Qoşqar Seyfullah oğlu ƏLİYEV
Qərbi Kaspi Universiteti, t.ü.f.d., dosent

Ləman İlham qızı BƏŞİRZADƏ
Qərbi Kaspi Universiteti, magistr

DİNAMİK PROQRAMLAŞDIRMA ÜSULUNUN İDARƏETMƏDƏ TƏTBİQİ MƏSƏLƏLƏRİ

Xülasə: Optimallaşdırma problemlərinə elm, texnologiya və iqtisadiyyatın demək olar ki, bütün sahələrində rast gəlinir. Müasir idarəetmə nəzəriyyəsində riyazi proqramlaşdırmanın əsasını təşkil edən optimallaşdırma metodlarından geniş istifadə olunur ki, bu da təqdim edilən mövzunu aktual edir. Məqalədə bu problemin tədqiqinin nəticələri və müəssisənin idarə edilməsi sahəsində dinamik proqramlaşdırma metodlarının tətbiqi təqdim olunur.
Açar sözlər: Dinamik proqramlaşdırma, qərarların qəbul edilməsi, optimal idarəetmə.

UOT: 004.424
JEL: C88
DOI: doi.org/10.54414/whgj6475

Giriş

Dinamik proqramlaşdırma hər bir qərarın nəticələrinin hesablanması və sonrakı qərarlar üçün optimal strategiyanın işlənib hazırlanması əsasında optimal həllərin tapılmasına imkan verən texnika və metodlar toplusunu öyrənən riyazi proqramlaşdırmanın bir sahəsidir. Dinamik proqramlaşdırma məsələləri çoxmərhələlidir, ona görə də "dinamik proqramlaşdırma" termini təkcə məsələlərin xüsusi növünü müəyyən etmir, həm də riyazi proqramlaşdırma məsələlərinin ayrı-ayrı siniflərinin həlli yollarının tapılması üsullarını xarakterizə edir.

«Dinamik proqramlaşdırma» ifadəsi ilk dəfə 1940-cı illərdə Riçard Bellman tərəfindən problemin həlli prosesini təsvir etmək üçün istifadə edilmişdir, burada bir problemin cavabı yalnız ondan «əvvəlki» problem həll edildikdən sonra əldə edilə bilər.

Müasir cəmiyyətdə informasiyanın həcmi planetin 20 il əvvəl mövcud olan bütün informasiya həcmini üstələyir. Bütün daxil olan məlumatların emalı çox vaxt aparır. Riyaziyyat və proqramlaşdırma baxımından inkişaf edir, getdikcə mürəkkəbləşirlər. Sürətli və keyfiyyətli informasiyanın emalı prosesi çətin məsələdir. Mürəkkəb tapşırıqları bir çox sadə tapşırıqlara bölməklə, maksimum nəticə əldə etmək üçün hesablamalara və lazımi proseslərin işlənməsinə sərf olunan vaxtı azaldırıq. Əgər bir mərhələli proseslər üçün qəbul edilən qərarlar, bir qayda olaraq, nisbətən sadədirsə, çoxmərhələli proseslərdə qərarın strukturu müqayisə olunmayacaq dərəcədə mürəkkəbdir. Bu problemləri həll etmək təkcə yeni başlayanlar üçün deyil, həm də təcrübəli proqramçı üçün çox çətindir. 1940-cı ildə Riçard Bellman ilk dəfə mürəkkəb müasir problemlərin həlli üçün ən mühüm əsas olan metodu işləyib hazırladı. "Bellman Metodu" və ya "Dinamik Proqramlaşdırma Metodu" mürəkkəb məsələləri daha sadə kəsişən alt problemlərə bölmək yolu ilə həll etmək üçün məşhur riyazi üsuldur. 1953-cü ildə R. Bellman dinamik proqramlaşdırma metodunu optimallıq prinsipindən istifadə etməklə həll edilən bir sıra konkret praktiki məsələlərə tətbiq etdi. Metodun əhatə dairəsi statik, xətti və riyazi proqramlaşdırmadan fərqli olaraq zamanla inkişaf edən çoxmərhələli proseslər idi.

Dinamik proqramlaşdırmanın əsas ideyası mühüm problemi nəzərdən keçirmək və onu daha kiçik, fərdiləşdirilmiş komponentlərə bölməkdir. Həyata keçirilməsinə gəldikdə, optimal üsullar alqoritmin səmərəliliyini artırmaq üçün məlumatların saxlanmasına və təkrar istifadəyə əsaslanır. Gördüyümüz kimi, proqram təminatının hazırlanmasında bir çox suallar dinamik proqramlaşdırmanın müxtəlif

formalarından istifadə etməklə həll olunur. Bu üsul, optimal həllərin sadə bir dəyişəndən istifadə etməklə hazırlana biləcəyini, mürəkkəb məlumat strukturu və ya alqoritm tələb etdiyini tanımaqdır.

