KYBERNETIKA ČÍSLO 4, ROČNÍK 4/1968

O Eatonově-Zadehově metodě

KAREL SLADKÝ

V práci je vyšetřována souvislost mezi metodami lineárního a dynamického programování pro nalezení optimálního řízení, které převádí markovský řetězec ze známého stavu do zadaného stavu s minimálními očekávanými náklady. Na několika konkrétních příkladech je ilustrována efektivnost jednotlivých metod.

0. ÚVOD

V práci [1] Eaton a Zadeh vyšetřovali úlohu nalézti takové řízení dynamického systému, jehož chování je popsáno markovským řetězcem s ohodnocenými přechody mezi jednotlivými stavy, které s minimálními očekávanými náklady převede uvažovaný systém ze známého stavu do zadaného konečného stavu. Eaton a Zadeh pro tento účel navrhli z metod dynamického programování jistý iterační postup spočívající na postupném „ohodnocování" očekávaných nákladů v jednotlivých stavech systému při použitém řízení a v současném zlepšování použitého řízení.

V práci [2] Derman dokázal, že optimální řízení výše popsané úlohy stačí hledat ve třídě stacionárních, čistých strategií. V [2] rovněž popsal postup, pomocí kterého nalezení optimálního řízení výše popsané úlohy je možno formulovat jako úlohu lineárního programování.

V existující literatuře byla velká pozornost věnována metodám pro nalezení optimálního řízení markovských řetězců s ohodnocenými přechody mezi jednotlivými stavy pro případ, že počet přechodů roste do nekonečna a kritériem optimality je buď průměrný zisk připadající na jeden přechod nebo diskontovaný celkový zisk (tj. součet hodnot získaných při jednotlivých přechodech, jestliže hodnota získaná v k-tém přechodu je vynásobena α^k ($k = 1, 2, 3, ..., \infty$); kde $0 \leqq \alpha < 1$. α se nazývá diskontní faktor).

Pro tento případ byla vypracována řada postupů pro nalezení optimálního řízení, které byly založeny na metodách lineárního nebo dynamického programování (viz [3], [4], [5], [6], [7], [8]). Mezi algoritmy lineárního a dynamického programování pro řešení tohoto typu úloh existuje úzká souvislost, na kterou bylo poukázáno v [5] a v [9].

Pro případ, kdy kritériem optimality řízeného markovského řetězce je dosažení určitého stavu s minimálními očekávanými náklady (lehko se lze přesvědčit, že dosažení určitého stavu v minimální očekávané době je zvláštním případem této úlohy), v existující literatuře je popsána pouze Eatonova-Zadehova iterační metoda (viz [1]) a Dermanem vypracovaná metoda lineárního programování (viz [2]), aniž by bylo poukázáno na vzájemnou souvislost obou metod.

Cílem této práce je hlubší rozbor a doplnění známých algoritmů, jestliže kritériem optimality řízeného markovského řetězce jsou očekávané náklady pro dosažení určitého stavu. Je ukázáno, že postupy pro nalezení optimálního řízení této úlohy založené na metodách dynamického programování je možno též obdržet na základě metod lineárního programování. Je diskutován i případ, kdy kritériem optimality jsou diskontované náklady.

V dodatku 2 je na několika konkrétních příkladech ilustrována efektivnost jednotlivých metod lineárního a dynamického programování pro řešení tohoto typu úloh.

1. FORMULACE PROBLÉMU A JEHO ŘEŠENÍ POMOCÍ METOD DYNAMICKÉHO PROGRAMOVÁNÍ

Uvažujme dynamický systém, jehož chování lze popsat markovským řetězcem. Vyšetřovaný systém se může nacházet v jednom z $(n + 1)$ stavů. V i-tém stavu $(i = n + 1)$ lze použít jedné z s_i zadaných strategií. Soubor použitých strategií v jednotlivých stavech budeme nazývat řízením systému. Při použití strategie k v i-tém stavu je třeba vynaložit náklady c_{ik} a při použití této strategie pravděpodobnosti přechodu uvažovaného systému do ostatních stavů jsou určeny vektorem $\mathbf{p}_i^k = [p_{i1}^k, p_{i2}^k, \ldots, p_{i,n}^k, p_{i,n+1}^k]$ (kde $p_{ij}^k \geqq 0$ a $\sum_{j=1}^{n+1} p_{ij}^k = 1$). Přechod mezi 2 po sobě následujícími stavy trvá jednotkovou dobu. Při každém přípustném řízení (tj. jestliže v každém stavu používáme pouze zadané strategie) lze stavu $(n + 1)$ dosáhnout s jednotkovou pravděpodobností (tedy v konečném čase) z libovolného stavu systému. Po dosažení $(n + 1)$-ého stavu uvažovaný systém v tomto stavu trvale setrvává. Úlohou je nalézti takové řízení uvažovaného systému, při kterém očekávané náklady na převedení systému z počátečního do $(n + 1)$-ého stavu budou minimální.

Poznámka. Řízení systému budeme v celém textu značit velkými písmeny v závorce; jednotlivé strategie (tj. prvky použitého řízení) budeme označovat malými písmeny. Symbol $p_{ij}^{(S)}$ (resp. $c_i^{(S)}$) označuje pravděpodobnost přechodu z i-tého do j-tého stavu (resp. náklady v i-tém stavu) při použitém řízení (S).

V práci [2] Derman dokázal, že řešení této úlohy stačí hledat ve třídě stacionárních, čistých strategií. Tato vlastnost optimálního řízení vyšetřované úlohy, která se zdá být intuitivně zřejmá, nebyla v práci [1] odvozena.

