Daugiažingsniai optimalaus valdymo uždaviniai: dinaminio programavimo metodas

Dinaminio programavimo metodas siejamas su amerikiečių matematiko Richardo Bellmano vardu. Metodas buvo pradėtas tyrinėti ir taikyti praėjusio amžiaus viduryje.

Metodas remiasi daugiažingsnio uždavinio išskaidymu į mažesnius uždavinius ir paties Bellmano suformuluotu **optimalumo principu**: *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

Optimalus elgesys turi tokią savybę: kokia bebūtų pradinė būsena ir pradinis sprendimas (valdymas), vėlesni sprendimai patys turi sudaryti optimalų elgesį atžvilgiu būsenos, gautos pradinių sprendimų rezultate. Suformulavus kiek vulgariau – optimalios trajektorijos „uodega“ pati yra optimali trajektorija.

Šį teiginį bandysime iliustruoti grafiškai. Tegu mėlyna linija vaizduoja optimalų kelią iš taško A į tašką B. Pakeliui yra taškas C. Bellmanas sako, kad jei mėlyna linija AB yra optimali, tai ir mėlyna dalis CB taip pat optimali. Iš tikrųjų, jei mėlyna CB nebūtų optimali, o C ir B jungtų geresnė raudona CB linija, tai iš A eitume į C mėlyna linija, o iš C į B eitume raudona linija, ir visas kelias iš A į B būtų geresnis. Tačiau juk geresnio už mėlyną liniją kelio iš A į B nėra. Todėl netiesa, kad gali egzistuoti „geresnė“ raudona linija CB.



Samprotavimas atrodo akivaizdus, tačiau jis remiasi prielaida, kad taške C esame laisvi rinktis, kuriuo – raudonu ar mėlynu keliu eiti. Tokia prielaida yra teisinga tik dėl to, kad galioja praeitoje paskaitoje suformuluoti du svarbūs daugiažingsnių uždavinių formulavimo principai. Primenu:

Būsena laikotarpio t pabaigoje priklauso tik nuo būsenos praeito laikotarpio pabaigoje ir nuo priimtų sprendimų, bet *tiesiogiai* nepriklauso nuo ankstesnių būsenų. Taigi modeliuojamo proceso „gylis“ yra tik vienas laikotarpis, kas nebūtinai atitinka tikrovę ir ką reikėtų įsidėmėti kaip uždavinio formulavimo pirmąją prielaidą.

Tikslo funkcija sumuoja atskirų atskirai įvertintų laikotarpių naudingumą (sakoma, yra *adityvi*). Tikrovėje nebūtinai taip yra, ir adityvumo prielaidą fiksuokime kaip mūsų formuluojamo uždavinio antrąją prielaidą.

Jei ne šios dvi prielaidos, Bellmano optimalumo principu būtų galima ir suabejoti. Todėl nagrinėkim daugiažingsnius optimalaus valdymo uždavinius, kuriems minėtos prielaidos galioja ir bandykime pasinaudoti jų sprendimui Bellmano optimalumo principu.

Vėl imkime elementarų praeitoje paskaitoje suformuluotą uždavinėlį, norėdami, kuo paprastesniu pavyzdžiu suprasti dinaminio programavimo esmę. Primenu šį uždavinį ir jo matematinę išraišką.

Tegu eurų reikia T metų dalinti vartojimui ir taupymui. Metų t pradžioje dalį turimų lėšų skiriame vartojimui, likusią dalį investuojame metams su grąža .

Pažymėkime metų t-1 pabaigoje turimas lėšas, o sprendimą, kiek suvartoti t metais. Tada dalis liks investicijoms, ir metų t pabaigoje turėsime lėšų, iš jų skirsime vartojimui ir t.t. Naudingumo funkcija – logaritminė: ln( yra vartojimo naudingumas. Be to, skirtingų laikotarpių naudingumo palyginimui naudokime diskontavimo koeficientą (gauti 1 kitąmet prilyginama gavimui šiemet). Užrašymų racionalizavimui žymėsime

Remdamiesi Bellmano optimalumo principu pradėsime konstruoti šio uždavinio optimalią trajektoriją nuo galo (nuo „uodegos“). Tuo tikslu apibrėšime labai svarbų visai šiai teorijai objektą – **rekurentinę Bellmano funkciją,** kurią žymėsime ir interpretuosime kaip **sumarinę maksimalią naudą per laikotarpius t+1, t+2, ... T, jei po t-tojo laikotarpio dar disponuojame lėšomis**

Šitoje vietoje ypatingai svarbu įsisąmoninti prasmę. Tik taip pavyks suprasti dinaminio programavimo ir apskritai rekursijos esmę. DP metodas labai neįprastas ir reikalauja pastangų šiek pertvarkyti savo įprastą mąstymą.

