

**Московский технологический университет
(МИРЭА/МГУПИ)**

Физико-технологический институт

*Кафедра аппаратного, программного и математического обеспечения
вычислительных систем*

**Курсовая работа по дисциплине “Теория систем”
Задача оптимальной закупки оргтехники**

Группа: ТМБО-01-15
Студент: И. И. Аметов
Преподаватель: Д. И. Коган

Москва, 2016

Содержание

Введение	2
Постановка задачи	3
Алгоритм программы	8
Текст программы	10
Литература	12

Введение

Мы живём в век продолжающегося бурного развития компьютеров и компьютерной техники. Появление компьютеров и глобальной сети интернет упростило и ускорило поиск информации. Если раньше в поисках нужной информации приходилось обращаться в библиотеки, выписывать журналы, искать книги, изучать архивы, то теперь можно буквально за несколько минут отыскать нужную информацию в компьютерной сети. Причём информация в глобальной сети представлена практически на всех языках и со всех стран.

Помимо информации глобальная сеть дала возможность выполнять работу удалённо, находясь в любом месте, где есть доступ к интернету. Всё больше развивается торговля через интернет. Сейчас трудно найти какую-либо компанию или фирму, у которой нет своей страницы в интернете. Развивается сфера оказания услуг через интернет. Причём услуги не только частного рода, но и государственные услуги: записаться на приём к врачу, оформить замену паспорта, оплатить коммунальные услуги и многое другое. Интернет становится неотъемлемой частью нашей жизни.

Казалось бы, с приходом интернета человек мог бы отказаться от бумаги и полностью перейти к использованию электронных форм документооборота. Отчасти это так, но лишь отчасти. Полностью отказаться от бумажного документа не получается. И если для общения нам сейчас вполне достаточно сервисов интернета: электронной почты, служб мгновенной доставки сообщений, звуковых и видеозвонков, то в сфере бюрократической (в хорошем смысле этого слова) жизни без бумажных документов зачастую не обойтись. Примеры: доверенности, накладные, реестры, домовые книги и многое другое. Как следствие, у нас остаётся потребность в печатной технике: принтерах, копирах и сканерах.

На рынке оргтехники представлены самые разнообразные марки, модели и модификации множества различных принтеров. И у каждой модели есть свои нюансы: во-первых, цена — неотъемлемая часть товарно-денежных отношений, во-вторых, цена расходников, в-третьих, количество листов печатаемых на одной заправке тонера (порошок для печати). Тут же возникает вопрос: как выбрать несколько принтеров таким образом, что бы, с одной стороны, затратить как можно меньше средств на покупку, а с другой стороны уменьшить расходы на печать каждого листа бумаги?

С учётом того, что потребности в принтерах есть практически в каждой организации, то эта задача весьма актуальна. Осуществив оптимальную закупку принтеров можно добиться снижения расходов на печать, что в свою очередь положительно скажется на бюджете организации. Помимо покупки принтеров полученные результаты можно адаптировать и к другим областям. Пример: закупка станков для завода. У каждого станка есть своя цена, помимо цены есть и расходные материалы: шлифовальные ленты для шлифовальных станков, фрезы для фрезерных станков и другие.

Постановка задачи

Задан некоторый набор из n принтеров, условно обозначим этот набор через $A = \{a_1, a_2, \dots, a_n\}$ у каждого принтера есть своя цена $C = \{c_1, c_2, \dots, c_n\}$. Причём функция отображающая цену принтера $C(x_i) = c_i$ взаимно однозначна. То есть, каждый конкретный принтер x_i однозначно и единственным образом привязан к c_i . У каждого принтера есть своя цена картриджа (устройства, содержащего краску для печати) $K = \{k_1, k_2, \dots, k_n\}$, цена картриджа также взаимно и однозначно соотносится с множеством X . Ресурс листов печатаемых каждым принтером на новом картридже обозначим через множество $R = \{r_1, r_2, \dots, r_n\}$, оно также взаимно однозначно связано с множеством X . Общую денежную сумму выделенную на приобретение принтеров обозначим через W . Осуществить покупку принтеров свыше этой суммы невозможно. Задача заключается в том, чтобы приобрести на сумму W такой набор принтеров, чтобы минимизировать расходы краски на распечатку.