Bizim dövrümüzdə elm təşkilat və idarəetmə məsələlərinə getdikcə daha çox diqqət yetirir ki, bu da mürəkkəb məqsədyönlü proseslərin strukturu və təşkili baxımından təhlil edilməsi zərurətinə səbəb olur. Təcrübə ehtiyacları «əməliyyat tədqiqatı» adı altında rahat şəkildə qruplaşdırılan xüsusi üsulları həyata keçirdi. Bu termin məqsədyönlü insan fəaliyyətinin bütün sahələrində qərarların əsaslandırılması üçün riyazi, kəmiyyət üsullarından istifadəni nəzərdə tutur.

Əməliyyat tədqiqatının məqsədi müəyyən bir problemi həll etmək üçün ən yaxşı hərəkət kursunu müəyyən etməkdir. Əsas rol riyazi modelləşdirməyə verilir. Riyazi modeli qurmaq üçün tədqiq olunan sistemin işləmə məqsədini ciddi şəkildə başa düşmək və məqbul dəyərlərin diapazonunu müəyyən edən məhdudiyyətlər haqqında məlumata malik olmaq lazımdır. Məqsəd və məhdudiyyətlər funksiyalar kimi göstərilməlidir.

Tədqiqat işinin məqsədi dinamik proqramlaşdırma üsulunun mahiyyətini mövcud nəzəri və tətbiqi biliklərin vasitəsilə açılmasıdır. Təqdim olunan məqalədə əsas məqsəd müxtəlif xarakterli problemlərin həlli nümunələrini nəzərdən keçirmək, onların hal dəyişənlərinin və idarəetmənin seçilməsi məsələlərinə yeni baxış təqdim etməkdir. Tədqiqat işində qarşıya qoyulan vəzifələr sənaye texnologiyasında, istehsalın təşkilində, iqtisadi planlaşdırmada və digər sahələrdə dinamik proqramlaşdırma metodundan istifadə olunaraq həyata keçirilən əməliyyatların optimal ardıcıllığının qurulmasına xüsusi diqqət yetirmək, nəzəri bilikləri tətbiq edərək konkret nümunələr üzərində dinamik proqramlaşdırma metodunun yerinə yetirilməsidir.

Nəzəri hissə

Dinamik proqramlaşdırma hər bir qərarın nəticələrinin hesablanması və sonrakı qərarlar üçün optimal strategiyanın işlənib hazırlanması əsasında optimal həllərin tapılmasına imkan verən texnika və metodlar toplusunu öyrənən riyazi proqramlaşdırmanın bir sahəsidir. Dinamik proqramlaşdırma məsələləri çoxmərhələlidir, ona görə də "dinamik proqramlaşdırma" termini təkcə məsələlərin xüsusi növünü müəyyən etmir, həm də riyazi proqramlaşdırma məsələlərinin ayrı-ayrı siniflərinin həlli yollarının tapılması üsullarını xarakterizə edir [1-3].

Bərabərsizlik şəklində çoxlu sayda dəyişən və məhdudiyyətləri ehtiva edən bir çox optimallaşdırma məsələlərini həll etmək üçün riyaziyyatın klassik aparatından istifadə etmək mümkün deyil. Və nəticədə, böyük verilənlər problemini yalnız bir neçə dəyişən daxil olmaqla, alt tapşırıqlara bölmək və sonra ümumi problemi hissələrə ayırmaq fikri ortaya çıxdı. Məhz bu ideya dinamik proqramlaşdırma metodunun yaradılması üçün əsas oldu. Optimallaşdırma problemlərinə elm, texnologiya və iqtisadiyyatın demək olar ki, bütün sahələrində rast gəlinir. Onlarla sənaye texnologiyasında, istehsalın təşkilində, iqtisadi planlaşdırmada, fizikanın, biologiyanın və hərbi işlərin müxtəlif məsələlərində məşğul olmaq lazımdır. Buna görə də dinamik proqramlaşdırmanın tətbiq dairəsi genişdir.

Müasir idarəetmə məsələlərində, iqtisadiyyatda və digər sahələrdə riyazi proqramlaşdırmanın əsasını təşkil edən dinamik proqramlaşdırma metodundan geniş istifadə olunur.

Diskret optimallaşdırma məsələləri orada baş verən prosesləri təhlil etmək üçün riyazi metodlardan istifadə olunduğu müxtəlif sahələrdə geniş istifadə olunur. Bu cür problemlərin həllinə ehtiyac, diskret optimallaşdırmanın tətbiqlərdə yaranan problemlərin həllində istifadəsi ilə bağlı mütəxəssislərin təhsilində mühüm elementə çevrilməsinə səbəb olur. Buna görə də diskret proqramlaşdırma məsələlərinin həlli texnologiyası tətbiqi riyaziyyat mütəxəssisləri üçün müasir riyazi təhsilin mühüm komponentlərindən birinə çevrilməlidir.

