§10. Дискретно оптимиране

§10.1. Постановка на задачата и оптимизационни модели

Ако разгледаме оптимизационните задачи от леко философски ъгъл, ясно ще забележим, че когато обектът на изследване е природно явление (т.е. има естествен произход), то задачите са непрекъснати. Когато оптимизираме структури, създадени от човека, почти винаги възниква дискретност. Това е резултат от много фактори – стандиртизация и унификация, серийност в производството и управлението, създаване на големи структури, в които процесите са прекъснати и квантувани и като такива си взаимодействат – транспортни, комуникации, различни мрежови образувания и т.н. и т.н.

В тези случаи възникват т.нар. дискретни (в частност целочислени) оптимизационни задачи, чийто най-общ математически модел е:

$$\max / \min \{f(x) : \ x \in D\},$$

където D е крайно или изброимо множество.

Най-често дискретността е върху отделните променливи:

$$\max \{f(x_1, \ldots, x_n) : \ g_i(x_1, \ldots, x_n) \leqslant b_i, \ i{=}1, \ldots, m, \ x_j \in D_j, \ j{=}1, \ldots, n\}$$

и ако f, g_i, $i = 1, \ldots, m$, са линейни функции на n променливи, а D_j, $j = 1, \ldots, n$, са множества от цели числа получаваме т.нар. линейна целочислена оптимизационна задача

$$\max \left\{ \mathbf{c}^{\mathrm{T}} \mathbf{x} : \ \mathbf{a}_i^{\mathrm{T}} \mathbf{x} \leqslant b_i, \ i = 1, \ldots, m, \ 0 \leqslant x_j \leqslant d_j \ - \text{цели}, \ j = 1, \ldots, n \right\}.$$

Естествено (и най-често) част от променливите приемат непрекъснати стойности – тогава говорим за смесено-целочислена задача. Тя възниква най-често при оптимизация на икономически и производствени процеси, но много интересни са и редица екзотични и класически модели, които (както по-късно ще видим) са важни и от теоретична гледна точка.

Задача за раницата

$$\max \left\{ \sum_{j=1}^{n} c_j x_j : \ \sum_{j=1}^{n} a_j x_j \leqslant b, \ 0 \leqslant x_j - \text{цели}, j = 1, \ldots, n \right\}$$

носи името си от желанието на крадеца да максимизира печалбата си като реши кои предмети с цени c_j и обем a_j да открадне, а разполага с раница с обем b. При горната формулировка говорим за целочислена задача за раницата. Ако $x_j \in \{0, 1\}$ – за двоична задача. Ако имаме повече ограничения – по тегло на предметите и т.н. – за многомерна задача за раницата.

Покритие на граф. Нека $G(V,E)$ е граф с върхове $v_i \in V$, $i = 1,\dots n$ и ребра $e_j \in E, j = 1,\dots,m$ и матрица на инцидентност с елементи

$$a_{ij} = \begin{cases} 1, \text{ ако реброто } j \text{ е инцидентно с върха } i, \\ 0, \text{ ако реброто } j \text{ не е инцидентно с върха } i. \end{cases}$$

Намирането на най-малкия брой ребра (минималното покритие), такива че всеки връх да е инцидентен с поне едно от тях, може да се формулира така:

$$\min\left\{\sum_{j=1}^{m} x_j : \ \sum_{j=1}^{m} a_{ij}x_j \geqslant 1, \ i = 1,\dots,n, \ x_j \in \{0,1\}\right\}.$$

Тогава тези x_j, които в решението ще имат стойност 1 образуват търсеното покритие.

Задача за четирите цвята. След откритието на Гутенберг започват да се печатат и политически карти – всяка държава се оцветява с отделен цвят. Веднага забелязали, че каквато и да е картата четири цвята са достатъчни, за да няма съседни държави, оцветени с еднакви цветове. Този факт бе доказан едва в наше време, като доказателството му и досега не е напълно прието като чисто математическо, защото при него се използва компютърна проверка на варианти.

Нека политическата карта представим като граф $G(X,A)$, където върховете $x_i \in X$, $i = 1,\dots n$ на графа са държавите, а $A = \{a_{ij}\}$ е матрицата на съседство на графа, т.е. $a_{ij} = 1$, ако x_i и x_j са свързани с ребро (т.е. държавите x_i и x_j имат обща граница) за $i,j = 1,\dots n$. Въвеждаме променливите

$$z_{ik} = \begin{cases} 0, \text{ върхът } x_i \text{ не е оцветен с цвета } k, \\ 1, \text{ върхът } x_i \text{ е оцветен с цвета } k. \end{cases}$$

Тогава ограниченията са

$$\sum_{k=1}^{t} z_{ik} = 1, \ i = 1,\dots,n \quad \text{– единствен цвят за връх,}$$

$$L(1 - z_{ik}) - \sum_{j=1}^{n} a_{ij}z_{kj} \geqslant 0, \quad i = 1,\dots,n, \ k = 1\dots,t,$$

където L е число по-голямо от броя на съседните върхове на връх с най-голям брой такива, т.е. $L > \max\limits_{i} \sum\limits_{j} a_{ij}$.

При тези ограничения търсим

$$\min \sum_{k=1}^{t} n^k \sum_{i=1}^{n} z_{ik}.$$

Трябва да отбележим, че политическата карта поражда плосък граф, т.е. ребрата не се пресичат. В противен случай твърдението за четирите цвята не е вярно.

Задача за търговския пътник. Имаме пълен граф с n върха и c_{ij}, $i,j = 1,\dots,n$ е разстоянието между връх i и връх j. Търси се хамилтонов цикъл в графа, който има минимална сума от разстоянията на ребрата си. С други думи, търговският пътник тръгва от връх 1 и преминавайки през всеки връх по веднъж трябва да се върне в 1 като при това измине най-кратък път. Въвеждаме

$$x_{ij} = \begin{cases} 1, & \text{от връх } i \text{ отива във връх } j, \\ 0, & \text{от връх } i \text{ не отива във връх } j. \end{cases}$$

Търсим

$$\min_x \sum_{i,j=1}^{n} c_{ij}x_{ij}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,\dots,n \text{ във всеки връх се влиза веднъж,}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,\dots,n \text{ от всеки връх се излиза веднъж.}$$

Дотук ограниченията не са достатъчни, за да гарантират, че пътят на търговския пътник ще представлява единствен цикъл – той може да се състои от няколко несвързани цикли. За да избегнем това въвеждаме още n цели променливи $u_i \geqslant 0$, $i = 1,\dots,n$ и ограничения

$$u_i - u_j + nx_{ij} \leqslant n-1, \quad i,j = 2,\dots,n.$$

Ако има подцикъл, несъдържащ върха 1, то като съберем поредните ограничения по неговите ребра u_i и u_j ще се унищожат и ще стигнем до противоречие $n \leqslant n-1$, така че не може в решението да има цикъл неминаващ през върха 1. За цикъл през връх 1 избираме последователно стойности за u_i и u_j по него ($u_i - u_j = -1$) и след като имаме предвид, че върхът 1 не влиза в тези ограничения не стигаме до противоречие.

