{63}{920}$ costs 5256

---

lower_bound = `knapsack_greedy` $\Rightarrow f_{lb}$= 5312 ε = 0.06 $\Rightarrow$ scale = 50.1132075472

Ds = [$\frac{11}{47}$, $\frac{2}{16}$, $\frac{91}{857}$, $\frac{15}{250}$, $\frac{1}{43}$, $\frac{6}{333}$]

Approx. knapsack: $\frac{105}{963}$ costs 5312

Проблемы

- Очень странная неинтерактивная модель. Противодействие взломщику honeypotами — это интерактивный процесс, обычно там теория игр. $\Rightarrow$ т.е. тут очевидно, что нужны дешевые тормозящие узлы и сервисы, раз взломщик обязан их просканировать.
- Даже в рамках этой модели, очень странно (хотя и удобно), что есть отдельные ограничения по типам ЛИС.
- Странно, что ограничения по ресурсам (стоимостям — одномерные), обычно — отдельно память, отдельно ядра процессора и т.п. (тогда рюкзака не будет)