Optimallıq prinsipi dinamik proqramlaşdırma məsələlərinin mərhələli həlli üçün əsasdır. Dinamik proqramlaşdırmanın iqtisadi problemlərinin tipik nümayəndələri istehsal və saxlama problemləri deyilənlər, kapital qoyuluşlarının bölüşdürülməsi problemləri, istehsalın planlaşdırılmasının planlaşdırılması problemləri və s. Müəssisənin fəaliyyətinin planlaşdırılmasında

zamanla məhsula olan tələbatın dəyişməsi nəzərə alınmaqla dinamik proqramlaşdırma tapşırıqlarından istifadə edilir.

Dinamik proqramlaşdırmanın dəqiq tərifini vermək çətindir. Onun yalnız üç xarakterik xüsusiyyətini qeyd edək. Dinamik proqramlaşdırma aparatında əsas tədqiqat metodu olan optimallıq prinsipinə əlavə olaraq, oxşar problemlər ailəsində müəyyən bir optimallaşdırma probleminin batırılması ideyası mühüm rol oynayır. Onu digər optimallaşdırma üsullarından fərqləndirən üçüncü xüsusiyyəti yekun nəticənin formasıdır. Çoxmərhələli, diskret proseslərdə optimallıq prinsipinin və immersion prinsipinin tətbiqi keyfiyyət meyarının optimal qiymətinə münasibətdə rekursiv-funksional tənliklərə gətirib çıxarır. Alınan tənliklər ilkin problem üçün optimal idarəetmə vasitələrini ardıcıl olaraq yazmağa imkan verir. Burada üstünlük ondan ibarətdir ki, bütün proses üçün nəzarətin hesablanması vəzifəsi prosesin ayrı-ayrı mərhələləri üçün nəzarətin hesablanmasının bir sıra sadə tapşırıqlarına bölünür.

Metodun əsas çatışmazlığı, Bellmanın təbirincə desək, "ölçülülüyün lənətidir" – problemin ölçüsünün artması ilə onun mürəkkəbliyi fəlakətli şəkildə artır.

Dinamik proqramlaşdırma idarə olunan proseslərin optimal planlaşdırılmasına imkan verən riyazi aparatdır. "İdarə olunan" dedikdə müəyyən dərəcədə təsir edə biləcəyimiz prosesləri nəzərdə tuturuq. Bu işin məqsədi stoxastik dinamik proqramlaşdırmanın üsullarını öyrənmək və praktiki məsələlərin həllində tətbiq etməkdir.

Dinamik proqramlaşdırma həlləri alt problemlərlə birləşdirərək problemləri həll edir. O, böl və fəth et metoduna bənzər ola bilər, burada problem üst-üstə düşməyən alt problemlərə bölünür, alt problemlər rekursiv şəkildə həll edilir və sonra orijinal problemin həllini tapmaq üçün birləşdirilir. Bunun əksinə olaraq, dinamik proqramlaşdırma alt tapşırıqlar üst-üstə düşdükdə, yəni alt tapşırıqların ümumi alt tapşırıqları olduqda tətbiq edilir. Bu kontekstdə böl və fəth alqoritmi lazım olduğundan daha çox işləyir, təkrar-təkrar ümumi alt problemləri həll edir. Dinamik proqramlaşdırma alqoritmi hər bir alt problemi yalnız bir dəfə həll edir və sonra onun cavabını cədvəldə saxlayır, beləliklə, hər bir alt problemi həll etdikdə cavabı yenidən hesablamaq işindən qaçır [6].

Optimallıq prinsipinə əsaslanan təxmini alqoritmlərin ümumi ideyası ondan ibarətdir ki, prosesin optimallaşdırılması lazım olan bütün interval o qədər kiçik intervallara bölünür ki, bu kiçik intervalda optimallaşdırma probleminin həlli artıq çətin olmayacaq. Bundan sonra problemin həlli "sondan" başlayır. Optimallıq prinsipinə görə, son mərhələdə optimal idarə-etmə əvvəlki addımlarda nəzarətin necə olmasından asılı deyil. Son mərhələdə nəzarət, birincisi, bu (və yalnız bu) addımın şərtləri üçün optimal olmalı və ikincisi, verilmiş son nöqtəyə aparmalıdır [7].

Hesabi hissə

Bir nümunəyə baxaq. İtalyan riyaziyyatçısı Fibonaççi dovşan populyasiyasının ideallaşdırılmış artımını nəzərə alaraq bir sıra silsilələr kəşf etdi.

Silsilə:

1, 1, 2, 3, 5, 8, 13, 21, ...