Задача за оптимален разкрой. Нека разполагаме с метални пръти с дължини $b_1,\dots,b_k$ и от тях трябва да се отрежат елементи с дължини $a_1,\dots a_p$ съответно $n_1,\dots n_p$ броя. Въвеждаме променливите x_{ij} – брой елементи от вид $i = 1,\dots,p$, които ще отрежем от прът $j = 1,\dots,k$ и $y_j \in \{0,1\}$, $j = 1,\dots,k$.

Търсим

$$\min_y \sum_{j=1}^{k} y_j$$

$$\sum_{i=1}^{p} a_i x_{ij} \leqslant y_j l_j, \quad j = 1,\dots,k,$$

$$\sum_{j=1}^{k} x_{ij} = n_i, \quad i = 1,\dots,p.$$

Първата група ограничения осигурява, че ще режем от минимален брой пръти (т.е. остатъкът ще се събере в цели пръти) и няма да режем повече от дължината на пръта. Втората група гарантира броя на елементите.

Този модел за първи път е формулиран от руския математик акад. Лев Канторович през 1937-38 г. Той предлага и метод за решаване – вариант на симплекс метода. По-късно симплекс методът е разработен от Дж. Данциг. През 80-те години двамата получават Нобелова награда за икономика.

Оптимизация на маршрути. До един събирателен център S (завод, град) трябва да се превозват пътници от няколко места, като всички те са свързани с достатъчно гъста мрежа от пътища. Нека a_i, $i = 1, \ldots, n$ е броят пътници от пункт i до S, а b е броят места в автобуса. Един от възможните начини за построяване на оптимизационен модел се заключава в определяне на множество от възможни (допустими) маршрути, което има на няколко порядъка по-голям брой маршрути от действително необходимите и да се оптимизира в това множество. Отделен въпрос е какво значи „допустим“. Например, психологическо изследване е установило, че когато човек пътува до работното си място повече от 45 мин. той рязко се демотивира и т.н.

Нека x_{ij} е броят на пътниците от пункт $i = 1, \ldots, n$ по маршрут $j = 1, \ldots, m$. Нека Y_i е индексното множество на всички маршрути, минаващи през пункт i, а I_j е индексното множество на всички пунктове по маршрут j; $y_j \in \{0, 1\},\ j = 1, \ldots, m$.

Ограниченията $\sum_{j \in I_j} x_{ij} = a_i, i = 1, \ldots, n$ гарантират, че маршрутите праз пункт i ще поемат a_i пътници, а $\sum_{i \in I_j} x_{ij} \leqslant y_j b$, $j = 1, \ldots, m$ гарантират, че ако маршрутът се реализира ($y_j = 1$) автобусът няма да се препълни. Търсим $\min \sum y_j$, което сочи, че ще използваме минимален брой автобуси.

В случай, че се разполага с няколко вида автобуси (да речем два) $b_1 < b$ и искаме при непълен голям автобус да използваме по-малкия, правим следното: въвеждаме $\overline{y}_j \in \{0, 1\}$, $j = 1, \ldots m$ и във ограниченията неравенства променяме десните страни на $\leqslant y_j b + \overline{y}_j b_1$ и добавяме ограничения $y_j + \overline{y}_j \leqslant 1$, $j = 1, \ldots m$ като търсим $\min \sum_j (y_j + c\overline{y}_j)$, $0 < c < 1$.

Доказва се, че матрицата на ограниченията притежава свойството „унимодулярност“, което гарантира, че при цели числа от дясно върховете на многостена също са цели, така че изискването за целочисленост на x_{ij} може да отпадне. Забележете, че същото свойство притежава и класическата транспортна задача.

Разни хитрости. Задачи с фиксирани добавки се срещат в случаите, когато в целевата функция се добавя фиксирана стойност, в случай

че дадена променлива е различна от нула, независимо от стойността ѝ (най-честият пример е таксуването в таксита). В целевата функция това може да се отрази така:

$$\min\left\{\sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} d_j z_j\right\},$$

$0 \leqslant x_j \leqslant U z_j$, $j = 1, \ldots, n$, $z_j \in \{0, 1\}$, U – достатъчно голямо число.

Това гарантира, че когато $x_j > 0$, то $z_j = 1$ и фиксираната добавка d_j влиза в целевата функция, ако $x_j = 0$, то задължително $z_j = 0$ понеже търсим минимум, а и $d_j > 0$.

Различни логически ограничения водят до задачи от тип или-или. Например условието е, ако една от променливите е положетелна другата да е задължително 0. Тогава избираме достатъчно голямо число U и двоична променлива $z \in \{0, 1\}$ и ограничения

$$0 \leqslant x_1 \leqslant Uz$$

$$0 \leqslant x_2 \leqslant U(1 - z).$$

Ако целевата функция има вида

$$\min \sum_j c_j x_{j_1} x_{j_2} \ldots x_{j_{k_j}} \quad x \in \{0, 1\},$$

то можем да я заменим с линейна като за всяко събираемо от нея прибавим ограничение

$$\frac{x_{j_1} + x_{j_2} + \cdots + x_{j_k}}{k} = z_k + y_k, \quad z_k \in \{0, 1\}, \quad 0 \leqslant y_k \leqslant 1 - \frac{1}{k}$$

и новата целева функция е $\min \sum_j c_j z_j$.

Подобни примери постоянно възникват в практически задачи, водещи до прекъснати целеви функции, несвързани и неизпъкнали допустими множества и др.