Označíme-li očekávané náklady na převedení uvažovaného systému z j-tého stavu do $(n + 1)$-ého stavu jako v_j, potom lze pro i-tý stav při použitém řízení (S) psát (pro $(n + 1)$-ý stav klademe $v_{n+1}^{(S)} = 0$):

$$v_i^{(S)} = \sum_{j=1}^{n} p_{ij}^{(S)} \cdot v_j^{(S)} + c_i^{(S)} . \tag{1}$$

Úkolem je minimalizovat očekávané náklady na převedení systému ze zadaného stavu do $(n + 1)$-ého stavu. Z principu optimality dynamického programování

(viz [10]) je zřejmé, že optimální řízení musí splňovat vztah

$$v_i = \min_k \left[\sum_{j=1}^{n} p_{ij}^k \,.\, v_j + c_{ik} \right], \tag{2}$$

kde k značí strategii přípustnou v i-tém stavu. Vztah (2) musí být splněn pro všechny stavy i, které se mohou vyskytnout při převedení systéum z uvažovaného počátečního stavu do $(n+1)$-ého stavu. Jestliže uvažujeme větší počet vhodně zvolených počátečních stavů systému, lze se stejnými úvahami přesvědčit, že vztah (2) je splněn při optimálním řízení pro všechna $i \leqq n$.

V práci [1] Eaton a Zadeh navrhli řešit systém rovnic (2) (pro $i = 1, 2, \ldots, n$) následujícím iteračním postupem:

Algoritmus 1. (Eaton-Zadeh). 0. Zvolíme (libovolně) hodnoty váhových koeficientů v_j $(j = 1, 2, \ldots, n)$.

1. Na základě známých váhových koeficientů v_j $(j = 1, 2, \ldots, n)$ nalezneme v i-tém stavu $(i = 1, 2, \ldots, n)$ strategii k_i minimalizující výraz na pravé straně rovnice (2) a ze vztahu (2) vypočteme novou hodnotu váhového koeficientu v_i (pro $i = 1, 2, \ldots, n$).

2. S takto nalezenými hodnotami váhových koeficientů opakujeme bod 1. Celý postup provádíme tak dlouho, až je dosaženo požadované přesnosti jednotlivých hodnot.

Podle předpokladů úlohy při jakémkoliv přípustném řízení $(n+1)$-ý stav je dosažen s jednotkovou pravděpodobností v konečném čase z libovolného stavu systému a po dosažení $(n+1)$-ého stavu uvažovaný systém trvale setrvává v tomto stavu. Proto při každém přípustném řízení (S) stav $(n+1)$ je absorbujícím stavem, matice pravděpodobnosti přechodů $\| p_{ij}^{(S)} \|$ (kde $i, j = 1, 2, \ldots, n, n+1$) je ergodická a tedy aspoň pro jedno $i \leqq n$ platí $\sum_{j=1}^{n} p_{ij}^{(S)} < 1$. Za těchto podmínek lze dokázat (viz [1]), že iterační postup podle algoritmu 1 konverguje k hledanému řešení rovnice (2). Rychlost konvergence algoritmu 1 může však být při různých úlohách značně rozdílná. Ve většině případů lze podstatně lepších výsledků (a zdaleka méně závislých na povaze vyšetřované úlohy) dosáhnout algoritmem 2, jak je demonstrováno na několika příkladech v dodatku 2.

Algoritmus 2. 1. *Ohodnocení řízení.* Z hodnot $p_{ij}^{(S)}$, $c_i^{(S)}$ náležejících jistému přípustnému řízení (S) nalezneme váhy jednotlivých stavů v_i řešením systému lineárních rovnic:

$$v_i = c_{i,k_i} + \sum_{j=1}^{n} p_{ij}^{k_i} \,.\, v_j \quad \text{pro} \quad i = 1, 2, \ldots, n, \tag{3}$$

když v i-tém stavu je při řízení (S) použito strategie k_i.

2. *Zlepšování řízení.* Z hodnot v_i určených v bodě 1 vypočteme pro každý stav novou strategii k, která minimalizuje výraz

$$\min_k \left[c_{ik} + \sum_{j=1}^{n} p_{ij}^k \cdot v_j\right]. \tag{4}$$

V případě, že řízení použité v předchozím iteračním kroku bude po zlepšování řízení nezměněno, je toto řízení optimálním. Jinak se vrátíme k bodu 1 a výpočet opakujeme pro nově zjištěné strategie v jednotlivých stavech.

Podstatným znakem algoritmu 2 na rozdíl od algoritmu 1 je, že optimální řízení je nalezeno po konečném počtu kroků.

Důkaz konvergence algoritmu 2 k hledanému optimálnímu řízení je možno provést obdobně k důkazu Howardova iteračního postupu pro případ, že kritériem optimality je diskontovaný celkový zisk (viz [6]). Důkaz algoritmu 2 je proto pouze naznačen v dodatku 1. Algoritmus 2 lze rovněž obdržet aplikací speciálně upravené revidované simplexové metody na vhodně transformovanou úlohu lineárního lomeného programování popisující řízení markovského řetězce, jak bude ukázáno v další části této práce.

V případě, že v algoritmu 2 řešíme soustavu lineárních rovnic (3) iterační metodou (která vždy musí konvergovat, neboť pro každé i jest $\sum_{j=1}^{n} p_{ij} \leqq 1$ a aspoň pro jedno i platí $\sum_{k=1}^{n} p_{ij} < 1$ a matice $\|p_{ij}\|$ je ergodická) a jednotlivé iterační kroky při řešení této soustavy kombinujeme se zlepšováním použitého řízení, algoritmus 2 přechází v postup navržený Eatonem a Zadehem.