Qeyd edə bilərik ki, ilk iki ədəddən sonrakı hər bir ədəd əvvəlki iki ədədin cəmidir. İndi bizə n-ci Fibonaççi ədədini hesablayacaq F(n) funksiyasını formalaşdıraq:

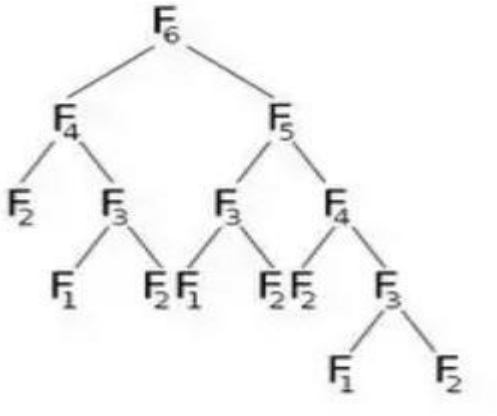
F(n) = nth Fibonacci Number
Əvvəlcədən məlumdur ki,
F(1) = 1
F(2) = 1
F(3) = F(2) + F(1) = 2
F(4) = F(3) + F(2) = 3
Ümumiləşdirmə aparaq:
F(1) = 1
F(2) = 1
F(n) = F(n-1) + F(n-2)

İndi onu rekursiv funksiya kimi yazmaq istəsək, əsas məlumat kimi F(1) və F(2) olur. Beləliklə, Fibonaççi funksiyamız aşağıdakı kim olacaq:

```
Procedure F(n):          //A function to return nth Fibonacci Number
  if n is equal to 1
    Return 1
  else if n is equal to 2
    Return 1
```

end if
Return F(n-1) + F(n-2)

İndi biz F(6) çağırsaq, o, F(5) və F(4) çağıracaq, bunlar da daha bir neçəsini çağıracaq. Onu qrafik olaraq təqdim edək:



Şəkildən göründüyü kimi, F(6) hesablanan zaman F(5) və F(4) çağırılır. Sonra F(5) öz növbəsində F(4) və F(3)-ü çağırır. F(5) hesablandıqdan sonra qətiyyətlə demək olar ki, F(5)–in hesablanması üçün çağırılan bütün funksiyalar ona qədər hesablanmışdı. Bu o deməkdir ki, F(4) –ü biz artıq hesablamışdıq. Amma F(6) hesablanan zaman biz yenə də F(4)-ü hesablamaq məcburiyyətində qalacağıq. Buna həqiqətənmi ehtiyac var? Biz elə edə bilərik ki, F(4) ilk dəfə hesablanan kimi onun qiymətinin yadda saxlayaq. Biz bu qiyməti **dp** adlı massivdə saxlayaraq, lazım olduqda istifadə edəcəyik. Əvvəlcə **dp** massivinin bütün elementlərinə -1 (və ya hesablamamızda əldə edilməyəcək hər hansı digər bir qiymət) qiyməti mənimsədilir. Onda F(6) hesablanan zaman modifikasiya edilmiş F(n) belə formalaşdırılacaq:

Procedure F(n):
if n is equal to 1
Return 1
else if n is equal to 2
Return 1
else if dp[n] is not equal to -1 //That means we have already calculated dp[n]
Return dp[n]
else
dp[n] = F(n-1) + F(n-2)
Return dp[n]
end if

Biz əvvəlki kimi eyni tapşırığı yerinə yetirdik, lakin sadə bir optimallaşdırma ilə, yəni yadda saxlama (memoization) üsulundan istifadə etdik. Əvvəlcə dp massivinin bütün dəyərləri *-1*-ə bərabər olacaqdır. F(4) çağırıldıqda onun boş olub olmadığını yoxlayırıq. F(4)-ün qiyməti -1-ə bərabərdirsə, onun qiymətini hesablayırıq və dp[4] -də yadda saxlayırıq. Əgər -1 -dən başqa hər hansı bir qiymətə bərabərdirsə, bu o deməkdir ki, biz artıq onun qiymətini hesablamışıq. Beləliklə, biz artıq hesablanmış qiymətdən istifadə edəcəyik.

Bu sadə optimallaşdırma metodu yadda saxlama (memoization) üsulundan istifadə edilən dinamik proqramlaşdırma adlanır.