§11. Метод на отсичащите равнини

Това е исторически първия алгоритъм за решаване на широк клас дискретни оптимизационни задачи. Разработен е от Гомори през 1957-58 год. като метод за решаване на линейните целочислени задачи. Най-общо идеята е следната: нека търсим

$$\max\left\{\mathbf{c}^{\mathrm{T}}\mathbf{x}:\ \mathbf{a}_i^{\mathrm{T}}\mathbf{x}=b_i,\ i=1,\dots,k,\ 0\leqslant x_j-\text{цели},\ j=1,\dots,n\right\}.$$

Знаем, че ограниченията $\mathbf{a}_i^{\mathrm{T}}\mathbf{x}=b_i$, $i=1,\dots,k$, $0\leqslant x_j$, $j=1,\dots,n$ (без целочисленост) представляват множеството от точки на изпъкнал многостен. В това множество са и точките с цели стойности Ω. Ако означим с Ω' изпъкналата обвивка на Ω, тя също ще представлява изпъкнал многостен и

$$\max\left\{\mathbf{c}^{\mathrm{T}}\mathbf{x}:\ \mathbf{x}\in\Omega'\right\}$$

представлява линейна оптимизационна задача, която можем да решим с някои от познатите ни методи. Оказва се, че намирането на изпъкналата обвивка е изключително сложна задача – по-сложна от изходната. Идеята на Гомори се състои в това, че не е необходимо да търсим цялата Ω', а само частта от нея „около“ непрекъснатото решение на линейната задача. Това се постига чрез последователни „отсичания“ от линейния многостен чрез т.нар. правилни отсичащи равнини. Това означава прибавяне на ново линейно ограничение, което не се удовлетворява от непрекъснатото решение (т.е. отсича го), но не отсича точка с цели координати. Всеки път се решава получената нова линейна задача и се предполага, че в края на процеса поредното решение ще се окаже целочислено, с което ще сме решили изходната задача.

Ще опишем (без доказателство) построяването на правилно отсичане. Нека

$$\sum_{j=1}^{n} a_j x_j = b \tag{11.1}$$

е едно от ограниченията на задачата в базисното представяне спрямо оптималния базис, за което b не е цяло – т.е. текущото решение не е целочислено. Означаваме

$$a_j=[a_j]+f_j,\quad b=[b]+f \quad \text{цяла част плюс дробна част.}$$

Понеже искаме x_j да имат цели стойности

$$\sum_j [a_j]x_j\leqslant b \quad\text{и}\quad \sum_j [a_j]x_j\leqslant [b]$$

и разликата между дясната и лявата част на последното неравенство е цяло число.

Добавяме нова целочислена променлива x

$$\sum_{j=1}^{n}[a_j]x_j + x = [b] \tag{11.2}$$

и след като от (11.2) извадим (11.1) получаваме търсеното правилно отсичане

$$\sum_{j}(-f_j)x_j + x = -f,$$

като x става базисна променлива в новата задача.

По-детайлно записано: нека $\mathbf{x}^* = (b_1, b_2, \ldots, b_k, 0, \ldots, 0)$ е текущото оптимално решение и нека b_k не е цяло число. Вземаме ограничението

$$x_k + \sum_{j=k+1}^{n} a_{kj}x_j = b_k$$

и го заместваме с „правилното“ отсичане

$$\sum_{j=k+1}^{n}(-f_{kj})x_j + x = -f_k,$$

като x става базисна променлива вместо x_k.

Пример.

$$\max\{21x_1 + 11x_2 :\ 7x_1 + 4x_2 + x_3 = 13,\ 0 \leqslant x_j\text{-цели},\ j = 1,2,3\}$$

записваме като

$$\max\ x_0$$

$$\left|\begin{array}{llll} x_0 & -21x_1 & -11x_2 & = 0 \\ & 7x_1 & + 4x_2 & +x_3 = 13. \end{array}\right.$$

След решаване имаме

$$\left|\begin{array}{llll} x_0 & & +x_2 & +3x_3 = 39 \\ & x_1 & +\frac{4}{7}x_2 & +\frac{1}{3}x_3 = 1\frac{6}{7}, \end{array}\right.$$

добавяме

$$-\frac{4}{7}x_2 - \frac{1}{3}x_3 + x_4 = -\frac{6}{7}$$

и решаваме

$$\left|\begin{array}{llll} x_0 & & +2\frac{3}{4}x_3 & +1\frac{3}{4}x_4 = 37\frac{1}{2} \\ & x_1 & & + x_4 = 1 \\ & & x_2 + \frac{1}{4}x_3 & -1\frac{3}{4}x_4 = 1\frac{1}{2}, \end{array}\right.$$

добавяме

$$-\frac{3}{4}x_3 + \frac{1}{4}x_4 + x_5 = -\frac{1}{2}$$

и пак решаваме

$$\left|\begin{array}{llllll} x_0 & +x_1 & & & +11x_5 & = 33 \\ & -x_1 & +x_3 & & - \ x_5 & = 1 \\ & 2x_1 & +x_2 & & + \ x_5 & = 3 \\ & x_1 & & +x_4 & & = 1, \end{array}\right.$$

за да получим окончателно целочисленото решение $\mathbf{x}^* = (0, 3, 1)$ и оптималната стойност на целевата функция $z(\mathbf{x}^*) = x_0 = 33$.

Този метод не можа да получи широко разпространение поради причини, които прозират и от нашето кратко описание – много дълъг изчислителен процес, при който неминуемо се губи точност, няма гаранция за сходимост на процеса в общия случай и т.н.

§12. Метод на разклоняване и граници

Става дума за един универсален подход за решаване на почти всички видове дискретни оптимизационни задачи. Идеята за създаването му възниква през 1963 год. след една забавна история, при която фирма публикува с рекламна цел задачата за търговския пътник без да знае решението и как се решават подобни задачи. След нарастващ скандал от страна на клиентите да получат наградата за това, че са изпратили верен отговор, фирмата се обръща към професионални математици, които решават въпросния пример и предлагат новия метод.