Dabar pradėkime vieną po kitos „nuo uodegos“ konstruoti rekurentines Bellmano funkcijas suformuluotam uždaviniui.

**t = T**. Kam lygi ? Tai maksimali nauda po T-tojo laikotarpio, jei liko taip ir nepanaudota lėšų. Pagal sąlygą – jokios naudos po T-tojo laikotarpio nėra. Taigi

**t=T-1.** Kam lygi ? Tai maksimali nauda po T-1-ojo laikotarpio, jei paskutiniajam T-tajam laikotarpiui dar liko nepanaudota lėšų. Taigi t.y. didžiausia galima naudingumo reikšmė, kokią galime gauti per paskutinį T-tąjį laikotarpį, jei dar turime lėšų ir vartojimui skirsime jų dalį . Kadangi yra didėjanti funkcija, tai, norint gauti maksimalią jos reikšmę, reikia imti kuo didesnį . Bet suvartoti negalime daugiau, negu turime (negu liko iš praeito laikotarpio): . Todėl maksimali reikšmė yra ir gauname

**t=T-2**. Kam lygi ? Tai maksimali nauda po T-2-ojo laikotarpio, jei dviem paskutiniams T-1 ir T laikotarpiams dar liko nepanaudota lėšų. Štai čia ir prasideda rekursija. Parašysim

Suprasti šį užrašymą – reiškia suprasti dinaminio programavimo esmę. Galvojame taip: štai disponuojame lėšomis Dalį suvartosime T-1 laikotarpiu, gausime naudą . Investavimui liks . T-1 laikotarpio pabaigoje jau turėsime . O kokią maksimalią naudą per paskutinį T laikotarpį galime turėti iš to Atsakymą jau žinome – ką tik skaičiavome, kad tą naudą įvertina jau žinoma funkcija , todėl ta nauda yra ). Belieka sudėti dabartinę (T-1 laikotarpio) naudą ir visų būsimų (šiuo atveju tai tik T) laikotarpių naudą ), neužmirštant jos diskontuoti (nes būsimas vienetas yra tik dabartinių vienetų). Gautą sumą belieka maksimizuoti pagal einamąjį sprendimą ir taip gaunama išraiška.

Konkrečios reikšmės dabar neskaičiuosime – dabar svarbu suprasti principą, o prie konkretaus sprendimo grįšime vėliau. Padarykime bendrą žingsnį ir užrašykime bendrą rekurentinę formulę mūsų uždaviniui.

**t=t-1**. Kam lygi ? Tai maksimali nauda po t-1 -ojo laikotarpio, kai laikotarpiams t, t+1, t+2,... T dar liko nepanaudota lėšų. Tęsiame rekursiją. Parašysim

Dar sykį apgalvojame: štai t laikotarpio pradžiai (t-1 pabaigai) disponuojame lėšomis Dalį suvartosime t laikotarpiu, gausime naudą . Investavimui liks . t-tojo laikotarpio pabaigoje jau turėsime lėšų. O kokią maksimalią naudą per laikotarpius t+1, t+2, ... T galime turėti iš to Atsakymą jau žinome – apskaičiavome, kad tą naudą įvertina jau žinoma funkcija , todėl ta nauda yra ). Belieka sudėti dabartinę (t-tojo laikotarpio) naudą ir visų būsimų (t+1, t+2, ... T) laikotarpių naudą ), neužmirštant jos diskontuoti. Gautą sumą belieka maksimizuoti pagal einamąjį sprendimą ir taip gauti išraišką.

**t=0.** Taip eidami atatupstom prieiname ir apskaičiuojame funkciją . Ką ji mums sako? Ogi tai yra sumarinė maksimali nauda per laikotarpius 1, 2, ... T, jei proceso pradžioje disponuojame lėšomis Belieka apskaičiuoti , tai ir bus ieškoma maksimali tikslo funkcijos reikšmė. Tada grįšime per visą „atbulom“ praeitą procesą ir išsiaiškinsime, kokių sprendimų dėka gavome šią maksimalią reikšmę. Uždavinys bus išspręstas.