2. SOUVISLOST S LINEÁRNÍM PROGRAMOVÁNÍM

Abychom mohli poukázat na vzájemnou souvislost mezi metodami lineárního a dynamického programování pro nalezení optimálního řízení výše formulované úlohy, použijeme na rozdíl od [2] poněkud odlišného vyjádření vyšetřované úlohy pomocí lineárního programování.

Budeme předpokládat, že v $(n + 1)$-ém stavu uvažovaného markovského řetězce lze použít jediné strategie, pro kterou:

$$p_{n+1,r} = 1\,; \quad p_{n+1,i} = 0 \quad (\text{pro} \quad i \neq r)\,; \quad c_{n+1,1} = c_{n+1} = 0\,. \tag{5}$$

Tím stav $(n + 1)$ přestává být absorbujícím stavem a je tak sestrojen pomocný ergodický markovský řetězec. Stále předpokládáme, že přechod mezi libovolnými dvěma po sobě následujícími stavy tohoto markovského řetězce trvá jednotkovou dobu.

Označme $h(j, k)$ (resp. $\tau(j, k)$) očekávané náklady (resp. očekávanou dobu přechodu) při přechodu z j-tého do k-tého stavu při určitém řízení. Dále označme

pomocí π_j limitní pravděpodobnost, že pomocný markovský řetězec se bude nacházet v j-tém stavu, když počet přechodů roste nade všechny meze.

Protože $c_{n+1} = 0$, $p_{n+1,r} = 1$, jest $h(r, n+1) = h(n+1, n+1)$. Podle předpokladů při každém přípustném řízení dosáhneme stavu $(n+1)$ s jednotkovou pravděpodobností v konečném čase. Potom pro průměrné náklady Θ připadající na jeden přechod (čili na jednotku času) dostáváme tento vztah:

$$\Theta = \frac{h(n+1, n+1)}{\tau(n+1, n+1)} = \sum_{j=1}^{n+1} \pi_j \cdot c_j \ .$$

Jak je známo (viz na př. [11] str. 79) platí:

$$\tau(n+1, n+1) = \frac{1}{\pi_{n+1}} \ .$$

Z těchto vztahů dostáváme:

$$(6) \qquad h(r, n+1) = \frac{h(n+1, n+1)}{\tau(n+1, n+1)} \cdot \tau(n+1, n+1) = \frac{1}{\pi_{n+1}} \sum_{j=1}^{n+1} \pi_j \cdot c_j \ .$$

Vyšetřovaná úloha se tak redukuje na nalezení takového přípustného řízení pomocného markovského řetězce, při kterém bude minimalizován výraz (6). Při vyšetřování této úlohy pomocí lineárního programování budeme předpokládat existenci optimálního řízení ve třídě stacionárních smíšených strategií (tj. když v každém stavu s jistou pravděpodobností neměnou v čase používáme určitou strategii). Za tím účelem zavedeme nové neznámé π_{jk} vyjadřující pravděpodobnosti, že po velkém počtu přechodů pomocný markovský řetězec se bude nacházet v j-tém stavu a že v tomto stavu bude v následujícím kroku použito strategie k.

Zřejmě platí:

$$(7) \qquad \pi_i = \sum_{k=1}^{s_i} \pi_{ik} \quad (\text{pro } i = 1, 2, \ldots, n, n+1)$$

a dále platí známé vztahy pro limitní pravděpodobnosti π_i markovského řetězce (viz na př. [6]):

$$(8) \qquad \pi_i = \sum_{j=1}^{n+1} p_{ji} \cdot \pi_j \quad (\text{pro } i = 1, 2, \ldots, n, n+1),$$

$$\sum_{j=1}^{n+1} \pi_j = 1 \ .$$

Soustava (8) je lineárně závislá; $(n+1)$-á rovnice této soustavy je lineární kombinací ostatních rovnic, a proto nebudeme dále tuto rovnici uvažovat.

Jestliže dosadíme vztahy (5), (7) do rovnic (6), (8), přípustné řízení pomocného markovského řetězce, při kterém bude minimalizován vztah (6), musí být řešením úlohy 1.

Úloha 1. Za podmínek:

$$\pi_{jk} \geqq 0\,, \quad \pi_{n+1} \geqq 0\,,$$

$$\sum_{j=1}^{n} \sum_{k=1}^{s_j} (\delta_{ji} - p_{ji}^k) \,.\, \pi_{jk} = 0 \quad \text{pro} \quad i = 1, 2, \ldots, n\,;\; i \neq r\,;$$

$$\sum_{j=1}^{n} \sum_{k=1}^{s_j} (\delta_{jr} - p_{jr}^k) \,.\, \pi_{jk} - \pi_{n+1} = 0\,;\; \sum_{j=1}^{n} \sum_{k=1}^{s_j} \pi_{jk} + \pi_{n+1} = 1\,;$$

nalezněte minimum účelové funkce:

$$h(r, n+1) = \frac{1}{\pi_{n+1}} \sum_{j=1}^{n} \sum_{k=1}^{s_j} \pi_{jk} \,.\, c_{jk}\,,$$

kde δ_{ji} značí stejně jako v celém dalším textu Kroneckerovo δ.

Úloha 1 je úlohou lineárního lomeného programování. Abychom tuto úlohu mohli převést na úlohu lineárního programování, provedeme transformace nezávisle proměnných popsané v [12] a [13]. Stejných výsledků bychom dosáhli i v případě, že bychom použili transformace navržené Dermanem v [2].

Jestliže zavedeme:

$$z_{jk} = \pi_{jk} \,.\, t\,, \quad t > 0\,,$$

$$z_{n+1} = \pi_{n+1} \,.\, t \quad \text{pro} \quad j = 1, 2, \ldots, n;\; k = 1, 2, \ldots, s_j$$

a hodnotu t zvolíme tak, aby

$$1 = \pi_{n+1} \,.\, t \Rightarrow z_{n+1} = 1\,,$$

úloha 1 přechází na úlohu 2.

