MTA Számítástechnikai és Automatizálási Kutató Intézete, **Közlemények** 15/1975.

A VIRTUÁLIS MEMÓRIA EGY ALGEBRAI ÉS EGY VALÓSZINÜSÉGELMÉLETI MODELLJE

Gyires Tibor

MTA Számítástechnikai és Automatizálási Kutató Intézete

Az utóbbi időben egyre nagyobb az érdeklődés az eddigiektől eltérő memória szervezés iránt, aminek a neve virtuális memória. Milyen okok tették szükségessé egy uj rendszer megtervezését, ami sok elméleti és gyakorlati probléma kiinduló pontja lett?

A számitógépek elterjedése lehetővé tette, hogy egyre bonyolultabb, összetettebb feladatokat számitógép segitségével oldjanak meg, amelyek megoldásához egyre nagyobb programokat kellett irni. A programok nagyságának a belső memória nagysága szabott felső határt. A belső memória drágasága miatt felvetődött a probléma, hogyan lehet lefuttatni egy olyan programot, ami meghaladja a memória nagyságát? A kérdésre két megoldás született:

overlay technika,
virtuális memória szervezés.

Az overlay technika a következő: a programot a felhasználó tetszés szerinti nagyságu darabokra, logikailag összefüggő u.n. szegmensekre bontja, amelyek közül egynek, a "main"–nak kitüntetett szerepe van. Ez a szegmens állandóan benn van a memóriába és ezen keresztül valósul meg a többi szegmens közötti kommunikáció. A memóriában mindig két szegmens van, a main és az aktiv szegmens. A fennmaradó szegmensek valamilyen külső tárolóra (disk mágnesszalag, stb.) kerülnek, melyek közül mindig azt cseréljük ki a memóriában levővel, amelyikre hivatkozás történik a programból.

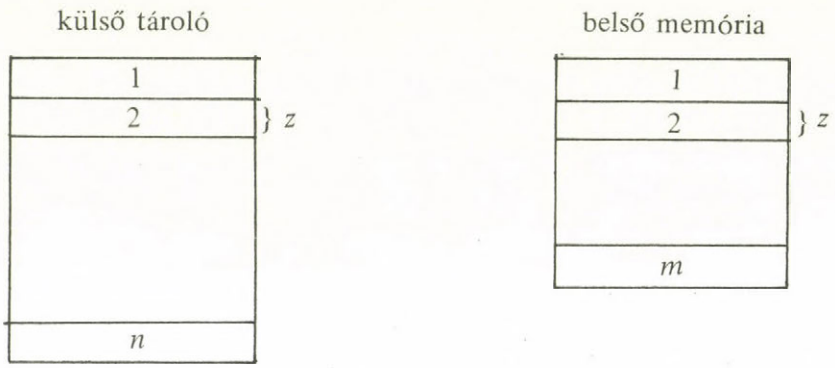
A virtuális memória szervezés esetében a programot és a memóriát változó, vagy egyenlő nagyságu nem szükségképpen logikailag összefüggő blokkokra osztjuk. Változó hosszuságu blokkok esetében szegmentált, egyenlő hosszuságu blokkok esetében page–elt szervezésről beszélünk. A blokkok között nincs kitüntetett. A nem betölthető blokkokat valamilyen külső tárolóra visszük, ahonnan szükség esetében behozzuk a memóriába. A program felosztása és a blokkok cseréje az overlay–jel ellentétben automatikus. A dolgozatban a page–elt szervezéssel fogunk foglalkozni.

Mindkét esetben az a cél, hogy a program futása közben a lehető legkevesebbszer hivatkozzon olyan programrészre, amely külső tárolón helyezkedik el, vagyis a program lehető legkevesebb interrupt–tal fusson le. A dolgozat a virtuális memória egy algebrai és egy valószinüségelméleti modelljét tartalmazza. A modellek célja különböző feltételekkel bizonyitani azt, hogy abban az esetben, ha a blokkok cseréjére az u.n. önjavitó algoritmust alkalmazzuk, a blokkcserék számát tekintve ez az algoritmus optimális.