Введём удельную стоимость печати одного листа $U = \{u_1, u_2, \dots, u_n\}$, где

$$u_i = \frac{k_i}{r_i}$$

То есть, мы разделили стоимость одного картриджа на количество распечатываемых страниц.

Тогда можно ввести условную “стоимость” $P = p_1, p_2, \dots, p_n$ для каждого принтера, причём наименее ценным принтером будет принтер с максимальной удельной стоимостью, а наиболее ценным — принтер с наименьшей удельной стоимостью. Стоимость принтера с максимальной удельной стоимостью обозначим как 1, следующий за ним принтер обозначим как 2 и так далее. И здесь мы приходим к задаче о ранце.

Простейшая задача о ранце

Простейшая задача о ранце (англ. Knapsack problem) — дано N предметов, n_i предмет имеет массу $c_i > 0$ и стоимость $p_i > 0$. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины W (ёмкость рюкзака), а суммарная стоимость была максимальна.

Добавим ещё одно множество $X = \{x_1, x_2, \dots, x_n\}$ определяющее является ли предмет i выбранным или нет, причём

$$x_i = \begin{cases} 1, & \text{если предмет } i \text{ включается в набор} \\ 0, & \text{если предмет } i \text{ не включается в набор} \end{cases}$$

Тогда математически задачу о ранце можно записать так:

$$\begin{cases} c_1x_1 + c_2x_2 + \dots + c_nx_n \rightarrow \max \\ p_1x_1 + p_2x_2 + \dots + p_nx_n \leq W \\ x_i \in \{0, 1\}, i = \overline{1, n} \end{cases}$$

Метод динамического программирования

Пусть $Q(k, s)$ — максимальная стоимость предметов, которые можно уложить в ранец вместимости s , если можно использовать только первые k предметов, то есть $\{a_1, a_2, \dots, a_k\}$, назовём этот набор допустимыми предметами для $Q(k, s)$. Причём,

$$Q(k, 0) = 0$$

$$Q(0, s) = 0$$

Найдём $Q(k, s)$. Здесь возможны два варианта:

1. Если предмет k не попал в ранец. Тогда $Q(k, s)$ равно максимальной стоимости ранца с такой же вместимостью и набором допустимых предметов $\{a_1, a_2, \dots, a_{k-1}\}$, то есть, $Q(k, s) = Q(k-1, s)$.
2. Если предмет k попал в ранец. Тогда $Q(k, s)$ равно максимальной стоимости ранца, где вес s уменьшаем на вес k -ого предмета и набор допустимых предметов $\{a_1, a_2, \dots, a_{k-1}\}$ плюс стоимость k , то есть $Q(k-1, s - p_k) + c_k$.

$$\text{То есть: } Q(k, s) = \max \{Q(k-1, s), Q(k-1, s - p_k) + c_k\}$$

Стоимость искомого набора равна $Q(n, W)$, так как нужно найти максимальную стоимость рюкзака, где все предметы допустимы и вместимость рюкзака W .

Определение вхождения предмета в набор

Будем определять, входит ли a_i предмет в искомый набор. Начинаем с элемента $Q(i, w)$, где $i = n, w = W$. Для этого сравниваем $Q(i, w)$ со следующими значениями:

1. Максимальная стоимость ранца с такой же вместимостью и набором допустимых предметов $\{a_1, a_2, \dots, a_{i-1}\}$, то есть $Q(i-1, w)$.
2. Максимальная стоимость ранца с вместимостью на w_i меньше и набором допустимых предметов $\{a_1, a_2, \dots, a_{i-1}\}$ плюс стоимость c_i , то есть $Q(i-1, w - w_i) + c_i$.