Úloha 2. Za podmínek:

$$z_{jk} \geqq 0\,;$$

$$\sum_{j=1}^{n} \sum_{k=1}^{s_j} (\delta_{ji} - p_{ji}^k) \,.\, z_{jk} = 0 \quad \text{pro} \quad i = 1, 2, \ldots, n;\; i \neq r\,;$$

$$\sum_{j=1}^{n} \sum_{k=1}^{s_j} (\delta_{jr} - p_{jr}^k) \,.\, z_{jk} = 1\,,$$

$$\sum_{j=1}^{n} \sum_{k=1}^{s_j} z_{jk} - t = -1 \tag{9}$$

nalezněte minimum účelové funkce:

$$h(r, n+1) = \sum_{j=1}^{n} \sum_{k=1}^{s_j} c_{jk} \,.\, z_{jk}\,.$$

Všimněme si blíže úlohy 2. Neznámá t se vyskytuje pouze v omezení (9) a nevyskytuje se v účelové funkci. Je patrné, že podmínka (9) bude při libovolných $z_{j,k} \geqq 0$ pro vhodné $t > 0$ splněna. Protože hodnota t se nevyskytuje v účelové funkci, nemusíme podmínku (9) v omezeních úlohy 2 dále uvažovat.

Potom úloha 2 má pouze n omezujících podmínek. Jak je známo z teorie lineárního programování (viz na př. [14]), optimální řešení jakékoliv úlohy lineárního programování stačí hledat mezi bázickými řešeními, tj. v našem případě ve třídě přípustných řešení, kde předpokládáme pouze n nenulových neznámých.

Při řízení markovského řetězce je nutno v každém z jeho n stavů zvolit jistou strategii (obecně smíšenou). V $(n + 1)$-ém stavu žádnou volbu nemůžeme provádět. Tento požadavek při formulaci vyšetřované úlohy pomocí lineárního programování znamená, že aspoň jedno $z_{j,k}$ (pro $j = 1, 2, \ldots, n$) musí být obsaženo ve zvoleném bázickém řešení.

Obě tyto podmínky je možno současně splnit pouze tehdy, jestliže v každém z n uvažovaných stavů zvolíme čistou strategii (tj. když právě jedno $z_{j,k}$ (pro $j = 1, 2, \ldots, n$) bude obsaženo v použitém bázickém řešení).

Při hledání optimálního řízení markovského řetězce pomocí lineárního programování budeme vycházet z úlohy 2. Na rozdíl od obecné úlohy lineárního programování v tomto případě a priori známe přípustné řešení a při zavedení nové proměnné do báze nemusíme zjišťovat kterou z proměnných báze máme vyloučit (neboť v bázickém řešení bude obsaženo pouze jedno z_{jk} (pro $j = 1, 2, \ldots, n$). Jestliže se zajímáme o očekávané náklady na řízení v různých stavech (tj. o hodnoty účelové funkce $h(r, n + 1)$ úlohy 2 pro různá r), jednotlivé případy se budou lišit pouze pravou stranou omezujících podmínek v úloze 2. V případě, že úlohu 2 řešíme simplexovou metodou, je možno po snadné úpravě provádět „simultánní" výpočet většího počtu úloh, které se vzájemně liší pouze pravou stranou omezujících podmínek.

Abychom mohli poukázat na souvislost mezi metodami lineárního a dynamického programování pro řešení zadané úlohy o nalezení optimálního řízení markovského řetězce, budeme na úlohu 2 aplikovat revidovanou simplexovou metodu.

Jestliže v i-tém stavu zvolíme strategii k_i, potom pro simplexové násobitele v_i platí (využíváme vztahu (D.4) dodatku 2):

$$\sum_{j=1}^{n} (\delta_{ij} - p_{ij}^{k_i}) \cdot v_j = c_{i,k_i} \quad (i = 1, 2, \ldots, n) . \tag{10}$$

Úpravou těchto rovnic dostáváme:

$$v_i = c_{i,k_i} + \sum_{j=1}^{n} p_{ij}^{k_i} \cdot v_j \quad (i = 1, 2, \ldots, n) , \tag{10*}$$

což je shodný tvar s rovnicemi pro určení váhových koeficientů v bodě 1 algoritmu 2.

Hodnoty účelové funkce $h(r, n + 1)$ můžeme vyčíslit též pomocí simplexových násobitelů (viz vztah (D.6) dodatku 3). V našem případě dostáváme $h(r, n + 1) = v_r$, což plně odpovídá významu váhových koeficientů jednotlivých stavů v algoritmech 1

a 2. Při řešení úlohy 2 pomocí revidované simplexové metody je proto zbytečné vypočítávat hodnoty účelové funkce běžným způsobem a stačí pouze vypočítávat hodnoty simplexových násobitelů.

Zlepšování přípustného řešení v revidované simplexové metodě lze dosáhnout zavedením do báze té nové proměnné (využíváme vztahu (D.5) dodatku 3), pro kterou je následující výraz záporný:

$$c_{ik} - \sum_{j=1}^{n} (\delta_{ij} - p_{ij}^{k}) \cdot v_j = c_{ik} + \sum_{j=1}^{n} p_{ij}^{k} \cdot v_j - v_i \,. \tag{11}$$