I. Algebrai közelítés

1.1. **Definició.** Egy page–elt virtuális memória rendszernek nevezünk egy $V = (N,M,f)$ hármast, ahol $N = \{1,2, \ldots, n\}$ a programot alkotó page–eket, $M = \{1,2, \ldots, m\}$ a memória fizikai page-it jelöli, ahol $1 \leqslant m \leqslant n$ és $f: N \to M$ "page függvény" azzal a tulajdonsággal, hogy

$$f(i) = \begin{cases} j, & \text{ha az } i\text{-edik page a } j\text{-edik fizikai page-ben van.} \\ \text{undef.} & \text{egyébként} \end{cases}$$



Ha a page hossza z (pl. z szó), akkor egy α virtuális címre $0 \leqslant \alpha < nz$ hasonlóan egy β virtuális címre $0 \leqslant \beta < mz$. Ha adott egy α virtuális cím, a rendszer software vagy hardware uton meghatároz egy (i,w) párt ugy, hogy $\alpha = (i-1)z + w$ ahol $0 \leqslant w < z$, és generálja az α-hoz tartozó β memória cimet, amelyre

$$\beta = [f(i) - 1]\, z + w, \quad \text{ha } f(i) \text{ definiálva van.}$$

Ha az $f(i)$ page függvény nem definiált, a program végrehajtásában interrupt lép fel "page várakozás" ideig, addig amig a rendszer nem tölti be a hiányzó page-t a külső tároló egy rendelkezésre álló fizikai page-ébe és módositja a page függvényt a memória aktuális tartalmának megfelelően. Interrupt esetén a következő problémák lépnek fel:

a.) helyettesitési probléma: melyik page-t vigyük ki a belső memóriából, hogy be tudjuk tölteni a hivatkozott page-t?

b.) elhelyezési probléma: hová töltsük a bejövő page-t?

c.) betöltési probléma: mikor töltsük be a hiányzó page-t?

Ha egy page-t akkor és csak akkor töltünk be, amikor a programból hivatkozás történik rá "demand page" -elérésről beszélünk, ellenkező esetben "nondemand" vagy "prepage"-elésről.

Demand page-elés esetén az elhelyezési probléma megoldása triviális. Ha a memória még nem telt be, a bejövő page-t tetszőlegesen helyezzük el egy üres fizikai page-be, ha a memóriában már nincs üres page, először a helyettesitési problémát megoldó u.n. helyettesitési algoritmust hajtjuk végre, majd az igy kijelölt page helyébe töltjük a bejövő page-t, vagyis a demand page-elő rendszerek vizsgálatát elegendő a helyettesitési probléma vizsgálatára korlá-

tozni. Mivel a későbbiekben egy feltétel fennállása esetén kimutatjuk, hogy egy demand page-elő rendszer az interruptok számát tekintve hatékonyabb, mint egy prepage-elt rendszer, a továbbiakban demand page-elt rendszereket vizsgálunk.

Mint az előzöekben, legyenek $N = \{1,2, \ldots ,n\}$ egy program page-i, $M = \{1,2, \ldots ,m\}$ a memória fizikai page-i, és $1 \leqslant m \leqslant n$. Jelölje N^k a k hosszuságu sorozatok halmazát N fölött, $k \geqslant 0$.

Jelölje $\omega = r_1, r_2, \ldots , r_T$ N^T-beli elem egy program page hivatkozásait valamely T-re. Ha $r_t = x$, ez azt jelenti, hogy a program a t-edik időpillanatban az x page-ra hivatkozott. Legyen $S \subseteq N$ egy memória allapot és

$$\mathfrak{M}_m = \{ S / S \subseteq N \text{ és } |S| \leqslant m \}$$

ahol $|X|$ jelöli az X halmaz elemeinek a számát.

1.2. **Definició.** Legyen M és N adott. Egy M-re vonatkozó page-elő algoritmusnak nevezzük az $A = (Q, q_0, g)$ hármast, ahol

Q az algoritmus kontroll állapotainak a halmaza,
q_0 eleme Q-nak, kezdeti kontroll állapot,
g $\mathfrak{M}_m \otimes Q \otimes N \longrightarrow \mathfrak{M}_m \otimes Q$ allokáló függvény azzal a tulajdonsággal, hogy amennyiben

$g(S,q,x) = (S', q')$, akkor $x \in S'$

Jelölje $S+x$ az $S \cup \{x\}$ halmazt, $x \notin S$
és $S-x$ az $S - \{x\}$ halmazt, $x \in S$