Максимальное значение записывается в $Q(i, w)$. Если решение записать в виде таблицы, где по столбцам записана вместимость ранца, а по строкам записаны предметы, доступные для размещения в рюкзаке, то можно записать формулу заполнения $i+1$ строки по i -й строке:

$$Q(i+1, p) = \begin{cases} Q(i, p), & \text{если } p_{i+1} > w \\ \max \{Q(i, p); c_{i+1} + Q(i, w - p_{i+1})\} \end{cases}$$

Пример:

Решим следующую задачу:

$$\begin{cases} 2x_0 + x_1 + 3x_2 + 4x_3 \rightarrow \max \\ 5x_0 + 3x_1 + 4x_2 + 2x_3 \leq 12 \end{cases}$$

Решение будет в виде таблицы:

	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	2 ₀	2 ₀	2 ₀	2 ₀	2 ₀	2 ₀	2 ₀	2 ₀
1	0	0	1 ₁	1 ₁	2 ₀	2 ₀	2 ₀	3 _{1,0}	3 _{1,0}	3 _{1,0}	3 _{1,0}	3 _{1,0}
2	0	0	1 ₁	3 ₂	3 ₂	3 ₂	4 _{2,1}	4 _{2,1}	5 _{2,0}	5 _{2,0}	5 _{2,0}	6 _{2,1,0}
3	0	4 ₃	4 ₃	4 ₃	5 _{3,1}	7 _{3,2}	7 _{3,2}	7 _{3,2}	8 _{3,2,1}	8 _{3,2,1}	9 _{3,2,0}	9 _{3,2,0}

Ответом для этой задачи будет $Q(3, 12) = 9_{3,2,0}$, то есть, максимально возможная стоимость составляет 9 и достигается выбором третьего, второго и нулевого предметов ($4 + 3 + 2 = 9$). При этом общий вес предметов составляет $2 + 4 + 5 = 11$.

Бикритериальная задача о ранце

Зачастую случается так, что помимо одного критерия приходится учитывать ещё и другие критерии, например: кроме удельной стоимости печати одного листа у принтеров есть ещё и энергопотребление или выделение озона при печати. Ясно, что чем меньше энергопотребление, тем меньше накладные расходы или чем меньше озона выделяется при печати, тем меньше вреда здоровью пользователей этого принтера.

Энергопотребление для нашей задачи в математическом виде можно описать аналогично удельной стоимости, то есть, принтеру с максимальным энергопотреблением мы присвоим “стоимость” в 1, следующему за этим принтером — 2 и так далее, принтеру с минимальным энергопотреблением в результате будет присвоена максимальная стоимость — n . Математически новую задачу можно описать так:

$$\begin{cases} c_1^1 x_1 + c_2^1 x_2 + \dots + c_n^1 x_n \rightarrow \max \\ c_1^2 x_1 + c_2^2 x_2 + \dots + c_n^2 x_n \rightarrow \max \\ w_1 x_1 + w_2 x_2 + \dots + w_n x_n \leq W \end{cases}$$

где $\{c_1^1, c_2^1, \dots, c_n^1\}$ — показатели по первому критерию (в нашем случае это удельная стоимость печати одного листа), $\{c_1^2, c_2^2, \dots, c_n^2\}$ — показатели по второму критерию (энергопотребление) и $\{w_1, w_2, \dots, w_n\}$ — стоимость принтера.

Теперь для каждого решения вводится совокупность эффективных оценок $E(Q)$, где Q — какое-либо решение. Пример: $Q(k, p)$ — это решение, в котором можно использовать предметы от 1 до k , p — максимальный допустимый вес ($p = 1, \dots, W$). Отсюда система эффективных оценок для $E(Q) = E(k, p)$. Если же $k = n$, а $p = W$, то мы получим эффективную оценку для всей задачи.