V revidované simplexové metodě se v každém iteračním kroku zavádí pouze jediná nová proměnná, která minimalizuje vztah (11), jestliže toto minimum uvažujeme přes všechna i a všechna k. Jak již bylo uvedeno, optimální řízení naší úlohy stačí hledat ve třídě „čistých" strategií a tím odpadá nutnost vyšetřovat, kterou z bázických proměnných máme vyloučit při zavedení nové proměnné do báze. Proto je možno při známých hodnotách simplexových násobitelů (neboli „váhových koeficientů") v naší úloze použít v každém iteračním kroku zlepšení existujícího řízení ve více stavech současně tak, že v každém stavu použijeme tu strategii, která minimalizuje pro daný stav (tedy pro jisté i) kriterium (11). Stačí potom minimalizovat pouze výraz $c_{ik} + \sum_{j=1}^{n} p_{ij}^{k} \cdot v_j$, což je shodný tvar s rovnicemi (4) algoritmu 2. Současně zavedení většího počtu nových proměnných do báze má za následek, že nemůžeme obdržet nové simplexové násobitele pomocí jednoduchých transformací, kterých se využívá v revidované simplexové metodě při zavedení nové proměnné do báze. V tomto případě je pak výhodnější vypočítávat nové simplexové násobitele (čili nové „váhové koeficienty") řešením soustavy lineárních algebraických rovnic z jejich definičního vztahu (10*). Tímto způsobem lze tedy odvodit algoritmus 2 jeho důsledek revidované simplexové metody lineárního programování.

V dodatku 2 je na několika konkrétních příkladech ukázáno použití simplexové a revidované simplexové metody na vyšetřovanou úlohu o nalezení optimálního řízení markovského řetězce. Tyto metody jsou porovnávány podle potřebné doby pro výpočet s popsanými metodami dynamického programování.

3. PŘÍPAD DISKONTOVANÝCH NÁKLADŮ

Uvažujme dále případ, že náklady na řízení budou diskontované. Rovnice (1), která je základní pro algoritmy 1 a 2 bude pak mít následující tvar:

$$v_i^{(S)} = \alpha \cdot \sum_{j=1}^{n} p_{ij}^{(S)} \cdot v_j^{(S)} + c_i^{(S)} \tag{1*}$$

kde α je daný diskontní faktor $(0 \leqq \alpha < 1)$.

O platnosti algoritmu 1 a algoritmu 2 v případě diskontovaných celkových nákladů lze se přesvědčit stejným způsobem jako v případě, kdy kritériem optimality jsou očekávané celkové náklady. Rovnice (2) bude potom mít tvar:

$$(2^*) \qquad v_i = \min_k \left[\alpha \cdot \sum_{j=1}^{n} p_{ij}^k \cdot v_j + c_{ik}\right] \quad (i = 1, 2, \ldots, n)$$

a pro algoritmus 2 soustava rovnic pro určení váhových koeficientů jednotlivých stavů v_i bude ve tvaru

$$(3^*) \qquad v_i = c_{ik} + \alpha \cdot \sum_{j=1}^{n} p_{ij}^k \cdot v_j \quad (i = 1, 2, \ldots, n)$$

a rovnice pro nalezení jednotlivých strategií při zlepšování řízení budou následovné:

$$(4^*) \qquad \min_k \left[c_{ik} + \alpha \cdot \sum_{j=1}^{n} p_{ij}^k \cdot v_j\right] \quad (i = 1, 2, \ldots, n).$$

Uvažujme dále, že při kritériu celkového diskontovaného zisku bude pro všechna i platit $p_{i,n+1} \to 0$. Protože pro jednotlivé p_{ij} (které jsou prvky stochastické matice) platí $\sum_{j=1}^{n} p_{ij} \leqq 1$ a jelikož $0 \leqq \alpha < 1$, budou pro tento případ a to i v limitě, kdy $p_{i,n+1} = 0$ splněny podmínky $\sum_{j=1}^{n} \alpha \cdot p_{ij} < 1$ pro všechna i. Za těchto podmínek se lze snadno přesvědčit, že pomocí algoritmů 1 a 2 i v tomto případě obdržíme hledané optimální řízení. V tomto limitním případě jde o řízení markovského řetězce při kriteriu celkového diskontovaného zisku. Algoritmus 2 přechází v tomto limitním případě v algoritmus udaný Howardem v práci [6], algoritmus 1 přechází v algoritmus popsaný d'Épenouxem v práci [3].

Závěrem mi budiž dovoleno poděkovat RNDr. Petru Mandlovi, Dr. Sc., za cenné rady a připomínky k této práci.

DODATEK 1.

Důkaz konvergence algoritmu 2 k optimálnímu řízení uvažovaného markovského řetězce je možno provést obdobně k důkazu Howardova iteračního postupu pro případ celkového diskontovaného zisku (viz [6]).

Předpokládejme, že řízení (B) uvažovaného markovského řetězce je získáno zlepšováním řízení (A) podle algoritmu 2.

Podle vztahu (3) potom platí:

$$(D.1) \qquad v_i^{(B)} = c_i^{(B)} + \sum_{j=1}^{n} p_{ij}^{(B)} \cdot v_j^{(B)}; \quad v_i^{(A)} = c_i^{(A)} + \sum_{j=1}^{n} p_{ij}^{(A)} \cdot v_j^{(A)}$$

a podle vztahu (4) pro $\gamma_i^{(B,A)}$ definované jako

$$\gamma_i^{(B,A)} = c_i^{(B)} + \sum_{j=1}^{n} p_{ij}^{(B)} \cdot v_j^{(A)} - c_i^{(A)} - \sum_{j=1}^{n} p_{ij}^{(A)} \cdot v_j^{(A)} \tag{D.2}$$

platí

$$\gamma_i^{(B,A)} \leqq 0 \quad \text{a aspoň pro jedno } i \quad \gamma_i^{(B,A)} < 0 \tag{D.2*}$$

(jinak by řízení (A) bylo optimální).