1.3. **Definició.** Az A page-elő algoritmus demand page-elő algoritmus, ha allokáló függvénye:

$$g(S,q,x) = \begin{cases} (S,q') & \text{ha } x \in S \\ (S + x, q') & \text{ha } x \notin S \text{ és } |S| < m \\ (S + x - y, q') & \text{ha } x \notin S, \quad |S| = m, \ y \in S \end{cases}$$

Jelölje $|X_t|$ a t-edik időpillanatban betöltésre kerülő page-k számát, $|Y_t|$ a kivitt page-k számát. Az algoritmus költségének kiszámitásánál az $|Y_t|$ page kivitelének költségét figyelmen kivül hagyjuk, hiszen feltételezhetjük, hogy a betöltés a kivitellel együtt történik.

1.4. **Definició.** Jelölje $h(k)$ k page betöltésének költségét, $k \geqslant 1$, $h(k) \geqslant h(1) = 1$. Legyen $\omega = r_1, r_2, \ldots , r_T$ a program hivatkozás stringje, S kezdeti memória állapot.

Az A page-elő algoritmus S-re és ω-ra vonatkozó költsége

$$C(A,S,\omega) = \sum_{t=1}^{T} h(|X_t|).$$

Ha A demand-page-elő algoritmus, $|X_t| \leqslant 1$, $1 \leqslant t \leqslant T$ és

$$C(A,S\omega) = \sum_{t=1}^{T} |X_t|$$

Egy algoritmus optimális, ha költsége minimális.
Érvényes a következő, Peter J. Denning-től származó tétel.

1.1. **Tétel.** *Legyen* $A = (Q,q_0,g)$ *egy page-elő algoritmus, és tegyük fel, hogy* $h(k) \geqslant k$, $k \geqslant 1$, $h(1) = 1$, *Akkor létezik egy* A' *demand page-elő algoritmus, ugy, hogy*

$$C(A',S_0,\omega) \leqslant C(A,S_0,\omega)$$

bármely S_0*-ra,* ω*-ra.*

A tétel bizonyitja, hogy a virtuális memória szervezés legalább olyan hatásfoku az interruptok optimális számát tekintve, mint az overlay szervezés, ezenkivül a kűlönböző programrészek cseréje a virtudlis memória szervezés esetében automatikus, az overlay esetében ez felhasználó feladata.

Milyen algoritmus szerint cserélhetjük a page-ket, hogy a program a lehető legkevesebb interrupt-al fusson le?

Ahhoz, hogy ezt elérjük, azt a page-t kellene kicserélni, amelyre a legkésőbb történik a programból hivatkozás. Ez az algoritmus viszont feltételezi a hivatkozás string ismeretét, amely az esetek többségében nem áll rendelkezésünkre, igy az ilyen tipusu algoritmusokkal kapcsolatos eredmények elméleti jelentőségüek, a gyakorlatban nem alkalmazhatók. A gyakorlatban alkalmazott algoritmusok a page cseréig gyüjtöttt információk alapján döntenek. Két hasonló, már alkalmazott algoritmus a FIFO, LRU (Least Recently USED). A FIFO algoritmus a jól ismert first-in first-out alapon cseréli a page-ket. Az LRU azt a page-t cseréli ki, amelyre a program a legrégebben hivatkozott.

Mindkét algoritmus kontroll állapotainak halmaza $Q = N^m$, legyen Q egy eleme $q = (y_1,y_2,\ldots,y_m)$.
$|S| = m$ esetén a két algoritmus allokáló függvénye:

$$g_{FIFO}(S,q,x) = \begin{cases} (S,q) & \text{ha } x \in S \\ (S + x - y_m, q') & \text{ha } x \notin S \end{cases}$$

$$g_{LRU}(S,q,x) = \begin{cases} (S,q'') & \text{ha } x \in S \\ (S + x - y_m, q') & \text{ha } x \notin S \end{cases}$$

ahol

$$q' = (x, y_1, \ldots, y_{m-1}) \qquad \text{és ha } x = y_k$$

akkor

$$q'' = (x, y_1, \ldots, y_{k+1}, \ldots, y_m)$$

Önjavító algoritmus

Célszerünek látszik egy olyan algoritmus bevezetése, amely azt a page-t jelöli ki cserére, amelynek a relativ gyakorisága a legkisebb (egy page relativ gyakorisága alatt értjük a program futás elindulása óta a page-re történt hivatkozások számának és az összhivatkozások számának a hányadosát). Nevezzük ezt önjavító algoritmusnak.