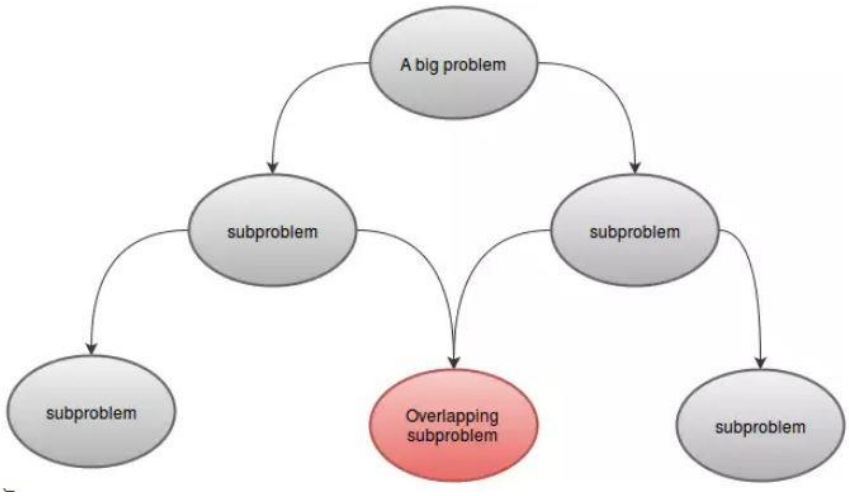
Qarşıya qoyulan məsələ bəzi xüsusiyyətlərə malik olduqda dinamik proqramlaşdırma ilə həll edilə bilər. Bunlara aiddir:

- **Alt məsələ:**

Dinamik proqramlaşdırma məsələsi bir və ya bir neçə alt məsələyə bölünə bilər. Məsələn: F(4) daha kiçik alt tapşırıqlara F(3) və F(2) –yə bölünə bilər. Alt problemlər bizim əsas problemimizə bənzədiyi üçün eyni texnika ilə həll edilə bilər.

- **Üst-üstə düşən alt tapşırıqlar:**

DP məsələsində üst-üstə düşən alt tapşırıqlar olmalıdır. Bu o deməkdir ki, eyni funksiyanın bir neçə dəfə hesablandığı bəzi ümumi hissə olmalıdır. Məsələn: F(5) və F(6) məsələsinin hər ikisi F(4) və F(3) kimi alt məsələlərə malikdir. Məhz bu səbəbdən hesablanan qiymətləri massivimizdə saxlamışıq.



- **Optimal struktur:**

Tutaq ki, sizdən g(x) funksiyasını minimuma endirmək tələb olunur. Bilirsiniz ki, g(x)-in qiyməti g(y) və g(z)-dən asılıdır. İndi g(y) və g(z) hər ikisini minimuma endirməklə g(x)-ni minimuma endirə bilsək, yalnız bu halda problemin optimal alt quruluşa malik olduğunu deyə bilərik. Əgər g(x) g(y)-i minimuma endirməklə minimuma endirilirsə və g(z)-ni minimuma endirmək və ya maksimuma çatdırmaq g(z)-ə təsir etmirsə, onda bu məsələnin optimal alt strukturu yoxdur. Sadə dillə desək, əgər problemin optimal həlli onun alt probleminin optimal həllindən tapıla bilərsə, o zaman problemin optimal alt quruluş xassəsinə malik olduğunu deyə bilərik.

Nəticələr cədvəldə və qrafikdə təqdim olunur [6-8].

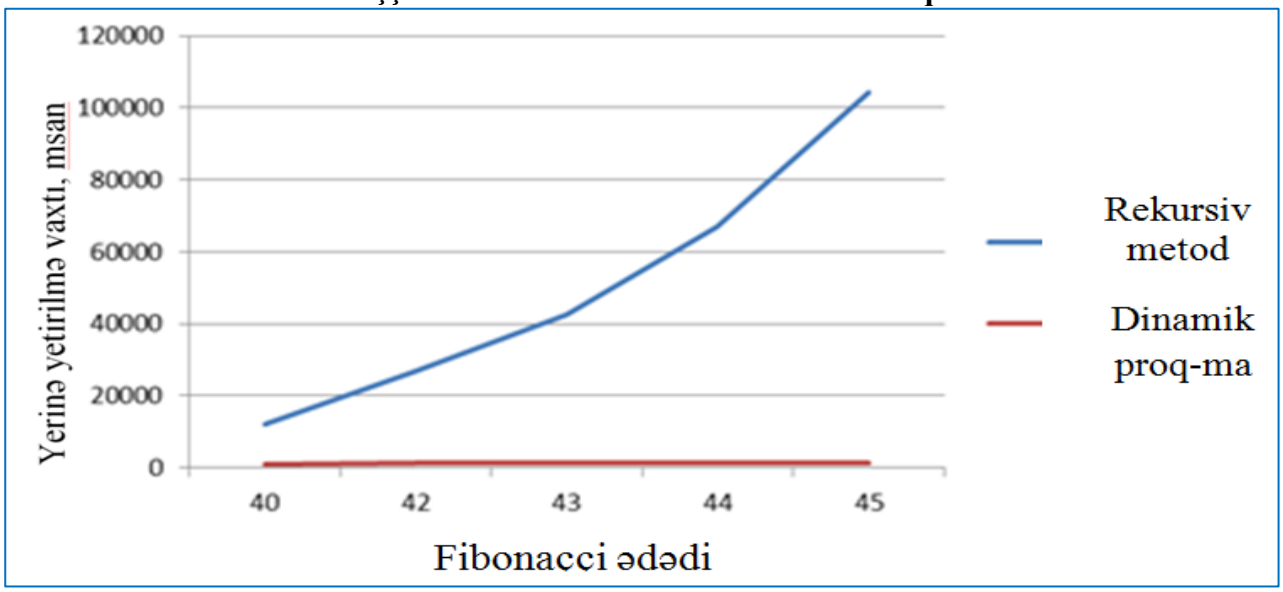
Cədvəl.