Нека търсим

$$\max_{\mathbf{x}} \{f(\mathbf{x}) : \ \mathbf{x} \in \Omega\},$$

където Ω е крайно дискретно множество, а върху функцията $f(\mathbf{x})$ не се налагат никакви ограничения. Нека Ω може да се разбие на краен брой непресичащи се подмножества

$$\Omega = \bigcup_i \Omega_i, \quad \Omega_k \bigcap \Omega_j = \emptyset.$$

Нека наричаме $\overline{\Omega}_i$, $\Omega_i \subset \overline{\Omega}_i$ разширение на Ω_i и въведем означенията

$$\max_{\mathbf{x}} \{f(\mathbf{x}) : \ \mathbf{x} \in \Omega_i\} \ - \text{задача } A_i;$$

$$\max_{\mathbf{x}} \left\{f(\mathbf{x}) : \ \mathbf{x} \in \overline{\Omega}_i\right\} \ - \text{задача } \overline{A}_i.$$

Съществено е, че $\max\limits_{\mathbf{x}} \left\{f(\mathbf{x}) : \ \mathbf{x} \in \overline{\Omega}_i\right\} \geqslant \max\limits_{\mathbf{x}} \{f(\mathbf{x}) : \ \mathbf{x} \in \Omega_i\}$, т.е. оптималната стойност на т.н. оценъчна задача $\overline{A}_i$ е по-голяма от тази на A_i. Това естествено произлиза от релацията между Ω_i и $\overline{\Omega}_i$.

Начинът за конструиране на оценъчната задача е специфичен за всеки вид дискретни задачи, но общото е, че тя се решава лесно, за разлика от изходната задача. Върху това какво значи „лесно“ ще философстваме по-късно.

Въвеждаме и следните означения:

R – рекорд; $R = \max\{f(\mathbf{x}^*), -\infty\}$, където $\mathbf{x}^*$ е допустимо решение на изходната задача A;

S – списък от задачи, получени при разбиването на A на $A_1, A_2, \ldots, A_k$, евентуално разбиването на A_j на $A_{j_1}, A_{j_2}, \ldots, A_{j_p}$ и т.н.

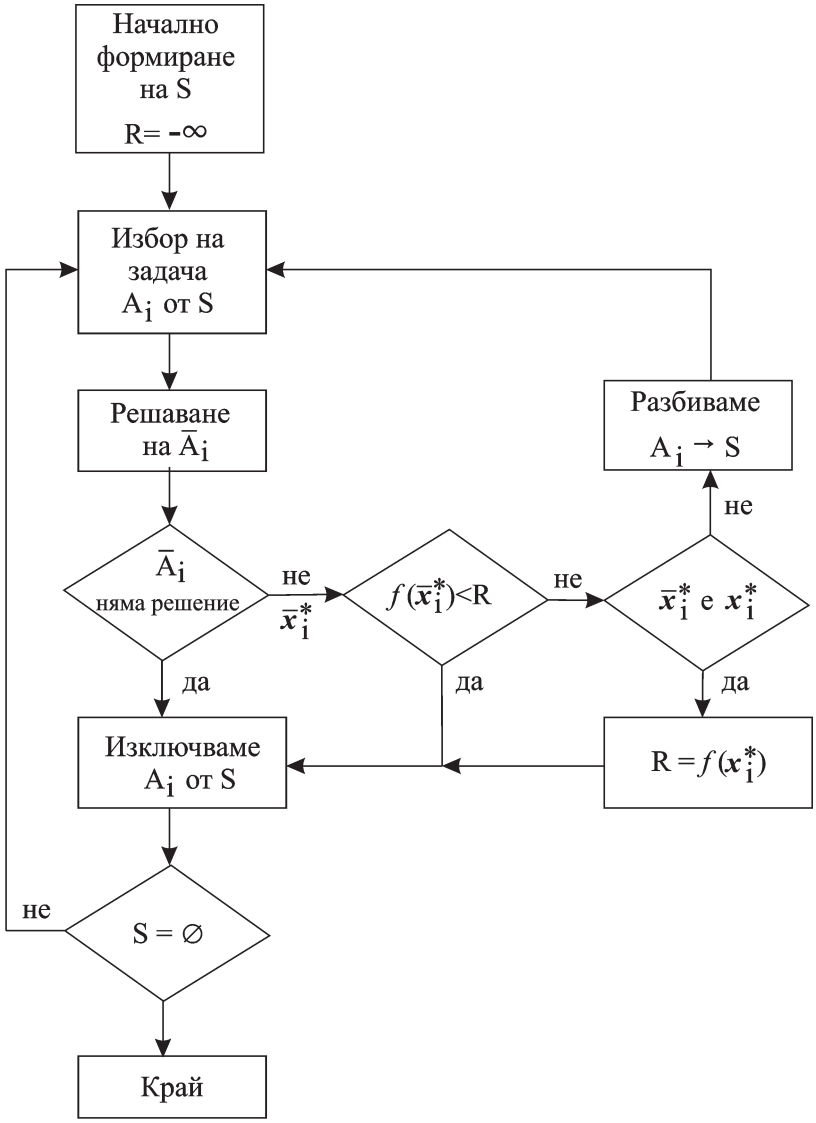
Същността на алгоритъма е описана чрез дадената на фигура 12.1 блок-схема.

Някои обяснения, които при внимателно разглеждане на блок-схемата са очевидни:

- ако $\overline{A}_i$ няма решение, то и A_i няма решение;
- ако $f(\overline{\mathbf{x}}_i^*) < R$, то без да проверяваме дали $\overline{\mathbf{x}}_i^*$ е решение и на A_i сме сигурни, че и неговата стойност е под рекорда;

- при „край“, ако $R = -\infty$, то задачата A няма решение, иначе решение е последното намерено.

Тук възникват редица въпроси, свързани с ефективността на алгоритъма. Колко „лесно“ се решават задачите $\overline{A}_i$, каква е разликата между стойностите на техните решения и тези на A_i, т.е. колко точни са оценките, няма ли S да стане много голям списък и т.н.



Фигура 12.1.

Принципните отговори са: колкото по-грубо е разширението $\overline{\Omega}_i$ на Ω_i, толкова по-лоша ще бъде оценката, но за сметка на това $\overline{A}_i$ ще се решава по-лесно. Компромисът тук се прави при всеки отделен вид задачи в зависимост от тяхната специфика. По-грубо разширение – по-лоша оценка – по-голям списък S. В екстремния случай ако решим

$\overline{\Omega}_i = \Omega$, то $|S| = 1$, но оценъчната ни задача всъщност е изходната! Ако задача A няма решение алгоритъмът се превръща почти в пълно изчерпване на вариантите. При целочислената линейна задача е най-естествено оценъчната да се получава чрез отпадане на изискването за целочисленост.