Kontroll állapotának halmaza $Q = N^m$, az algoritmus egy állapota $q = (y_1, y_2, \ldots, y_m)$ és jelölje $\tau(y,t)$ az y page-re a t időpillanatig történt hivatkozások számát, v a t időpillanatig történt összes page hivatkozások számát.

Rendezzük az $y_1, y_2, \ldots, y_m$ page-ket ugy, hogy

$$\frac{\tau(y_i, t)}{v} \leqslant \frac{\tau(y_{i+1}, t)}{v}, \qquad i = 1,2, \ldots, m-1$$

Az algoritmus allokáló függvénye

$$g(S,q,x) = \begin{cases} (S, q') & \text{ha } x \in S \\ (S + x, q') & \text{ha } x \notin S, \ |S| < m \\ (S + x - y_1, q'') & \text{ha } x \notin S \ \ |S| = m, y_1 \in S \end{cases}$$

ahol

$$q'' = (x, y_2, \ldots, y_m)$$

és ha

$$y_k = x, \quad q' = (y_1, \ldots, y_{k+1}, y_k, \ldots, y_m)$$

1.5. **Definició.** Egy program egy $P = (N, U, u_0, f)$ négyes, ahol

N: a program page-k halmaza,
U: a program állapotok halmaza,
u_0: eleme U-nak, kezdeti program állapot,
f: állapot függvény, melyre $f: N \times U \longrightarrow U$

(Egy program állapot lehet például a memória aktuális tartalma.)

Jelölje $p(x,t) = P(r_t = x)$ annak a valószínüségét, hogy a t-edik időpillanatban az x-page-ra történt hivatkozás, $x \epsilon N$ és $\omega = r_1, r_2, \ldots, r_T$.

Egy program l-ed rendü, ha $|U| = l + 1$.

Jelöljön $<$ N fölött egy bináris relációt ugy, hogy ha

$$x < y \Rightarrow p(x,t) \leqslant p(y,t), t > 0.$$

$x \leqslant y$ jelenti, hogy $x < y$ vagy $x = y$.
$s = \min S$ olyan S-beli elem, melyre $s \leqslant x$ bármely $x \epsilon S$.

1.2.**Tétel.** *Ha egy P program 0-ad rendü, és értelmezve van N fölött egy $<$ bináris reláció, akkor az A optimális page-elő algoritmusnak a következő g: $\mathfrak{A}_m \times N \longrightarrow \mathfrak{A}_m$ allókáló függvénye van*

$$g(S,x) = \begin{cases} S & \text{ha } x \epsilon S \\ S + x - s, & \text{ha } x \notin S \end{cases} \quad \text{ahol}$$

$s = \min S$ *és* $|S| = m$.

Bizonyitás [1]-ben.

A tételben szereplő allokáló függvény azonos az önjavitó algoritmus allokáló függvényével, vagyis az önjavitó algoritmus optimális.

II. Az önjavitó algoritmus egy valószinüségelméleti közelitése

Legyen m a belső memória page-inek száma. Osszuk fel valamely adott programot $y_1, y_2, \ldots, y_m, \ldots, y_{m+n}$ page-ekre. Tegyük fel, hogy a page-ekhez rendre az adott programra vonatkozó hivatkozási valószinüségek

$$p_1 > p_2 > \ldots > p_m > \ldots p_{m+n} > 0$$

sorozata tartozik. Tegyük fel továbbá, hogy az egyes page-ekre történő egymásutáni hivatkozások függetlenek.

Legyen v a page–hivatkozások száma, ebből történjen

v_1 - szer az y_1-re
v_2 - szer az y_1-re
.
.
.
v_{m+n} - szer az y_{m+n}-re

hivatkozás, $\sum_{j=1}^{m+n} v_j = v$

Megjegyzés. Amennyiben a valószinüségeknek a sorozata egy rendezést követ, ugy a hozzájuk tartozó relativ gyakoriságoknak a sorozata is elegendő nagy M-re közel 1 valószinüséggel ugyanazt a rendezést követi.