Fibonaççi ədədinin hesablanması nəticələrinin cədvəli.

Fibonaççi ədədi		40	42	43	44	45
Yerinə yetirilmə vaxtı, ms	Rekursiv metod	11929	26963	41552	66825	104383
	Dinamik proqramlaşdırma	1125	1174	1282	1350	1406

Qrafik 1.

Fibonaççi ədədinin hesablanması nəticələrinin qrafiki.



Yuxarıda təqdim olunan nəticələrdən aydın görünür ki, dinamik proqramlaşdırmadan istifadə edərək hesablama vaxtı rekursiya ilə hesablama vaxtı ilə müqayisədə xeyli aşağıdır. Fərq xüsusilə 40-cı və daha böyük Fibonaççi ədədi hesablanarkən nəzərə çarpır [9-10].

Dinamik proqramlaşdırmada vəziyyəti başa düşmək üçün bir nümunəyə baxaq. N elementdən r elementi neçə üsulla seçə bilərik? Bu ${}^{n}C_{r}(n,r)$ kimi göstərilir. İndi bir element götürək.

- Əgər siz element seçməsəniz, ondan sonar ${}^{n-1}C_{r}$ ilə verilən qalan *n-1* elementlərindən *r* element götürməlisiniz.
- Əgər siz element seçmisinizsə, ondan sonra ${}^{n-1}C_{r}$ ilə verilən qalan *n-1* elementlərindən *r-1* element götürməlisiniz.

Bu ikisinin cəmi bizə üsulların ümumi sayını verir, yəni:

$$ {}^{n}C_{r} = {}^{n-1}C_{r} + {}^{n-1}C_{r-1} $$

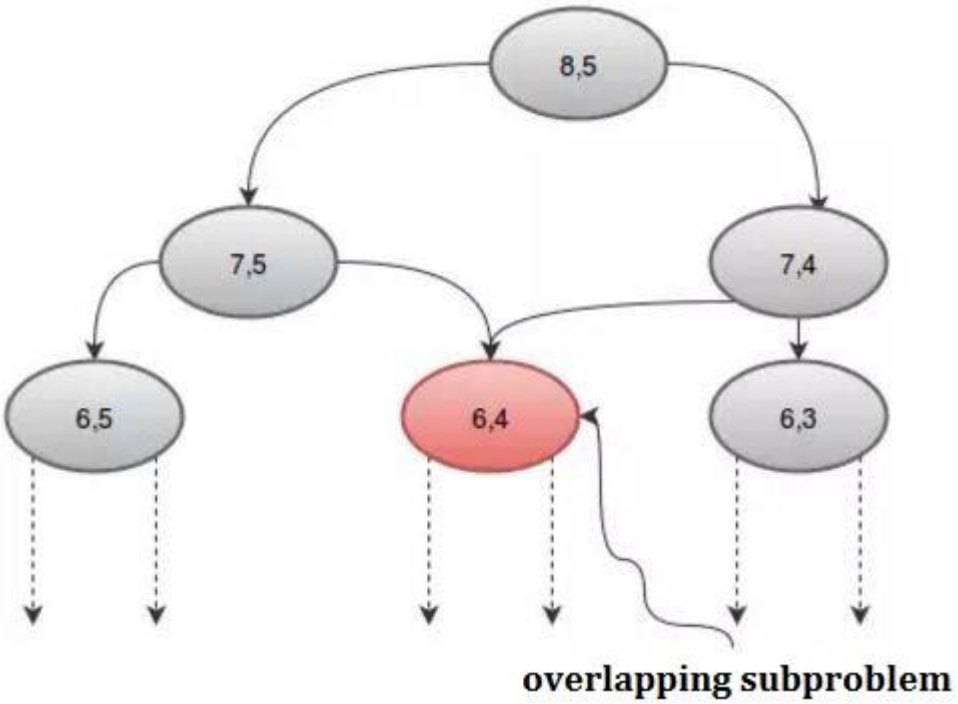
Əgər ${}^{n}C_{r}(n,r)$ –i n və r-i parametr kimi qəbul edən funksiya saysaq və ${}^{n}C_{r}$ - i funksiyanın qiyməti kimi qəbul etsək, onda yuxarıda dediyimiz münasibəti bu cür yaza bilərik:

$$ {}^{n}C_{r}(n,r) = {}^{n}C_{r}(n-1,r) + {}^{n}C_{r}(n-1,r-1) $$

Bu, rekursiv münasibətdir. Onu başa çatdırmaq üçün baza registrlərini təyin etməliyik. Bilirik ki, ${}^{n}C_{r}(n,r)=n$,həmçinin ${}^{n}C_{n}(n,r)=1$. Bu iki bərabərliyi əsas başlanğıc kimi qəbul edərək, ${}^{n}C_{r}$ -i hesablayan alqoritmi bu şəkildə qura bilərik:

```
Procedure nCr(n, r):
if r is equal to 1
 Return n
else if n is equal to r
  Return 1
end if
Return nCr(n-1,r) + nCr(n-1,r-1)
```