Система эффективных оценок строится на основе рекуррентного соотношения и начинается с $E(1, p)$ (1-й предмет, p — максимальный вес).

$$E(1, p) = \begin{cases} (0, 0), \text{ если } w_1 > p \\ (C_1^1, C_1^2), \text{ если } w_1 \leq p \end{cases}$$

так заполняется первая строка. После заполнения k -й строки переходим к последней строке:

$$E(k+1, p) = \begin{cases} E(k, p), \text{ если } w_{k+1} > p \\ \text{eff}\{E(k, p) \cup [(c_{k+1}^1, c_{k+1}^2) \oplus E(k, p - w_{k+1})]\} \end{cases}$$

Пример:

Бикритериальная задача задана следующим образом:

$$\begin{cases} 5x_1 + 6x_2 + 2x_3 + 3x_4 + x_5 \rightarrow \max \\ 4x_1 + 3x_2 + 5x_3 + 4x_4 + 3x_5 \rightarrow \max \\ 3x_1 + 2x_2 + 4x_3 + 2x_4 + x_5 \leq 8 \end{cases}$$

Решим её:

	1	2	3	4	5	6	7	8
1	0	0	$(5, 4)_1$	$(5, 4)_1$	$(5, 4)_1$	$(5, 4)_1$	$(5, 4)_1$	$(5, 4)_1$
2	0	$(6, 3)_2$	$(5, 4)_1$ $(6, 3)_2$	$(5, 4)_1$ $(6, 3)_2$	$(11, 7)_{1,2}$	$(11, 7)_{1,2}$	$(11, 7)_{1,2}$	$(11, 7)_{1,2}$
3	0	$(6, 3)_2$	$(5, 4)_1$ $(6, 3)_2$	$(5, 4)_1$ $(6, 3)_2$ $(2, 5)_3$	$(11, 7)_{1,2}$	$(11, 7)_{1,2}$ $(8, 8)_{3,2}$	$(11, 7)_{1,2}$ $(7, 9)_{3,1}$ $(8, 8)_{3,2}$	$(11, 7)_{1,2}$ $(7, 9)_{3,1}$ $(8, 8)_{3,2}$
4	0	$(3, 4)_4$ $(6, 3)_2$	$(5, 4)_4$ $(6, 3)_2$	$(9, 7)_{4,2}$	$(11, 7)_{1,2}$ $(8, 8)_{1,4}$	$(8, 8)_{1,4}$ $(5, 9)_{3,4}$ $(11, 7)_{1,2}$	$(14, 11)_{1,2,4}$	$(14, 11)_{1,2,4}$ $(11, 12)_{3,2,4}$
5	$(1, 3)_5$	$(3, 4)_4$ $(6, 3)_2$	$(4, 7)_{4,5}$ $(7, 6)_{2,5}$	$(9, 7)_{4,2}$	$(11, 7)_{1,2}$ $(10, 10)_{4,2,5}$	$(12, 10)_{1,2,5}$ $(9, 11)_{5,4,1}$	$(14, 11)_{1,2,4}$ $(6, 12)_{3,4,5}$	$(15, 14)_{1,2,4,5}$

Здесь ответом является $(15, 14)$ и достигается этот результат набором из первого, второго, четвёртого и пятого предметов.

Методы компромисса

Из последней задачи видно, что иногда возможны несколько ответов. В этом случае лицо принимающее решения должно выбрать из всей совокупности решений наиболее подходящее для него решение. Существует много методов поиска компромисса. Вот некоторые из них:

Лексикографическое упорядочение критериев

Критерии, по которым производится отбор, упорядочиваются по убыванию их важности. Если в процессе решения задачи получился только один ответ, то алгоритм отбора на этом заканчивается. Если же в процессе решения возникло несколько возможных решений, то

в этом случае решения упорядочивают по самому важному критерию. Если решение удовлетворяющее самому важному критерию оказалось единственным, то алгоритм заканчивается, если получилось так, что есть несколько решений одинаковых по самому важному критерию, то в этом случае обращают внимание на второй по важности критерий. Если опять осталось несколько решений, то переходят к третьему критерию и так далее.