Да приложим алгоритъма за задачата за раницата

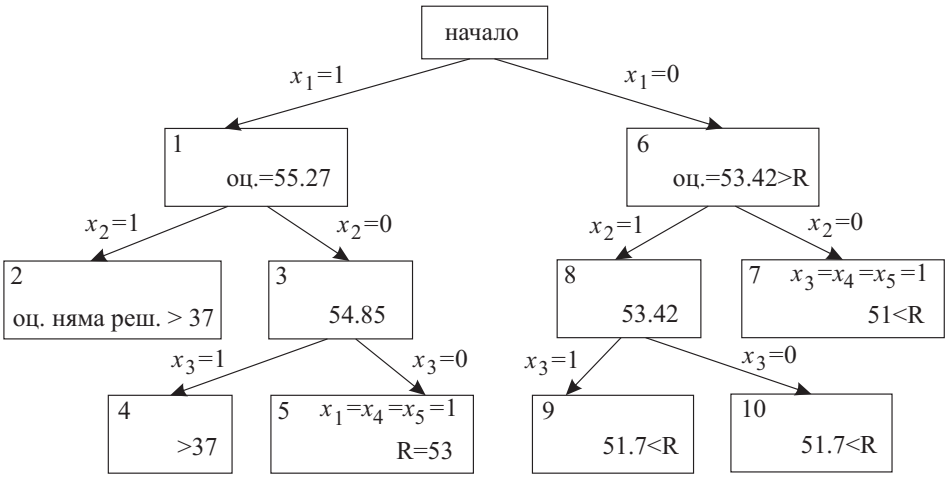
$$\max\left\{\mathbf{c}^{\mathrm{T}}\mathbf{x}:\ \mathbf{a}^{\mathrm{T}}\mathbf{x} \leqslant b,\ x_j \in \{0,1\},\ j = 1,\ldots,n\right\}\ - A,$$

като $\dfrac{c_1}{a_1} \leqslant \dfrac{c_2}{a_2} \leqslant \ldots \leqslant \dfrac{c_n}{a_n}$, което не намалява общността.

Ако $\displaystyle\sum_{j=1}^{k} a_j < b$, то $\overline{\mathbf{x}}^* = (1,1,\ldots,\underbrace{\alpha}_{k+1},0,\ldots,0)$, където $\alpha = \dfrac{b - \sum_{j=1}^{k} a_j}{a_{k+1}}$ е решение на оценъчната задача $\overline{A}$ $(0 \leqslant x \leqslant 1)$.

Пример.

$$\begin{array}{cccccccc} c_j & 32 & 32 & 30 & 8 & 13 & 11 \\ a_j & 21 & 22 & 21 & 6 & 10 & 9, \end{array} \qquad b = 37.$$



Фигура 12.2.

Изчислителният процес се състои в последователно намиране на все по-добри допустими цели решения, последното от които е оптималното, но този факт се установява едва след изчерпване на S, т.е. чрез директна проверка или отсичане с оценъчната задача на всички възможни варианти. Втората част на процеса обикновено е много по-трудна. Очевидното предимство на този метод пред метода на отсичащите равнини е, че тук получаваме добри допустими решения, докато при другия – или оптимума (което е свързано с неясно колко пресмятания) или нищо.

§13. Динамично оптимиране

Този метод е предложен от Р. Белман през 1950-53 год. и успешно се прилага при много широки класове задачи (дори и не само дискретни), но с общото свойство, че процесът на оптимизация може да се разбие на последователни стъпки с обща рекурентна връзка между тях.

Нека търсим

$$\max_{\mathbf{x}} \left\{ z(\mathbf{x}) = \sum_{j=1}^{n} f_j(x_j) : \sum_{j=1}^{n} a_j x_j \leqslant b, 0 \leqslant x_j - \text{цели}, j = 1, \ldots, n \right\}.$$

Целевата функция, в която всяко събираемо зависи само от една променлива, се нарича адитивна, като на f_j не са наложени никакви ограничения.

Нека като първа стъпка отделим x_n и търсим

$$\max_{x_1,\ldots,x_{n-1}} \left\{ \sum_{j=1}^{n} f_j(x_j) = \max_{x_n} f_n(x_n) + \right.$$

$$\left. + \max_{x_1,\ldots,x_{n-1}} \left[\sum_{1}^{n-1} f_j(x_j) : \sum_{j=1}^{n-1} a_j x_j \leqslant b - a_n x_n \right] \right\}.$$

$f_n(x_n)$ има краен брой стойности и ако знаехме стойността на второто събираемо, задачата е решена. Продължаваме по същия начин

$$\max_{x_1,\ldots,x_{n-2}} \left\{ \sum_{j=1}^{n-1} f_j(x_j) = \max_{x_{n-1}} f_{n-1}(x_{n-1}) + \right.$$

$$\left. + \max_{x_1,\ldots,x_{n-2}} \left[\sum_{1}^{n-2} f_j(x_j) : \sum_{j=1}^{n-2} a_j x_j \leqslant b - a_n x_n - a_{n-1} x_{n-1} \right] \right\}.$$

Ако $b - \sum_{j=n}^{n-k} a_j x_j$ означим с b_k, $k = 0, 1, 2, \ldots n-2$, то получаваме следната рекурентна зависимост

$$\max_{x_1,\ldots,x_{n-k-1}} \left\{ \sum_{j=1}^{n-k} f_j(x_j) = \max_{x_{n-k}} f_{n-k}(x_{n-k}) + \right.$$

$$\left. + \max_{x_1,\ldots,x_{n-k-1}} \left[\sum_{1}^{n-k-1} f_j(x_j) : \sum_{j=1}^{n-k-1} a_j x_j \leqslant b_k \right] \right\} \tag{13.1}$$

за $k = 0, 1, 2, \ldots n-2$. Естествено $b_k = b_k(x_{n-k}, \ldots, x_n)$ и ако второто събираемо в горната рекурентна зависимост (13.1) означим с $F_{n-k-1}(b_{k-1})$ можем да я запишем по-компактно

$$\max_{x_1,\ldots,x_{n-k-1}} \left\{ \sum_{j=1}^{n-k} f_j(x_j) = \max_{x_{n-k}} f_{n-k}(x_{n-k}) + F_{n-k-1}(b_{k-1}) \right\}.$$

Накрая стигаме до

$$\max_{x_1} \left\{ f_1(x_1) + f_2(x_2) = \max_{x_2} f_2(x_2) + \max_{x_1} [f_1(x_1) :\ a_1 x_1 \leqslant b_{n-2}] \right\}$$

и

$$\max_{x_1} \{ f_1(x_1),\ a_1 x_1 \leqslant b_{n-2} \},$$

което можем да пресметнем директно. Всъщност пресмятаме $f_1(x_1)$ а всички възможни стойности на x_1 и запомняме резултатите в таблица, тъй като те ще бъдат необходими при т.нар. обратен ход - пресмятаме по рекурентната зависимост $f_2(x_2)$ и т.н. до

$$\max_{x_1,\ldots,x_{n-1}} \left\{ \max_{x_n} f_n(x_n) + F_{n-1}(b_{n-1}) \right\},$$

при което намираме оптималната стойност x_n^*. Това вече можем да направим, понеже знаем (имаме пресметнати в таблица при предишния ход) стойности на второто събираемо за всяка възможна стойност на x_n. След това по същия начин пресмятаме x_{n-1}^* и т.н. до x_1^*.