Dinaminio programavimo metodu gali būti sprendžiami labai įvairūs daugiažingsnio optimizavimo uždaviniai, tačiau visais atvejais svarbiausias elementas – rekurentinė Bellmano funkcija ir jos apskaičiavimas. Tas apskaičiavimas anaiptol nėra paprastas. Skirčiau čia dvi galimybes:

* analitinių išraiškų gavimas (kitaip tariant, užrašymas tam tikromis formulėmis). Tai pavyksta anaiptol ne visada, tiesiogiai skaičiuojant išraiškas žingsnis po žingsnio formulės tampa labai sudėtingos. Kartais čia gali padėti vadinamas neišreikštinių koeficientų metodas, kitais kartais mus gali dominti ne tiek pačios optimalios trajektorijos, kiek stacionarios būsenos (*steady state*), į kurias begaliniame laike tos trajektorijos konverguoja. Tokias būsenas rekurentinės formulės taip pat gali padėti įvertinti;
* rekurentinių funkcijų skaičiavimas lentelėmis (tabuliavimas). Šiuo atveju mums nereikia analitinių išraiškų, o daugiažingsnį optimalaus valdymo uždavinį sudarančios funkcijos gali būti netolydžios, nediferencijuojamos, apskritai užduodamos tik tam tikruose taškuose lentelėmis ir sveikaskaitinės. Tradiciniai Lagrange‘o daugikliais besiremiantys metodai čia būtų visiškai bejėgiai, o dinaminio programavimo metodas gali pasiūlyti puikiausią sprendimą.

Imkime keletą pavyzdžių, iliustruojančių tai, kas pasakyta.

**1.1.Tiesioginis rekurentinių formulių skaičiavimas – analitinės išraiškos.**

Kaip pavyzdį imkime aukščiau mūsų pradėtą nagrinėti uždavinį. Gavome tokias rekurentinių formulių išraiškas:

Norėdami gauti analitinę išraišką, įstatome aukščiau gautą išraišką

Dabar reikia maksimizuoti pagal u skliaustuose { } esantį reiškinį. Skaičiuojame jo išvestinę pagal u ir prilyginam nuliui (dėl logaritmo įgaubtumo tikrai turime maksimumą):

Iš čia Todėl . Kadangi skaičiuojame , tai esame T-1 laikotarpyje, vadinasi, sprendimas, kiek suvartoti, yra T-1 laikotarpio sprendimas, o skirstome iš praeito laikotarpio atėjusiais lėšas . (Beje, akivaizdu, kad , kitaip toks maksimizavimas nebūtų teisėtas)

Dabar optimizuojančią lygybę reikia įstatyti į išraišką ir gausime

Čia yra nariai, nepriklausantys nuo ; konkrečios jų išraiškos ateityje neprireiks, todėl nepasivarginta juos konkrečiai apskaičiuoti.

Bandome tęsti tiesioginį rekurentinių formulių skaičiavimą. Į įstatysim gautą išraišką:

Dabar vėl skaičiuosim pagal u skliaustuose { } esančio reiškinio išvestinę, prilyginsim nuliui ir gausim maksimumą. Patikrinkit, kad dabar optimizavimo sąlyga bus , todėl

Taip tęsdami toliau, gausime, kad

ką kitais metodais jau esame gavę anksčiau. išraišką būtų galima apskaičiuoti, bet ji pernelyg sudėtinga, o optimalią trajektoriją randame ir be jos.

Kaip matome, tiesioginis rekurentinių formulių skaičiavimas nėra pats lengviausias dalykas, o bent kiek sudėtingesniame uždavinyje būtų ir visai neįmanomas. Kartais išeitį gali pasiūlyti vadinamas neapibrėžtinių koeficientų metodas, kurį pailiustruosime vėlgi to paties uždavinėlio pavyzdžiu.