Proseduru qrafik şəkildə nəzərdən keçirsək, bəzi rekursiyaların bir dəfədən çox çağırıldığını görə bilərik. Məsələn: n = 8 və r = 5 götürsək, alırıq:



Biz **dp** massivindən istifadə edərək təkrar çağırışdan qaça bilərik. 2 parametr olduğu üçün bizə 2D massivi lazım olacaq. Biz dp massivinin bütün elementlərinə *-1* mənimsədirik, burada *-1* qiymətin hələ hesablanmadığını bildirir. Prosedurumuz belə olacaq:

```
Procedure nCr(n,r):
if r is equal to 1
   Return n
else if n is equal to r
   Return 1
else if dp[n][r] is not equal to -1      //The value has been calculated
   Return dp[n][r]
end if
dp[n][r] := nCr(n-1,r) + nCr(n-1,r-1)
Return dp[n][r]
```

$^{n}C_{r}$ -i təyin etmək üçün bizə 2 parametr lazım oldu: ***n*** və ***r***. Bu parametrlər vəziyyətlər adlanırlar. Burada sadəcə olaraq nəzərə almaq lazımdır ki, vəziyyətlərin sayı *dp* massivinin ölçüsünü təyin edir. Bu halda dinamik proqramlaşdırma üçün hazırlanan alqoritmlər aşağıdakı ümumi sxemə uyğun olacaq:

```
Procedure DP-Function(state_1, state_2, ...., state_n)
Return if reached any base case
Check array and Return if the value is already calculated.
Calculate the value recursively for this state
Save the value in the table and Return
```

Vəziyyətlərin təyin edilməsi dinamik proqramlaşdırmanın əsas tərkib hissəsidir. O, məsələnin həllini təyin edən parametrlərin sayından asılıdır, onların hesablanmasının optimallaşdıraraq, bütün məsələnin optimallaşdırılmasına nail olmaq olar.

Dinamik proqramlaşdırmanın həllinin qurulması üçün növbəti addımları atmaq lazımdır. Dinamik proqramlaşdırma ilə həll edilən məsələlər həddən artıq çoxdur. Bunu nəzərə alaraq, dinamik proqramlaşdırma ilə həll edilən məsələlərin həllinin qurulması prosesinə baxaq. Bu mövzuya aid digər nümunələr dinamik proqramlaşdırmanın necə işlədiyini anlamağa kömək edir. Bu nümunədə biz əvvəldən dinamik proqramlaşdırma vasitəsilə həlli necə tapacağımıza nəzər yetirək.

Dinamik proqramlaşdırma həllinin qurulması üçün iki üsul var. Onlar aşağıdakılardır:

• İterativ (dövrlərdən istifadə etməklə)

• Rekursiv (rekursiyadan istifadə etməklə).

Məsələn, iki s_1 və s_2 sətirlərinin ən uzun ümumi ardıcıllığının uzunluğunu hesablamaq üçün alqoritm belə görünür:

```
Procedure LCSlength(s1, s2):
Table[0][0] = 0
for i from 1 to s1.length
   Table[0][i] = 0
endfor
for i from 1 to s2.length
   Table[i][0] = 0
endfor
for i from 1 to s2.length
   for j from 1 to s1.length
      if s2[i] equals to s1[j]
         Table[i][j] = Table[i-1][j-1] + 1
      else
         Table[i][j] = max(Table[i-1][j], Table[i][j-1])
      endif
   endfor
endfor
Return Table[s2.length][s1.length]
```

Bu iterativ həlldir. Onun bu şəkildə kodlaşdırılmasının bir neçə səbəbi var:

- İterasiya rekursiyadan daha sürətlidir.
- Zaman və məkanın mürəkkəbliyini müəyyən etmək daha asandır.
- Mənbə kodu qısa və təmizdir.

İlkin proqram koduna baxaraq, onun necə və niyə işlədiyini anlamaq olar, lakin bu həlli necə həll edəcəyinizi görmək çətindir. Bununla belə, yuxarıda qeyd olunan iki yanaşma iki fərqli psevdokoda çevrilir. Biri iterasiyadan (aşağıdan yuxarı - Bottom-Up), digəri isə rekursiyadan (yuxarıdan aşağıya) istifadə edir. Sonuncu həm də yadda saxlama (memoization) üsulu kimi tanınır. Bununla belə, iki həll daha çox və ya daha az ekvivalentdir və digərinə çevrilə bilər. Bu nümunədə bir problemin rekursiv həllinin necə tapılacağına baxaq.