Последователните таблици имат следния вид

b_1	$F_1(b_1)$	$\overline{x}_1$
0	$F_1(0)$	$\overline{x}_1(0)$
1	$F_1(1)$	$\overline{x}_1(1)$
$\vdots$	$\vdots$	$\vdots$
b		

b_2	$F_2(b_2)$	$\overline{x}_2$
0	$F_2(0)$	$\overline{x}_2(0)$
1	$F_2(1)$	$\overline{x}_2(1)$
$\vdots$	$\vdots$	$\vdots$
b		

$\ldots$

b_n	$F_n(b_n)$	$\overline{x}_n$
0	$F_n(0)$	$\overline{x}_n(0)$
1	$F_n(1)$	$\overline{x}_n(1)$
$\vdots$	$\vdots$	$\vdots$
b		

и $x_{n-1}^* = \overline{x}_{n-1}(b - a_n x_n^*)$, $x_{n-2}^* = \overline{x}_{n-2}(b - a_n x_n^* - a_{n-1} x_{n-1}^*)$ и т.н.

Веднага изниква въпросът колко големи са тази таблица. Например, при $b = 20$ и $n = 5$ броят на всички възможни варианти е

$$\frac{(n+b-1)!}{b!(n-1)!} = C_{24}^4 = 10\,626$$

и нашите таблици ще съдържат

$$\frac{(b+1)[(n-1)(b+2)+2]}{2} = 945$$

числа, т.е. докато броят на вариантите расте експоненциално с n и b, то размерът на пресмятанията в нашия случай расте с квадрата на b и линейно по n.

Нека решим по описания метод задачата

$$\max_{\mathbf{x}} \{3x_1 + 7x_2 + 15x_3,\ x_1 + 2x_2 + 3x_3 \leqslant 4,\ 0 \leqslant x_j - \text{цели}\}$$

като попълним трите таблици

b_1	F_1	$\overline{x}_1$
0	0	0
1	3	1
2	6	2
3	9	3
4	12	4

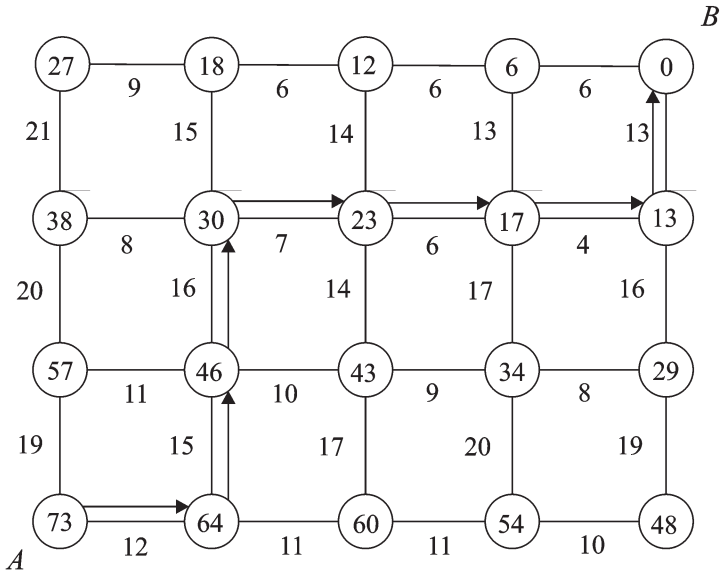
b_2	F_2	$\overline{x}_2$
0	0	0
1	3	0
2	7	1
3	10	1
4	14	2

b_3	F_3	$\overline{x}_3$
0	0	0
1	3	0
2	7	0
3	15	1
4	18	1

и $x_3^* = 1$, $x_2^* = \overline{x}_2(4-3) = \overline{x}_2(1) = 0$, $x_1^* = \overline{x}_1(4 - 3.1 - 2.0) = \overline{x}_1(1) = 1$ и максимумът е 18.

Основната идея в динамичното оптимиране се базира на т.нар. принцип на Белман, който гласи, че оптималната траектория на дадена система от състояние A до състояние B не зависи от това как тя е попаднала в състояние A. Системи с такова свойство се наричат Марковски системи. При по-внимателно вглеждане ще откриете, че точно на това се основават и горните рекурентни отношения.

Все пак идеята на динамичното оптимиране ще стане ясна от следния пример: Летателен апарат (самолет, ракета) трябва да достигне от т. A до т. B, отдалечени по разстояние и височина с минимален разход на гориво.



Фигура 13.3.

Хоризонталното и вертикалното разстояние са разделени на части като разходът на гориво за всяка отсечка е даден. В крайната точка B поставяме оценка 0. В точка B може да се достигне по хоризонталната

отсечка или по вертикалната. В първия случай разходът е 6, а във втория 13, така че в съответните възли поставяме 6 и 13. В съседния на вече оценените възли поставяме $\min\{6+13, 13+4\} = 17$ – това е минималното количество гориво за всички възможни начини да се стигне от този възел до B (а те са два).

По този начин оценяваме последователно всички възли и за A получаваме оценка 73, което е и стойността в оптималното решение. Самото оптимално решение възстановяваме по обратен път.

§14. Сложност на алгоритмите

Първото впечатление при сравняване на дискретните с непрекъснатите оптимизационни задачи е, че поради крайния брой на допустимите си решения те би трябвало да се решават по-лесно. Това не е вярно поради две основни причини – те са краен брой, но много голям – броят им расте експоненциално с размера на задачата и достига „невъобразими“ стойности. Втората причина е, че в арсенала за решаване на непрекъснатите задачи е цялата класическа математика с всичките си методи за локално изследване, докато в дискретната оптимизация принципът за локалност не важи.