**1.2. Neapibrėžtinių koeficientų metodas – analitinės išraiškos.**

Nagrinėdami gautas išraiškas, matome, kad jas sudaro du nariai – vienas, priklausomas nuo , jo pavidalas ir kitas – nepriklausantis nuo . Todėl padarome prielaidą, kad

Čia kaip tik ir yra tie neapibrėžtiniai koeficientai, kurių pagalba bandysime gauti reikalingas formules. Iš aukščiau gautų ir išraiškų matome, kad

Dabar imame bendrą rekurentinę formulę

Ir tikriname, ar prielaida dėl pasiteisins. Kitaip tariant, ar pavyks išreikšti tuo pat mūsų pasirinktu pavidalu. Skaičiuojam įstatę išraišką

Vėl maksimizuojame skliaustuose { } esančią išraišką, skaičiuodami jos išvestinę pagal u ir prilygindami ją nuliui:

Ši reikšmė maksimizuoja skliaustuose { } esantį reiškinį, todėl ją įstatome į išraišką (indeksų t , t-1 prie x ir u nerašome dėl formulių užrašymo paprastumo).

Dabar, naudodamiesi logaritmavimo taisyklėmis, atskirsime narius prie *ln(x)* ir visus kitus, kur nefigūruoja. Gausim

(

Kas netingi, gali apskaičiuoti (...) pažymėtą konglomeratą, bet mums svarbu tik tiek, kad tai nuo nepriklausantis reiškinys, kuris ir yra mūsų prielaidos . Reiškinys, kuris priklauso nuo, yra pavidalo , kuo mes ir siekėme įsitikinti. Prilyginę

matome, kad pavyko išreikšti pavidalu

vadinasi, mūsų prielaida pasiteisino. Dar ir labai pasisekė (dėl uždavinėlio paprastumo), kad rekurentiškai priklauso tik nuo , bet nepriklauso nuo (kas anaiptol ne kiekviename uždavinyje nutinka) todėl sekos skaičiuoti nereikia, nebent norėtume gauti konkrečias išraiškas. Apskaičiuosime seką , remdamiesi aukščiau gauta rekurentine išraiška .

ir taip toliau. Pagaliau gauname

Kadangi jau esame išsiaiškinę, kad optimalioje trajektorijoje , tai nesunkiai randame, kad

ką jau esame gavę ir anksčiau. Visas kitas reikšmes nesunkiai apskaičiuosime iš gautų formulių.

**1.3 Stacionarios būsenos (*steady state*) radimas.**

Kaip jau minėjau, tyrėją dažnai domina ne tiek pačios kintamųjų trajektorijos, kiek tai, į ką jos (arba svarbūs kintamųjų santykiai) konverguoja per pakankamai ilgą laiką. Priimdami, kad ir dėl to , pagrindinę mūsų rekurentinę Bellmano lygtį užrašysime taip:

Tai visiškai rekurentinė lygtis: funkcija išreiškiama pati per save. Ir kaip tokią lygtį išspręsti? Vienas iš būdų -bandyti atspėti galimą jos pavidalą ir įsitikinti spėjimo teisingumu. Kadangi mūsų uždavinėlį jau esame visaip tyrinėję, tai nebus netikėtas spėjimas, kad

Įstatome šį spėjimą į mūsų lygtį

Vėl maksimizuojame { } skliaustuose esantį reiškinį, skaičiuodami išvestinę pagal u:

Gautą optimalią reikšmę įstatom į pradinę lygtį

Vėl surenkam narius prie ir atskirai visus likusius

Matome, kad spėjimas apie pavidalą pasiteisino. Kaip ir aukščiau, kas netingi, gali pasivarginti apskaičiuoti (...) paliktą reiškinį, bet tolesnei eigai mums labiau rūpi reikšmė. Kadangi gautoji formulė turi galioti visiems , tai neišvengiamai kairėje ir dešinėje prie esantys dydžiai turi sutapti, t.y. . Todėl

Iš čia galime gauti ir kitus mus dominančius dydžius. Stacionarios būsenos atveju šiame uždavinėlyje tikriausiai labiausiai dominantis dalykas yra santykis , t.y. kokia turimų lėšų dalis stacionariame procese turėtų būti skiriama vartojimui (o kokia likti investavimui). Iš gautų formulių lengvai matome, kad

Šį rezultatą esame gavę ir anksčiau (taikydami Eulerio lygtį).