Pro rozdíl $v_i^{(B)}$ a $v_i^{(A)}$ dostaneme:

$$v_i^{(B)} - v_i^{(A)} = c_i^{(B)} + \sum_{j=1}^{n} p_{ij}^{(B)} \cdot v_j^{(B)} - c_i^{(A)} - \sum_{j=1}^{n} p_{ij}^{(A)} \cdot v_j^{(A)} = \tag{D.3}$$

$$= \gamma_i^{(B,A)} + \sum_{j=1}^{n} p_{ij}^{(B)} \cdot \left(v_j^{(B)} - v_j^{(A)}\right)$$

Ukážeme dále, že pro všechna i platí:

$$v_i^{(B)} - v_i^{(A)} \leqq 0 \quad \text{a že aspoň pro jedno } i \text{ jest} \quad v_i^{(B)} - v_i^{(A)} < 0 \,.$$

Předpokládejme proto, že existuje j tak, že $v_j^{(B)} - v_j^{(A)} > 0$, a označme symbolem $\mathcal{M}$ množinu stavů takových, že pro $m \in \mathcal{M}$, $v_m^{(B)} - v_m^{(A)} = \max_j \left[v_j^{(B)} - v_j^{(A)}\right]$. Zřejmě $\mathcal{M} \neq \emptyset$. Protože $\sum_{j=1}^{n} p_{ij}^{(B)} \leqq 1$, $\gamma_i^{(B,A)} \leqq 0$ rovnost (D.3) může pro $i = m \in \mathcal{M}$ být splněna pouze tehdy, jestliže

$$\gamma_m^{(B,A)} = 0 \,, \sum_{j=1}^{n} p_{mj}^{(B)} = 1 \,, \quad \text{a když} \quad p_{mj}^{(B)} = 0 \quad \text{pro} \quad j \notin \mathcal{M}$$

$$p_{mj}^{(B)} > 0 \quad \text{pro} \quad j \in \mathcal{M} \,.$$

Provedením této úvahy pro všechny prvky množiny $\mathcal{M}$ lze se přesvědčit, že pravděpodobnost přechodu z libovolného stavu množiny $\mathcal{M}$ do stavu $(n + 1)$ je rovna nule, což odporuje předpokladům úlohy.

Proto $v_m^{(B)} \leqq v_m^{(A)} \Rightarrow v_j^{(B)} \leqq v_j^{(A)}$. Aplikací této nerovnosti na vztah (D.3) a využitím (D.2*) se lze lehce přesvědčit, že aspoň pro jedno i musí být splněno $v_i^{(B)} < v_i^{(A)}$. Protože existuje pouze konečný počet řízení složených pouze z čistých strategií, algoritmus 2 konverguje v konečném počtu kroků.

Ukážeme dále, že řešení sestrojené pomocí algoritmu 2 je pro vyšetřovanou úlohu optimálním řešením. Nechť (O) je řešení nalezené podle algoritmu 2 (a tedy pro libovolné řízení (A) je pro všechna i splněno $\gamma_i^{(A,O)} \geqq 0$) a nechť existuje řízení (S) a stav j takový, že $v_j^{(S)} < v_j^{(O)}$. Označme symbolem $\mathcal{N}$ množinu stavů, pro které $m \in \mathcal{N}$ splňuje $v_m^{(S)} - v_m^{(O)} = \min_j \left[v_j^{(S)} - v_j^{(O)}\right]$. Z rovnosti (D.3) se lze obdobnými úvahami jako v předchozím případě přesvědčit o neplatnosti předpokladu $v_j^{(S)} < v_j^{(O)}$.

Uvedeme zkušenosti s jednotlivými výpočetními postupy získané řešením 5 příkladů na samočinném počítači LGP-21. Bylo hledáno takové řízení, které s minimálními očekávanými náklady převádí markovský řetězec do $(n + 1)$-ého stavu. Každá ze zkoumaných úloh byla řešena:

1. algoritmem 1 (Eatonovou-Zadehovou metodou);
2. algoritmem 2;
3. simplexovou metodou aplikovanou na úlohu 2;
4. revidovanou simplexovou metodou aplikovanou na úlohu 2.

Optimální řízení pro jednotlivé příklady je pro stručnost zapsáno ve tvaru $(O) \equiv \equiv (k_1, k_2, \ldots, k_n)$, kde k_i značí strategii použitou v i-tém stavu při optimálním řízení. Byly řešeny následující příklady:

Příklad 1. Markovský řetězec se 3 stavy, kde $n = 2$, $s_1 = 2$, $s_2 = 1$.

$$\boldsymbol{P}_1^1 = [0{,}5; 0{,}4; 0{,}1], \quad \boldsymbol{P}_1^2 = [0{,}2; 0{,}7; 0{,}1], \quad c_{11} = 30; \quad c_{12} = 10,$$

$$\boldsymbol{P}_2^1 = [0{,}5; 0{,}4; 0{,}1], \quad c_{21} = 1{,}0.$$

Optimální řízení $(O) = (2; 1)$ a pro očekávané náklady platí:

$$v_1^{(O)} = 51{,}54; \quad v_2^{(O)} = 44{,}62.$$

Příklad 2. Markovský řetězec se 3 stavy, kde $n = 2$, $s_1 = 3$; $s_2 = 2$.

$$\boldsymbol{P}_1^1 = [0{,}25; 0{,}25; 0{,}5], \quad \boldsymbol{P}_1^2 = [0{,}75; 0{,}1875; 0{,}0625],$$

$$\boldsymbol{P}_1^3 = [0{,}125; 0{,}625; 0{,}25], \quad c_{11} = 8{,}0; \quad c_{12} = 2{,}75; \quad c_{13} = 4{,}25,$$

$$\boldsymbol{P}_2^1 = [0{,}00; 0{,}5; 0{,}5], \quad \boldsymbol{P}_2^2 = [0{,}875; 0{,}0625; 0{,}0625], \quad c_{21} = 16; \quad c_{22} = 15.$$