Това подтиква изследванията върху качествата и класификацията на алгоритмите по отношение на ефективността им. Така се раждат теориите за сложност и изчислимост, сводимост, NP-пълнота и т.н.

Накратко и съвсем неформално ще разгледаме някои от тези неща. Отначало са дефинициите.

Задача или масова задача – набор от параметри и променливи, отношенията между тях и общ въпрос, на който трябва да се отговори.

Индивидуална задача – масова задача с конкретни стойности на параметрите.

Алгоритъм – процедура, описана върху структурата на масова задача, водеща до отговор на въпроса в нея.

Алгоритъмът решава масовата задача, ако може да реши всяка индивидуална задача.

Времето за решаване на дадена индивидуална задача е функция на размера ѝ, но понеже това понятие не е точно дефинирано – на дължината на входните данни (дължина на входа) без да конкретизираме начина на кодиране и бързината на изчислителното устройство.

Полиномиален алгоритъм – който решава всяка индивидуална задача за време не по-голямо от полином от дължината на входа. Например, времето за решаване на система линейни уравнения от ред n е не повече от Cn^2 (в случая n е пропорционално на дължината на входа).

Алгоритъм, който не се поддава на такава оценка се нарича експоненциален.

Сводимост на една задача към друга имаме, когато съществува полиномиален алгоритъм, с който едната задача може да се преобразува в другата.

За решаването на много класове задачи съществуват полиномиални алгоритми – задачи от линейната алгебра, линейната оптимизационна задача, редица от задачи върху графи и др. Множеството от тези задачи образуват т.нар. клас P (polinomial).

Задачите, за които не са намерени полиномиални алгоритми, образуват класа NP. Такива са всички задачи, на които правихме модели в началото – задачи за раницата, търговския пътник, разкрой и

много други. За тях, както отбелязахме няма намерени полиномиални алгоритми, но не значи, че такива изобщо не съществуват – това не е доказано. Много любопитен факт е, че някои задачи от класа NP имат свойството, че всяка друга задача от NP е полиномиално сводима към тях. Такива са задачите за търговския пътник, за раницата и др. Наричат ги NP-пълни задачи. Това означава, че ако се намери полиномиален алгоритъм за някоя от тях, то ще съществуват полиномиални алгоритми за всяка от задачите от NP, т.е. $P = NP$ и ще настъпи всеобщо щастие – всички задачи се рашават за полиномиално време! Засега, въпреки многото усилия, този факт не е установен.

Все пак пълният песимизъм не е уместен, защото на практика редица задачи от NP се решават за разумно време и само най-лошите (the worst) примери искат експоненциално време за решаване.

§15. Приближени алгоритми

Алгоритъм, чиято цел е намиране на допустимо решение (колкото може по-добро) на дадена задача, наричаме приближен алгоритъм за решаване на задачата. Намирането на оптималното решение не е основната цел на никой приближен алгоритъм и дори ако то бъде намерено не може да се докаже, че е оптимално, но ценни са тези от тях, които могат да гарантират качеството на намереното допустимо решение, т.е. отношението му към оптимума. Такива алгоритми наричаме приближени алгоритми с гарантирана точност.

Нека D е масова задача за минимизация, A – приближен алгоритъм за нея, $I \in D$ е индивидуална задача, $A(I)$ – решение на I с алгоритъма A и $OPT(I)$ е оптималното решение на I.

Ако $R_A(I) = \dfrac{A(I)}{OPT(I)}$ е относителната грешка за I, то

$$R_A = \inf\{r \geqslant 1; R_A(I) \leqslant r \text{ за всяка } I \in D\}$$

наричаме грешка на алгоритъма A, а

$$R_A^\infty = \inf\{r \geqslant 1; \text{ и съществува цяло } N, \text{ такова че за всяка} I,$$

$$\text{за която } OPT(I) > N,\ R_A(I) \leqslant r\}$$

– асимптотична грешка.

Разликата между двете е, че R_A^∞ не взима предвид грешката при задачи с малки размери, които не са достатъчно предствавителни за целия клас задачи.

Да илюстрираме казаното върху класа задачи, наречени „опаковка в контейнери“ – Bin Packing Problem. Имаме n предмета u_i с тегла $0 < s(u_i) < 1$, $i = 1, \ldots, n$, които са рационални числа. Задачата е да се разпределят в най-малък брой контейнери с обеми 1.

Доказано е, че тази задача е NP-пълна и не е известен полиномиален алгоритъм за решаването ѝ.

Да разгледаме алгоритъма, наречен FF – First Fit, т.е. поредния предмет се поставя в първия контейнер, в който е възможно. Нека за задачата I, $FF(I)$ и $OPT(I)$ са намереното с него и оптималното решение, съответно.

Понеже след завършване работата на FF не е възможно да останат два контейнера, запълнени по-малко от $1/2$, $FF(I) \leqslant [2\sum_{i=1}^{n} s(u_i)]$, но $\sum_{i=1}^{n}[s(u_i)] \leqslant OPT(I)$, то $FF(I) \leqslant 2OPT(I)$.

Следователно, FF дава решение, което надвишава оптималното не повече от два пъти. Оказва се, че може да се получи по-добра оценка, която при това е точна, т.е. не може да се подобри.

Теорема 15.1. *За всяка задача* $I \in BIN$,

$$FF(I) \leqslant \frac{17}{10}(OPT(I) + 2)$$

и съществуват задачи с произволно големи размери, за които

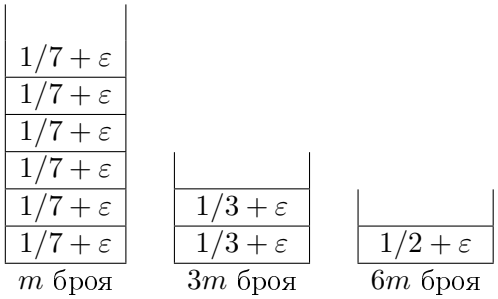
$$FF(I) > \frac{17}{10}(OPT(I) - 1).$$

Ето един любопитен пример, илюстриращ теоремата: имаме $\{u_i\}$, $i = 1, \ldots 18m$ при $m = 1, 2, 3, \ldots$ задачата е с произволна големина и

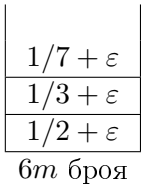
$$s(u_i) = \begin{cases} \frac{1}{7} + \varepsilon, & i = 1, \ldots 6m \\ \frac{1}{3} + \varepsilon, & i = 6m + 1, \ldots, 12m \\ \frac{1}{2} + \varepsilon, & i = 12m + 1, \ldots, 18m, \end{cases}$$

където $\varepsilon > 0$ е достатъчно малко число.