Pastaba. Atvejį galėjome gauti ir iš 1.2 punkte gautų išraiškų, perėję prie ribos, tačiau 1.3 skirsnis čia buvo parodytas kaip pavyzdys, kaip daryti, jei iš karto mus domintų *steady state*, o ne konkrečių trajektorijų skaičiavimas.

**Užduotis.** Pabandykite pritaikyti 1.2 ir 1.3 punktuose išdėstytą sprendimo būdą praeitą kartą suformuluotam uždaviniui

**Rekurentinė Bellmano lygtis bendram daugiažingsnio optimizavimo uždaviniui**

Dabar, kai jau patyrinėjome įvairiais aspektais paprastą pavyzdį, galime pereiti prie bendrojo uždavinio. Jį formulavome taip:

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |
|  | (3) |
|  | (4) |

Rekurentinė Bellmano funkcija, kurią žymėsime bendru atveju išreiškia **sumarinę maksimalią tikslo funkcijos reikšmę per laikotarpius t+1, t+2, ... T, jei po t-tojo laikotarpio modeliuojamo sistemos būsena yra .** Ją užrašysime taip:

Laikotarpių t, t+1, ...T

maksimali nauda

Visų likusių (t+1, t+2 , ... T) laikotarpių maksimali „nauda“

Laikotarpio t „nauda“

Uždavinio sprendimas vyksta, skaičiuojant vieną po kitos funkcijas ir įvertinant, kokių sprendimų (valdymų) dėka gaunama optimali trajektorija (žr. paprasčiausią pavyzdį aukščiau).

Deja, tenka pasakyti, kad rekurentinių funkcijų skaičiavimas anaiptol ne visada pavyksta taip, kaip buvo parodyta labai paprasto pavyzdžio atveju. Tačiau pats rekursijos principas, prieinant kūrybiškai, gali būti labai naudingas ir įdomus. Toliau bus pateikiamas vienas iš daugelio galimų pavyzdžių – McCall tarplaikinės darbo paieškos modelis.

Tačiau dinaminio programavimo principas gali pasirodyti labai naudingas, kai susiduriame su uždaviniais, kuriuose apskritai apie kokį nors išvestinių skaičiavimą ir analitinių išraiškų gavimą negali būti nė kalbos. Įdomu tai, kad kintamųjų sveikaskaitiškumas, funkcijų apibrėžimas lentelėmis ir pan. gali būti neįveikiama kliūtis tradiciniams metodams, tačiau dinaminio programavimo metodui tai kaip tik yra sprendimą palengvinantys privalumai.

Imkime konkretų, vėlgi maksimaliai supaprastintą uždavinio pavyzdį.

**2.1 Rekurentinių Bellmano funkcijų tabuliavimo pavyzdys**

Penkis įrengimų kompleksus reikia paskirstyti trims įmonėms. Nauda, kurią gauname iš kiekvienos įmonės, kai jai skiriamas atitinkamas įrengimų kompleksų skaičius, parodyta lentelėje. (Pavyzdžiui, jei II įmonė gaus 4 įrengimų kompleksus, tai nauda bus 8 milijonai eurų.)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** |
| III įmone | 0 | 3 | 5 | 6 | 7 | 8 |
| II įmone | 0 | 2 | 4 | 6 | 8 | 10 |
| I įmone | 0 | 4 | 7 | 10 | 10 | 8 |
|  |  |  |  |  |  |  |

Reikia taip paskirstyti įrengimus, kad sumarinė iš visų įmonių gauta nauda būtų maksimali.

Samprotaukime daugiažingsnio dinaminio uždavinio terminais (nors pats uždavinys ir nėra išdėstytas laike). *Valdymo* kintamasis šiuo atveju aiškus: tai t-tajai įmonei skiriamas įrenginių skaičius. Sakykim, reikštų, kad nusprendėme I įmonei skirti 3 įrengimus.

Tikslo funkcija rodytų t-tosios įmonės duodamą naudą, jei šiai įmonei būtų skiriama įrengimų. Šiuo atveju funkcijos yra tik trys ir jų reikšmės užduotos lentele. Pavyzdžiui, Apie jokias tokių funkcijų išvestines negali būti nė kalbos, todėl mūsų anksčiau nagrinėtas tyrimo būdas (kaip ir tradiciniai su Lagrange‘o daugikliais susiję būdai) šiuo atveju visiškai bejėgiai.