Метод идеальной точки

Пусть D — множество допустимых решений некоторой задачи. Решение x принадлежащее D обладает несколькими критериями: $K_1(x)$, $K_2(x)$, ..., $K_l(x)$. В этом случае можно ввести некоторое l мерное пространство. Тогда вся совокупность решений даёт точки в пространстве формирующие область T . Далее в пространстве назначается идеальная точка J не принадлежащая области T . В результате остаётся найти ближайшую к J точку.

Алгоритм программы

Главная функция, с которой начинается исполнение программы — это функция (`main`).

В ней осуществляется ввод начальных данных с клавиатуры. Запрашивается количество принтеров доступных для покупки. После чего вводятся следующие данные по каждому отдельно взятому принтеру: цена принтера, стоимость картриджа, количество распечатываемых листов на одном картридже и критерий энергопотребления (чем значение критерия, тем меньше потребляет принтер). После заполнения сведений о принтерах вводится бюджет на закупку оргтехники.

После этого данные передаются в функцию `knap sack-bicrit`. В этой функции реализована рекурсивная схема работы: программа вызывает саму себя для того, чтобы получить сведения о:

1. Возможных решениях при том же уровне бюджета, но на единицу уменьшенным количеством принтеров.
2. Если цена последнего введённого принтера меньше, либо равна бюджету, то ищется решение для бюджета уменьшенного на стоимость последнего принтера и уменьшенного на единицу числа принтеров.

Из полученных двух решений выбирается максимальное решение.

Пример работы программы:

```
> (main)
Введите количество принтеров: 5
Введите цену принтера N 0: 3
Введите цену картриджа принтера N 0: 2
Введите количество печатаемых листов принтера N 0: 10
Введите критерий энергопотребления принтера N 0: 4
Введите цену принтера N 1: 2
Введите цену картриджа принтера N 1: 1
Введите количество печатаемых листов принтера N 1: 10
Введите критерий энергопотребления принтера N 1: 3
Введите цену принтера N 2: 4
Введите цену картриджа принтера N 2: 4
Введите количество печатаемых листов принтера N 2: 10
Введите критерий энергопотребления принтера N 2: 5
Введите цену принтера N 3: 2
Введите цену картриджа принтера N 3: 3
Введите количество печатаемых листов принтера N 3: 10
Введите критерий энергопотребления принтера N 3: 4
Введите цену принтера N 4: 1
Введите цену картриджа принтера N 4: 5
Введите количество печатаемых листов принтера N 4: 10
Введите критерий энергопотребления принтера N 4: 3
Введите бюджет на закупку: 8
```

'((2 9/10 14 4 3 1 0))

Здесь оценка по первому критерию оказалась равной $2\frac{9}{10}$, по второму критерию: 14, а набор принтеров: {5, 4, 2, 1} (при отсчёте от единицы).