Optimální řízení $(O) = (1; 1)$ a pro očekávané náklady platí:

$$v_1^{(O)} = 21{,}33; \quad v_2^{(O)} = 32{,}00.$$

Příklad 3. Markovský řetězec se 3 stavy, kde $n = 2$, $s_1 = 3$, $s_2 = 3$.

$$\boldsymbol{P}_1^1 = [0{,}25; 0{,}5; 0{,}25]; \quad \boldsymbol{P}_1^2 = [0{,}5; 0{,}3; 0{,}2],$$

$$\boldsymbol{P}_1^3 = [0{,}75; 0{,}1875; 0{,}0625], \quad c_{11} = 5{,}0; \quad c_{12} = 9{,}0; \quad c_{13} = 15,$$

$$\boldsymbol{P}_2^1 = [0{,}4; 0{,}4; 0{,}2], \quad \boldsymbol{P}_2^2 = [0{,}5; 0{,}2; 0{,}3],$$

$$\boldsymbol{P}_2^3 = [0{,}6; 0{,}2; 0{,}2], \quad c_{21} = 7{,}0; \quad c_{22} = 9{,}0; \quad c_{23} = 19.$$

Optimální řízení $(O) = (1; 2)$ a pro očekávané náklady platí

$$v_1^{(O)} = 24{,}29; \quad v_2^{(O)} = 26{,}43.$$

Příklad 4. Markovský řetězec se 4 stavy, kde $n = 3$, $s_1 = 2$, $s_2 = 2$, $s_3 = 3$.

$$\boldsymbol{P}_1^1 = [0{,}3; 0{,}3; 0{,}3; 0{,}1], \quad \boldsymbol{P}_1^2 = [0{,}2; 0{,}2; 0{,}2; 0{,}4], \quad c_{11} = 1{,}0; \quad c_{12} = 2{,}0,$$

$$\boldsymbol{P}_2^1 = [0{,}1; 0{,}1; 0{,}1; 0{,}7], \quad \boldsymbol{P}_2^2 = [0{,}4; 0{,}1; 0{,}1; 0{,}4], \quad c_{21} = 1{,}0; \quad c_{22} = 3{,}0,$$

$$\boldsymbol{P}_3^1 = [0{,}1; 0{,}2; 0{,}3; 0{,}4], \quad \boldsymbol{P}_3^2 = [0{,}3; 0{,}2; 0{,}1; 0{,}4],$$

$$\boldsymbol{P}_3^3 = [0{,}2; 0{,}2; 0{,}2; 0{,}4], \quad c_{31} = 0{,}8; \quad c_{32} = 0{,}7; \quad c_{33} = 4{,}0.$$

Optimální řízení $(O) \equiv (1; 1; 1)$ a pro očekávané náklady platí:

$$v_1^{(O)} = 3{,}09\,; \quad v_2^{(O)} = 1{,}68\,; \quad v_3^{(O)} = 2{,}05\,.$$

Příklad 5. Markovský řetězec s 5 stavy, kde $n = 4$, $s_1 = 3$, $s_2 = 2$, $s_3 = 2$, $s_4 = 3$.

$$\boldsymbol{P}_1^1 = [0{,}1; 0{,}1; 0{,}1; 0{,}1; 0{,}6]\,, \quad \boldsymbol{P}_1^2 = [0{,}1; 0{,}0; 0{,}5; 0{,}1; 0{,}3]\,,$$
$$\boldsymbol{P}_1^3 = [0{,}2; 0{,}0; 0{,}1; 0{,}2; 0{,}5]\,, \quad c_{11} = 5{,}0\,; \quad c_{12} = 6{,}0\,; \quad c_{13} = 0{,}2\,,$$
$$\boldsymbol{P}_2^1 = [0{,}2; 0{,}3; 0{,}0; 0{,}0; 0{,}5]\,. \quad \boldsymbol{P}_2^2 = [0{,}0; 0{,}0; 0{,}2; 0{,}3; 0{,}5]\,,$$
$$c_{21} = 3{,}0\,; \quad c_{22} = 7{,}0\,,$$
$$\boldsymbol{P}_3^1 = [0{,}1; 0{,}3; 0{,}2; 0{,}0; 0{,}4]\,, \quad \boldsymbol{P}_3^2 = [0{,}1; 0{,}2; 0{,}0; 0{,}1; 0{,}6]\,,$$
$$c_{31} = 4{,}0\,; \quad c_{32} = 0{,}9\,,$$
$$\boldsymbol{P}_4^1 = [0{,}2; 0{,}3; 0{,}0; 0{,}1; 0{,}4]\,, \quad \boldsymbol{P}_4^2 = [0{,}2; 0{,}3; 0{,}0; 0{,}1; 0{,}4]\,,$$
$$\boldsymbol{P}_4^3 = [0{,}0; 0{,}5; 0{,}2; 0.3; 0{,}0]\,, \quad c_{31} = 1{,}0\,; \quad c_{32} = 0{,}3\,; \quad c_{33} = 8{,}0\,.$$

Optimální řízení $(O) \equiv (3; 1; 2; 2)$ a pro očekávané náklady platí:

$$v_1^{(O)} = 1{,}04\,; \quad v_2^{(O)} = 4{,}58\,; \quad v_3^{(O)} = 2{,}13\,; \quad v_4^{(O)} = 2{,}09\,.$$

Jednotlivé metody byly naprogramovány v jazyce ACT-V. Potřebné doby pro výpočet jednotlivých příkladů jsou uvedeny v závislosti na použité metodě v tabulce 1.

Tabulka 1.