Norint galutinai suformuluoti daugiažingsnį dinaminio optimizavimo uždavinį, dar reikia apibrėžti *būsenos* kintamąjį. Tuo tikslu įsivaizduokim įrengimų dalinimą įmonei kaip daugiažingsnį procesą: turim iš viso įrengimus ir pirmu sprendimu (laikotarpiu) pirmai įmonei atidavėm įrengimų, tada antram sprendimui (laikotarpiui) mums dar liko nepaskirstyta įrengimų ir t.t. Taigi būsenos kintamasis reikštų, kiek dar liko nepaskirstyta įrengimų po to, kai 1, 2 , ..., t įmonės jau įrengimais apdalintos. Dabar galime užrašyti dinaminį uždavinį

Uždavinys užrašytas abstrakčiam įmonių skaičiui; mūsų atveju turime T=3,

Kadangi spręsime šį uždavinį dinaminio programavimo metodu, turime jam užrašyti rekurentinę Bellmano funkciją. Šiuo atveju interpretuosime kaip **sumarinę maksimalią naudą, gaunamą iš įmonių t+1, t+2, ... T, jei po t-tosios įmonės dar liko nepaskirstytų įrengimų.** Nesunku įsitikinti, kad

Tačiau kaip dabar apskaičiuoti funkcijas Vėl pradėkime nuo galo, nuo , šio uždavinio atveju tai . Kadangi sąlygoje pasakyta, kad ketvirtos įmonės nėra, likusių nepaskirstytų įrengimų parduoti negalime, tai po trijų įmonių likusių įrengimų jokios naudos neatneš ir belieka konstatuoti, kad

Traukiamės į . Tai nauda iš paskutinės, trečiosios įmonės, jei po pirmų dviejų jai dar liko įrengimų. Ta nauda yra užduota lentele. Būsenos kintamojo reikšmė gali būti bet kokia nuo 0 iki 5, nes gal pirmoms dviem įmonėms atidavėme visus įrengimus, o gal nė vieno. Taigi reikšmes parodo tokia lentelė

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** |
|  | 0 | 3 | 5 | 6 | 7 | 8 |

Dabar traukiamės į ir skaičiuojame maksimalią naudą iš dviejų paskutinių įmonių, jei po pirmosios dar liko nepaskirstyta įrengimų. Pirmiausia išsiaiškiname galimas reikšmės. Ir vėl tai gali būti bet kokia reikšmė nuo 0 iki 5, nes gal pirmai įmonei atidavėme visus įrengimus, o gal nė vieno. Tačiau jei ankstesnėje lentelėje sprendimas buvo aiškus – atiduoti visus įrengimus paskutinei trečiai įmonei, tai dabar jau reikia spręsti, kiek iš likusių įrengimų atiduosime antrai įmonei. Todėl skaičiavimo lentelė atrodys sudėtingiau

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|   | **0** | **1** | **2** | **3** | **4** | **5** |
| 0 | *0+0* | *0+3* | ***0+5*** | 0+6 | 0+7 | 0+8 |
| 1 |  | 2+0 | ***2+3*** | *2+5* | 2+6 | 2+7 |
| 2 |  |  | 4+0 | *4+3* | *4+5* | 4+6 |
| 3 |  |  |  | 6+0 | *6+3* | *6+5* |
| 4 |  |  |  |  | 8+0 | *8+3* |
| 5 |  |  |  |  |  | 10+0 |
|  | **0** | **3** | **5** | **7** | **9** | **11** |

Panagrinėkim, kaip sudaryta ši lentelė. Imkim stulpelį, kuriame   Tai reiškia, kad po pirmos įmonės dar liko 3 nepaskirstyti įrengimai. Jei antrai įmonei atiduosim 1 įrengimą, tai iš antros įmonės nauda bus 2 (žr. sąlygą), o du įrengimai pereis tolesniam skirstymui, iš kurio naudos gausime 5 (žr. jau apskaičiuotą aukščiau lentelę). Šiame geltonai atžymėtame langelyje apskaičiuota ne kas kita, kaip rekursinės formulės fragmentas .

Imkim stulpelį, kuriame   Tai reiškia, kad po pirmos įmonės dar liko 5 nepaskirstyti įrengimai. Jei antrai įmonei atiduosim 4 įrengimus, tai iš antros įmonės nauda bus 8 (žr. sąlygą), o vienas įrengimas liks tolesniam skirstymui; iš jo naudos gausime 3 (žr. jau apskaičiuotą aukščiau lentelę). Šiame žaliai atžymėtame langelyje apskaičiuota ne kas kita, kaip rekursinės formulės fragmentas .