Текст программы

Listing 1: cw.rkt

```
1 #lang racket
2
3 (require math/array)
4
5 (define (backpack costs weights maxweight item)
6   (cond
7     [(= item 0) (if (>= maxweight (array-ref weights (vector item)))
8                     (cons (array-ref costs (vector item)) (list item))
9                     (list 0))]
10    [(= maxweight 0) (list 0)]
11    [else (let ([old-cost (backpack costs weights maxweight (- item 1))])
12             [new-cost (if (>= maxweight (array-ref weights (vector item)))
13                           (cons (+ (array-ref costs (vector item))
14                                     (car (backpack costs weights
15                                           (- maxweight
16                                             (array-ref weights
17                                               (vector item)))
18                                           (- item 1))))
19                           (cons item (cdr (backpack costs weights
20                                           (- maxweight
21                                             (array-ref weights
22                                               (vector item)))
23                                           (- item 1))))))
24             (list 0))])
25             (if (> (car old-cost) (car new-cost))
26                 old-cost
27                 new-cost)))]))
28
29 (define (gte list1 list2)
30   ; Больше либо равно
31   (if (and (>= (car list1) (car list2))
32         (>= (car (cdr list1)) (car (cdr list2))))
33       true
34       false))
35
36 (define (present-greater? el lst)
37   ; Есть ли в lst элемент больший чем el
38   (cond
39     [(eq? lst '()) false]
40     [(gte (car lst) el) true]
41     [else (present-greater? el (cdr lst))])
42
43 (define (select-best-iter list1 list2 result)
44   ; Создание списка с наилучшими вариантами
45   (cond
46     [(and (eq? list1 '())
47           (eq? list2 '())) result]
48     [(eq? list1 '())
49      (let ([element (car list2)]
50            [new-list (cdr list2)])
51        (if (present-greater? element result)
52            (select-best-iter list1 new-list result)
53            (select-best-iter list1 new-list (cons element result)))))]
54     [else
55      (let ([element (car list1)]
56            [new-list (cdr list1)])
57        (if (present-greater? element list2)
58            (select-best-iter new-list list2 result)
59            (select-best-iter new-list list2 (cons element result)))))))]
60
61 (define (select-best list1 list2)
62   (select-best-iter list1 list2 '()))
63
64 (define (add-solution-iter sol-list cost1 cost2 item result)
65   (cond
66     [(eq? sol-list '()) result]
67     [else
68      (let ([solution (car sol-list)])
69        (add-solution-iter (cdr sol-list) cost1 cost2 item
70                          (cons
```

```

71         (cons (+ (car solution)
72                 cost1)
73               (cons (+ (cadr solution)
74                       cost2)
75                     (cons item (cddr solution))))
76               result))))))
77
78 (define (add-solution sol-list cost1 cost2 item)
79   (add-solution-iter sol-list cost1 cost2 item '()))
80
81 (define (knapsack-bicrit costs1 costs2 weights maxweight item)
82   (define (get-arr sa ind)
83     (array-ref sa (vector ind)))
84   (define (get-weight ind)
85     (get-arr weights ind))
86   (define (get-cost1 ind)
87     (get-arr costs1 ind))
88   (define (get-cost2 ind)
89     (get-arr costs2 ind))
90   (cond
91     [(= item 0)
92      (list (if (>= maxweight (get-weight item))
93                (list (get-cost1 item) (get-cost2 item) item)
94                (list 0 0)))]
95     [(= maxweight 0)
96      (list (list 0 0))]
97     [(>= maxweight (get-weight item))
98      (let ([old-sol (knapsack-bicrit costs1 costs2 weights maxweight (- item 1))]
99            [new-sol (add-solution (knapsack-bicrit costs1
100                                         costs2
101                                         weights
102                                         (- maxweight (get-weight item))
103                                         (- item 1))
104                                   (get-cost1 item)
105                                   (get-cost2 item)
106                                   item)])
107        (select-best old-sol new-sol))]
108     [else (knapsack-bicrit costs1 costs2 weights maxweight (- item 1))]))
109
110 (define (main)
111   (let ([n 0]
112         [prices '()]
113         [c1 '()]
114         [tp 0]
115         [tl 0]
116         [c2 '()])
117     (printf "Введите количество принтеров: ")
118     (set! n (read))
119     (for ([i n])
120       (printf "Введите цену принтера N ~a: " i)
121       (set! prices (cons (read) prices))
122       (printf "Введите цену картриджа принтера N ~a: " i)
123       (set! tp (read))
124       (printf "Введите количество печатаемых листов принтера N ~a: " i)
125       (set! tl (read))
126       (set! c1 (cons (- 1 (/ tp tl)) c1))
127       (printf "Введите критерий энергопотребления принтера N ~a: " i)
128       (set! c2 (cons (read) c2)))
129     (set! prices (reverse prices))
130     (set! c1 (reverse c1))
131     (set! c2 (reverse c2))
132     (printf "Введите бюджет на закупку: ")
133     (knapsack-bicrit (list->array c1) (list->array c2) (list->array prices) (read) (- n 1))))

```

Литература

В процессе написания курсовой работы использовалась следующая литература:

1. Лекции по теории систем. Коган Д.И.
2. Структура и интерпретация компьютерных программ. Харольд Абельсон, Джеральд Джей Сассман, Джули Сассман. Добросвет. 2006.