Závislost doby výpočtu jednotlivých úloh na použité metodě řešení

	Algoritmus 2	Algoritmus 1	Simplexová metoda	Revidovaná simplexová metoda
Příklad 1	6 min. 100%	rel. chyba 20%	11 min. 180%	7 min. 120%
Příklad 2	7 min. 100%	rel. chyba 2%	13 min. 190%	8,5 min. 120%
Příklad 3	7 min. 100%	rel. chyba 7%	12 min. 170%	8 min. 110%
Příklad 4	18 min. 100%	rel. chyba 3%	25 min. 140%	19 min. 110%
Příklad 5	22 min. 100%	rel. chyba 20%	51 min. 230%	36 min. 160%

(Mimo absolutní doby potřebné pro výpočet jsou udány též procentuální závislosti vztažené k době výpočtu každého příkladu podle algoritmu 2. Algoritmus 1 je hodnocen maximální relativní chybou váhových koeficientů v době, kdy algoritmem 2 bylo nalezeno přesné řešení téže úlohy.)

Jak je patrno z této tabulky, nejvýhodnějším postupem se jeví algoritmus 2. Dobré výsledky rovněž dává revidovaná simplexová metoda. Eatonův-Zadehův algoritmus je vhodný pouze pro přibližné zjištění očekávaných nákladů a jeho konvergence je silně ovlivněna výchozími hodnotami váhových koeficientů i vlastnostmi vyšetřované úlohy.

DODATEK 3

Zde připomeneme některé známé vztahy pro revidovanou simplexovou metodu, kterých využíváme v této práci. Tyto vztahy lze nalézt na př. v [14].

Uvažujme úlohu lineárního programování:

Za podmínek:

$$x_i \geqq 0$$

$$\sum_{j=1}^{n} a_{ij} \,.\, x_j = b_i \quad \text{pro} \quad i = 1, 2, \ldots, m \,,$$

kde $m < n$ nalezněte minimum účelové funkce:

$$z = \sum_{j=1}^{n} c_j \,.\, x_j \,.$$

Předpokládejme, že uvažované bázické řešení úlohy je tvořeno proměnnými $x_1, x_2, \ldots$ $\ldots, x_m$. Potom pro simplexové násobitele, které označíme $v_1, v_2, \ldots, v_m$ platí tento definiční vztah:

$$\sum_{j=1}^{m} a_{ji} \,.\, v_j = c_i \quad \text{pro} \quad i = 1, 2, \ldots, m \,. \tag{D.4}$$

Na základě známých simplexových násobitelů je možno provést:

zlepšení existující řešení úlohy zavedením i-té proměnné do báze, jestliže pro i-tou proměnnou výraz

$$\bar{c}_i = c_i - \sum_{j=1}^{m} a_{ji} \,.\, v_j \tag{D.5}$$

je záporný;

výpočet hodnoty účelové funkce na základě vztahu:

$$z = \sum_{j=1}^{m} b_j \,.\, v_j \,. \tag{D.6}$$

(Došlo dne 20. prosince 1967.)

[1] J. H. Eaton, L. A. Zadeh: Optimal Pursuit Strategies in Discrete-State Probabilistic Systems. Journal of Basic Engineering, Ser. D *84* (březen 1962), 23—29.
[2] C. Derman: On Sequential Decisions and Markov Chains. Management Science *9* (1962), 1, 16—24.
[3] A. S. Manne: Linear Programming and Sequential Decisions. Management Science *6* (1960), 3, 259—267.
[4] P. Wolfe, G. Dantzig: Linear Programming in a Markov Chain. Operations Research *10* (1962), 5, 702—710.
[5] F. d' Épenoux: A Probabilistic Production and Inventory Problem. Management Science *10* (1963), 1, 98—108. (Též ve francouzské versi v časopise Revue Française de Recherche Opérationelle *4* (1960), 14, 3—16.)
[6] R. A. Howard: Dynamic Programming and Markov Processes. Technology Press and Wiley Press, New York 1960. (Též ruský překlad Sovetskoje radio 1965.)
[7] D. J. White: Dynamic Programming, Markov Chains and the Method of Successive Approximations. Journal of Mathem. Analysis and Applications *6* (1963), 2, 373—376.
[8] J. Mac Queen: A Modified Dynamic Programming Method for Markovian Decision Problems. Journal of Mathem. Analysis and Applications *14* (1966), 1, 38—43.
[9] G. de Ghellink: Les Problémes de Décisions Séquentielles. Cahiers du Centre d'Études de Recherche Opérationelle *2*, (1960), 2, 161—179.
[10] R. Bellman: Dynamic Programming. Princeton Univ. Press, 1957.
[11] J. G. Kemeny, J. L. Snell: Finite Markov Chains, Van Nostrand, New York 1960.
[12] M. Klein: Inspection-Maintenance-Problem under Markovian Deterioration. Management Science *9* (1962), 1, 25—32.
[13] A. Charnes, W. W. Cooper: Programming with Linear Fractional Functionals. Naval Research Logistics Quarterly *9* (1962), 3, 181—186.
[14] G. Dantzig: Linear Programming and Extensions, Princeton University Press, 1963. (Též ruský překlad, Progress 1966; slovenský překlad SVTL 1967.)

SUMMARY

On Eaton-Zadeh's Method

KAREL SLADKÝ

The paper deals with controlling of a dynamic system the behaviour of which can be described by a Markov chain. The problem is to determine such a control that enables to reach a fixed specified state from a given initial state at the minimum expected cost. It is shown that the methods for determination of the optimal control rule of the considered system based on dynamic programming can be also derived from the linear programming approach. In Appendix 2 several examples are calculated for comparing 4 methods discussed in this paper.

Ing. Karel Sladký, Ústav teorie informace a automatizace ČSAV, Vyšehradská 49, Praha 2.