Kairiajame apatiniame kampe langeliai lieka tušti, nes negalima įmonei skirti daugiau įrengimų, negu jų iš viso liko.

Užpildę šią lentelę, sužinome, kaip reikia optimaliai pasielgti, jei po pirmos įmonės liks atitinkamas skaičius įrengimų. Vėl imkim stulpelį. Jei po pimos įmonės bus likę trys įrengimai, tai nieko neskyrę antrai, per dvi likusiais įmones laimėsim tik 6. Bet jei skirsime vieną ar du įrengimus, laimėsime 7. Jei skirsime visus tris – vėl bus tik blogiau, laimėsime 6. Todėl optimalu skirti vieną ar du, o reikšmė (maksimali nauda per dvi likusiais įmones, jei po pirmosios dar liko 3 įrengimai) yra 7. Tą patį padarome su visais stulpeliais – kaip reikšmę išrenkame kiekvieno stulpelio maksimumą.

Apskaičiavę , galime trauktis į . Kadangi pradinis skirstomų įrengimų skaičius žinomas, tai iš tikrųjų mums tereikia apskaičiuoti reikšmę. Lentelė dabar atrodo taip:

|  |  |
| --- | --- |
|   | **5** |
| 0 | 0+11 |
| 1 | 4+9 |
| 2 | 7+7 |
| 3 | ***10+5*** |
| 4 | 10+3 |
| 5 | 8+0 |
|  | **15** |

Lentelė sudaryta panašiai, kaip ir ankstesnė, tik nereikia 6 stulpelių, nes pradinis įrengimų skaičius žinomas Jei pirmai įmonei atiduosim 3 įrengimus, tai iš pirmos įmonės nauda bus 10 (žr. sąlygą), o du įrengimai liks tolesniam skirstymui; iš jų naudos gausime 5 (žr. jau apskaičiuotą reikšmių lentelę). Šiame geltonai atžymėtame langelyje apskaičiuota ne kas kita, kaip rekursinės formulės fragmentas .

Iš lentelės aišku, kad jei pirmai įmonei neskirsim įrengimų, bendra nauda bus tik 11, jei skirsim vieną – nauda bus 13 ir t.t. Todėl optimalu pirmai įmonei skirti tris įrengimus (3), tada nauda iš visų trijų įmonių lygi 15 ir yra maksimali. Belieka atsekti likusius du sprendimus. Kadangi iš 5 įrengimų tris atidavėm pirmai įmonei, tai likusioms dviem liko tik du. Pereinam į ankstesnę lentelę, stulpelį   Ten mūsų laimėjimas 5, ir jį galima pasiekti dviem būdais: nieko neskirti antrai įmonei ir viską atiduoti trečiai arba paskirstyti po vieną įrengimą abiem likusioms įmonėms. Abi strategijos garantuoja 15 naudą, todėl turime du optimalius planus.

Dar vienas įdomus tabuliavimo pavyzdys – įrengimo pakeitimo uždavinys. Turime įrengimą, kurį naudojame t = 1, 2 , ..., T laikotarpiais. Žinoma, kad x pilnų laikotarpių atitarnavęs įrengimas duoda naudą v(x) eurų. Kadangi didėjant įrengimo amžiui x, jo duodama nauda vis mažėja ir eksploatavimas net gali tapti nuostolingas, kiekvieno laikotarpio pradžioje įrengimą galima pakeisti nauju. Seną galima parduoti už s(x) eurų, o naujas kainuoja p eurų. Nupirktas naujas įrengimas iš karto pirmuoju laikotarpiu ima duoti naudą v(0), nes yra „nulinio“ amžiaus.

Kaip dažnai keisti įrengimą 12 laikotarpių periode? Sąlygos funkcijos užduodamos lentelėmis

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 0 | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| p |  | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| s |  | 6 | 5 | 4 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| v | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | 0 |

Šiam uždaviniui iš karto užrašysime rekurentinę Bellmano funkciją. Verta atkreipti dėmesį, kad šį kartą ji užrašoma gerokai kitaip, negu ankstesniuose pavyzdžiuose. Tegu yra nauda, kurią gausime iš įrengimo per laikotarpius t, t+1, .... T, jei po t-tojo laikotarpio įrengimo amžius yra (tiek pilnų laikotarpių jau atitarnavo).