Ezt a következő meggondolások alapján lehet belátni:

Legyen a valószinüségek sorozata

$$p_1 > p_2 > p_m > p_{m+1}, \ldots > p_{m+n}$$

A nagy számok gyenge törvénye értelmében

$$\forall \epsilon > 0, \delta > 0 \; \exists N_0^{(k)}, \text{ hogy } N_0^{(k)} \leqslant N\text{-re}$$

$$P\left(\left| \frac{\nu_k}{\nu} - p_k \right| < \epsilon \right) > 1 - \delta \qquad k = 1, 2, \ldots, m + n$$

Jelölje

$$p_1 - p_2 = \Delta_1$$
$$p_2 - p_3 = \Delta_2$$
$$\cdot$$
$$\cdot$$
$$\cdot$$
$$p_{m+n-1} - p_{m+n} = \Delta_{m+n-1}$$

$\Delta_i > 0$, $i = 1,2, \ldots, m + n - 1$ a feltétel szerint és jelölje

$$\min_{i=1, \ldots, m+n-1} \Delta_i = \Delta, \qquad \text{és} \qquad \max_{i=1, \ldots, m+n-1} N_0^{(i)} = N_0$$

és válasszuk

$$\epsilon < \frac{\Delta}{2}$$

Igy $\epsilon > 0, \delta > 0 \; \exists N_0$, hogy $N_0 \leqslant N$-re

$$P\left(\left| \frac{\nu_k}{\nu} - p_k \right| < \epsilon, \; k = 1,2, \ldots, m + n \right) > 1 - \delta$$

A $(p_k - \epsilon, p_k + \epsilon)$ $(k = 1, \ldots, m + n)$ diszjunkt intervallumok, ezért

$$\lim_{\nu \to \infty} P\left(\frac{\nu_1}{\nu} > \frac{\nu_2}{\nu} > \ldots > \frac{\nu_{m+n}}{\nu} \right) = 1$$

Feltételezhetjük, hogy page-eknek pozitiv hivatkozási valószinüségük van, ami azt jelenti, hogy az y_i page előbb-utóbb bekerül a memóriába. De a következő cserénél, mivel az algoritmus a legkisebb relativ gyakoriságu page-t viszi ki, a megjegyzés szerint az y_i-nél kisebb valószinü-

ségü page kerül kivitelre, amennyiben van ilyen. Ha a page hivatkozások száma elég nagy, ez azt eredményezi, hogy a memóriában közel 1 valószinüséggel a legnagyobb valószinüségü $y_1, y_2, \ldots, y_{m-1}$ page-k maradnak benn, mig az y_m page bonyolitja le a cseréket, vagyis $p_1 > p_2 \cdots, > p_m > \cdots, p_{m+n}$ valószinüségnek megfelelő $y_1, y_2, \ldots, y_m, \ldots y_{m+n}$ rendezés alakul ki.

1.3.**Tétel.** *Az önjavitó algoritmus alkalmazása esetén az interruptok számának várható értéke aszimptotikusan minimális.*

Bizonyitás. Legyen μ_j egy karakterisztikus valószinüségü valtozó

$$\mu_j = \begin{cases} 1, \text{ ha a } j\text{-edik hivatkozás cserére vezet} \\ 0 \text{ egyébként} \end{cases}$$

Jelölje A azt az eseményt, hogy a rendezésben v_i pozíciója a j-edik lépésben nagyobb mint m. Rögzitsük $N_0 < k$-t. $k < N$-re, az interruptok számának várható értéke

$$\frac{1}{\nu}(E \sum_{j=1}^{n} \mu_j) = \frac{1}{\nu}\left[\sum_{j=1}^{N} \sum_{i=1}^{m+n} p_i P(A) \right] =$$

$$= \frac{1}{\nu}\left[\sum_{j=1}^{k} \sum_{i=1}^{m+n} p_i P(A) + \sum_{i=k}^{N} \sum_{i=1}^{m} p_i P(A) + \right.$$

$$\left. + \sum_{j=k}^{N} \sum_{i=m+1}^{m+n} p_i P(A) \right]$$

A $P(A)$ valószinüségek a megjegyzés szerint $i \leqslant m$ esetén 0-hoz, $i > m$ esetén 1-hez tartanak, igy az interruptok várható száma:

$$\lim_{\nu \to \infty} \frac{\nu - k}{\nu} \sum_{i=m+1}^{m+n} p_i \longrightarrow p_{m+1} + \cdots + p_{m+n}$$

Ez az összeg minimális, mert ha bármely olyan algoritmussal is dolgozunk, amelyik nem a legkisebb relativ gyakoriságu page-t cseréli ki a memóriából, vagyis nem önjavitó algoritmus, ez az érték csak nöhet.