Прилагайки FF получаваме следното разпределение:
$FF(I) \rightarrow$



т.е. предметите се разполагат в $10m$ контейнера, а $OPT(I)$ заема $6m$ контейнера, като
$OPT(I) \rightarrow$



и следователно

$$\frac{10m}{6m} OPT(I) < \frac{17}{10} OPT(I).$$

Конструирани са примери, които удовлетворяват и долната граница, но те са доста по-обемисти.

Да разгледаме усъвършенствания алгоритъм FFD – First Fit Decreasing – процедира се както при FF, но предметите са подредени по намаляващ обем.

Теорема 15.2. *За всяка задача* $I \in BIN$,

$$FFD(I) \leqslant \frac{11}{9}(OPT(I) + 4)$$

и съществуват задачи с произволно големи размери, за които

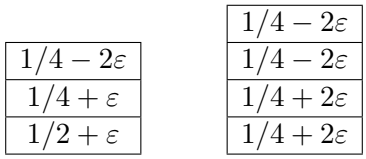
$$FFD(I) \geqslant \frac{17}{10} OPT(I).$$

Следващият пример удовлетворява и двете неравенства, а и двата примера ни показват защо бе необходимо да дефинираме и понятието асимптотическа точност на алгоритъма.

Нека $\{u_i\}$, $i = 1, \dots 30m$ при $m = 1, 2, 3, \dots$ и

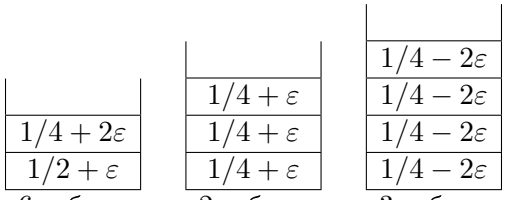
$$s(u_i) = \begin{cases} \frac{1}{2} + \varepsilon, & i = 1, \dots 6m \\ \frac{1}{4} + 2\varepsilon, & i = 6m+1, \dots, 12m \\ \frac{1}{4} + \varepsilon, & i = 12m+1, \dots, 18m \\ \frac{1}{4} - 2\varepsilon, & i = 18m+1, \dots 30m. \end{cases}$$

$OPT(I) \rightarrow$



$6m$ броя $3m$ броя $OPT(I) = 9m;$

Прилагайки FFD получаваме следното разпределение:
$FFD(I) \rightarrow$



$6m$ броя $2m$ броя $3m$ броя $FFD(I) = 11m.$

Нека се върнем към задачата за търговския пътник и потърсим оценки за някои приближени алгоритми. Най-естественият подход е ако се намирате в даден град да отидете в най-близкия (ако има няколко такива – в този с най-малък номер). Допълнително изискване за разстоянията да бъде изпълнено неравенството на триъгълника

$$d(a, c) \leqslant d(a, b) + d(b, c)$$

като задачата остава пак NP-пълна. Този алгоритъм е наречен NN – Nearest Neighbour.

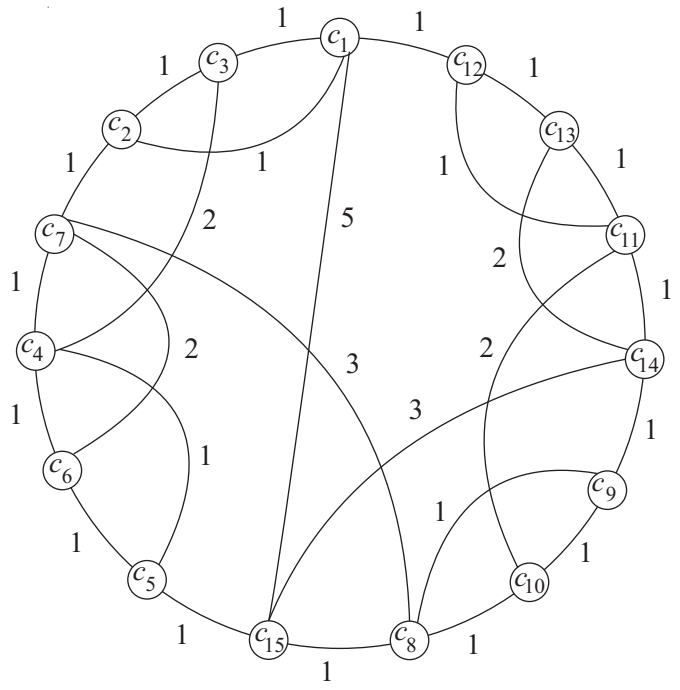
Теорема 15.3. *За всички индивидуални задачи с m града и изпълнено неравенство на триъгълника за разстоянията между тях имаме*

$$NN(I) \leqslant \frac{1}{2} \left([\log_2 m] + 1 \right) OPT(I)$$

и за произволно големи m съществуват задачи, за които

$$NN(I) > \frac{1}{3} \left(\log_2(m+1) + \frac{4}{3} \right) OPT(I).$$

Втората оценка в теоремата показва, че $R_{NN} = \infty$ и този „най-естествен“ алгоритъм съвсем не е перспективен. Ето пример при $m = 15$.



Фигура 15.4.

В случая $NN(I) = 27$, а $OPT(I) = 15$ и

$$R_{NN}(I) = \frac{27}{15} > \frac{16}{9} = \frac{1}{3}\left(\log_2(m+1) + \frac{4}{3}\right).$$

Изобщо, задачата за търговския пътник е много „неблагодарна“ и за точно и за приближено решаване, но и за нея са намерени алгоритми с оценка 2 и дори $3/2$, основани съответно на „минимално обхващащо дърво“ и „дихотомия“ в графа.

По въпросите, които твърде бегло споменахме по-горе може подробно да се осведомите в увлекателната книга в превод на руски: М. Гэри, Д. Джонсон, Вычислительные машины и трроднорешаемые задачи, Мир, 1982.