Kitaip tariant, yra didesnioji nauda iš dviejų alternatyvų: jei įrengimo nekeičiam, jis t laikotarpyje duoda naudą ir į tolesnius laikotarpius pereina vienu laikotarpiu „pasenęs“, duodamas ten naudą ; jei įrengimą keičiam, jį parduodam, perkam naują, t laikotarpyje gaunam naudą ir į tolesnius laikotarpius įrengimas pereina jau amžiaus 1, duodamas ten naudą .

Mėginkime tabuliuoti (skaičiuoti lentele) šį uždavinį.

Šis uždavinys gali pasitarnauti pavyzdžiu dar vienam įdomiam dinaminio programavimo taikymo variantui su begaliniu laiku iliustruoti.

Įrengimas naudojamas laikotarpiais t = 1, 2 ,3, ... Begalinis laikas reiškia, kad įrengimą tiesiog ketinama naudoti nuolat, nepaskiriant konkretaus veiklos pabaigos laikotarpio. Tačiau kas kažkokį laiką įrengimą reikia keisti nauju.

Naujo įrengimo kaina eurų; seną, atitarnavusį pilnų laikotarpių, galima parduoti už eurų. Įrengimo, atitarnavusio pilnų laikotarpių duodama nauda skaičiuojama kaip eurų (kuo didesnis , tuo ta nauda mažesnė. Būsimo laikotarpio naudingumui su esamojo *t* laikotarpio naudingumu palyginimui naudojamas diskontavimo koeficientas

Kaip dažnai reikėtų keisti įrengimą, kad suminė nauda būtų didžiausia?

Skirtingai, negu anksčiau, dinaminio programavimo procesą pradėsime ne nuo pabaigos, o nuo pradžios (nes pabaiga be galo nutolusi). Tegu yra nauda per laikotarpius , t.y. per laikotarpis po *t*-tojo (kaip ir žymėjome anksčiau). Taigi pradedam nuo . Skaičiuojame, kad įrengimą reikia nupirkti, būdamas visai naujas jis duos naudą , o nauda po pirmoji laikotarpio bus . Todėl

Jei įrengimo po pirmojo laikotarpio antrojo laikotarpio pradžioje nepakeisime, tada

Jei nepakeisime ir po antrojo, tada

Taip skaičiuodami, matome, kad jei įrengimo nepakeisime po k-1 -jo laikotarpio, tai

Tačiau dabar nuspręskime k laikotarpių atititarnavusį įrengimą pakeisti. Todėl

(parduodame seną už perkame naują už , eksploatuojame jau naują, gaudami naudą , o nuo laikotarpio po k-tojo nauda ). Toliau istorija kartojasi

Įrengimą vėl keičiame po k laikotarpių

Ir taip toliau visą begalinį procesą. Dabar galime suskaičiuoti, kad

Turime laužtiniuose skliaustuose pasikartojantį narį su daugikliais ir t.t., sudarančiais geometrinę progresiją. Todėl pagaliau

Šį reiškinį turėtume maksimizuoti pagal *k*. Jei sąlyga prasminga, galima tikėtis, kad toks maksimumas egzistuoja: nei per retai, nei per dažnai keisti įrengimą neapsimokės.

Pavyzdžiui, jei , tada atitinkamos reikšmės:

|  |  |
| --- | --- |
| k=1 | 44,00 |
| k=2 | 217,73 |
| k=3 | 259,62 |
| k=4 | **268,69** |
| k=5 | 264,71 |
| k=6 | 246,55 |
| k=7 | 240,29 |

Vadinasi, įrengimą geriausia keisti atitarnavus 4 laikotarpius

|  |  |  |
| --- | --- | --- |
|  | () | () |
| 0 | 10 |  |
| 1 | 9 | 6 |
| 2 | 8 | 5 |
| 3 | 7 | 4 |
| 4 | 6 | 3 |
| 5 | 5 | 2 |
| 6 | 4 | 0 |
| 7 | 3 | 0 |
| 8 | 2 | 0 |
| 9 | 1 | 0 |
| 10 | 0 | 0 |