A tétel igaz akkor is, ha nem tételezzük fel, hogy a $p_1, p_2, \ldots, p_{m+n}$ egymástól különbözőek. Ebben az esetben az algoritmus a megegyező valószinüségü page-k közül tetszés szerint jelöl ki egyet cserére.

Az eddigiekben feltételezzük, hogy a page-eknek egymásra való hivatkozásai függetlenek. Természetes módon merül fel a kérdés mit mondhatunk akkor, ha a függetlenséget függőségi megszoritásokkal helyettesitjük. Ilyen kérdésekkel foglalkoznak G. Ingargiola és J.F. Korsch [4] cikkükben.

Az előzőekben tárgyalt önjavitó algoritmus feltételezete azt, hogy a page hivatkozások száma egy nagy N számnál nagyobb. Felvetődik a kérdés, hogy milyen algoritmust alkalmazzunk abban az esetben, ha a page hivatkozások száma véges. Ezzel a kérdéssel nem azonos, de hozzá hasonló a "two-armed bandit" prébléma, amelyre megadott algoritmus véges vagy megszámlálhatóan végtelen nagy N-re is optimális. A "two-armed bandit" problémához hasonlóan a page problémát 3 page-re pl. a következőképpen fogalmazhatjuk meg:

Jelölje $N = \{1,2,3,\}$ a programot alkotó page-eket, $M = \{1,2\}$ a memória fizikai page-eit, és a page-eknek egymásra való hivatkozásainak a száma legyen véges. A feladat olyan algoritmus megadása, amely alkalmazása esetén a program a lehető legkevesebb interrupt-tal fut le.

Az igy megfogalmazott feladat nem minden algoritmus esetén azonos a "több pisztolyos bandita" próblémával sem. A 3-as számú page kint-tartozkodása véletlenszerű és ugyanakkor függ a másik két lehetséges page -párra vonatkozó stratégiától.

Irodalom

[1] Aho, Alfred V. – Denning, Peter J. – Ullman, Jeffery D.: Principles of Optimal Page Replacement. Journal of the Association for Computing Machinery. (1971), Vol. 18, No. 1. 80-93 p.

[2] Denning, P.J.: Virtual Memory. Computing Surveys. Vol. 2. No. 3. (1970), 153-189 p.

[3] Parmelee, R. P. etc.: Virtual Storage and Virtual Machine Concepts. IBM Systems Journal. 1972. No. 2. 99-130 p.

[4] Ingargiola, G. – Korsh, J. F.: Finding Optimal Demand Paging Algorithms. Journal of the Association for Computing Machinery. Vol. 21. No. 1. 1974. 40-53 p.

[5] Joseph, M.: An Analysis of Paging and Program Behaviour. Computer Journal. Vol.13. No.1. 1970. 48-54 p.

[6] F.R. Gantmacher: Matrizen Rechnung. II. Berlin, (1959)., 78 p.

Summary

An algebraic and a probabilistic model of the virtual memory

T. Gyires

In this paper the author gives a survey about new results of theoratical investigations connected with paging virtual memory.

In the probabilistic model the optimality of the self-correcting algorithm is proved in the case of independent reference string.

At last a problem is raised concerning with finite reference string.

Р е з ю м е

Алгебраическая и стохастическая модели виртуальной памяти

Т. Дирош

В данной работе афтор даёт польный обзор о современном состоянии теоретических исследований связанных с системами странимной (раде) виртуальной памяти.

В вероятностной модели в случае независимого ряда ссылок афтор доказивает оптимальность самоусовершенствующегося алгоритма.

В конце афтор указывает на проблему возникающую в случае конечного ряда ссылок.