Əməliyyat tədqiqatı modellərində məhdudiyyətlərin və məqsəd funksiyasının asılı olduğu dəyişənlər diskret (əksər hallarda tam ədəd) və davamlı (davamlı) ola bilər. Öz növbəsində məhdudiyyətlər və məqsəd funksiyası xətti və qeyri-xətti bölünür. Bu modellərin həlli üçün müxtəlif üsullar mövcuddur, onlardan ən məşhuru və effektivi məqsəd funksiyası və bütün məhdudiyyətlər xətti olduqda xətti proqramlaşdırma metodlarıdır. Digər növ riyazi modellərin həlli üçün dinamik proqramlaşdırma üsulları, tam proqramlaşdırma, qeyri-xətti proqramlaşdırma, çoxməqsədli optimallaşdırma və şəbəkə modellərinin üsulları nəzərdə tutulmuşdur.

Əməliyyat tədqiqatının demək olar ki, bütün üsulları iterativ xarakter daşıyan hesablama alqoritmləri yaradır. Bu o deməkdir ki, hər bir addımda (iterasiya) biz tədricən optimal həllə yaxınlaşan həll əldə etdikdə problem ardıcıllıqla (iterativ) həll olunur.

Alqoritmlərin iterativ təbiəti adətən eyni tipli böyük hesablamalara gətirib çıxarır. Bu alqoritmlərin əsasən kompüter tətbiqi üçün hazırlanmasının səbəbi budur.

Əməliyyatların tədqiqi üsullarının əksəriyyəti ilk növbədə dəqiq müəyyən edilmiş məzmunlu tapşırıqlarla əlaqələndirilir. Klassik riyaziyyat aparatının çoxlu sayda dəyişənləri və/yaxud bərabərsizliklər şəklində məhdudiyyətləri ehtiva edən bir çox optimallaşdırma məsələlərini həll etmək üçün az faydası olduğu ortaya çıxdı. Şübhəsiz ki, yüksək ölçülü bir problemi yalnız bir neçə dəyişən daxil olmaqla daha kiçik ölçülü alt problemlərə bölmək və

sonra ümumi problemi hissələrlə həll etmək ideyasının cəlbediciliyi. Dinamik proqramlaşdırma metodu məhz bu fikrə əsaslanır.

Dinamik proqramlaşdırma riyazi metoddur, onun yaradılması və inkişafi ilk növbədə Bellmana məxsusdur. Metod resursların bölüşdürülməsi, dəyişdirilməsi və inventarların idarə edilməsi, yükləmə problemləri də daxil olmaqla çox geniş problemləri həll etmək üçün istifadə edilə bilər. Dinamik proqramlaşdırmanın xarakterik xüsusiyyəti, hər biri bir idarə olunan dəyişənlə əlaqəli olan bir problemin mərhələlərlə həllinə yanaşmadır. Müxtəlif mərhələləri birləşdirən təkrarlanan hesablama prosedurları toplusu son mərhələyə çatdıqda bütövlükdə problemin mümkün optimal həllini təmin edir.

Dinamik proqramlaşdırma adının mənşəyi çox güman ki, sabit vaxt intervallarında qərar qəbulu problemlərində (məsələn, inventarların idarə edilməsi məsələlərində) dinamik proqramlaşdırma metodlarından istifadə ilə bağlıdır. Bununla belə, zaman amilinin nəzərə alınmadığı problemləri həll etmək üçün dinamik proqramlaşdırma metodlarından da uğurla istifadə olunur. Bu səbəbdən problemin həlli prosesinin mərhələli xarakterini əks etdirən çoxmərhələli proqramlaşdırma termini daha uyğun görünür.

Beləliklə, dinamik proqramlaşdırma üsulu ilə idarəetmənin optimallaşdırılması prosesində iki dəfə çoxmərhələli proses keçir:

- ilk dəfə - sonundan əvvələ qədər, bunun nəticəsində hər bir addımda şərti optimal nəzarət və verilmişdən başlayaraq prosesin sonuna qədər bütün addımlarda optimal qazanc (həmçinin şərti) tapılır. ;

ikinci dəfə - əvvəldən axıra qədər, bunun nəticəsində prosesin bütün mərhələlərində optimal idarəetmələr tapılır.

Demək olar ki, dinamik proqramlaşdırma üsulu ilə optimal idarəetmənin qurulması proseduru iki mərhələyə bölünür: ilkin və yekun. İlkin mərhələdə, hər bir addım üçün sistemin vəziyyətindən (əvvəlki addımlar nəticəsində əldə edilmiş) şərti optimal idarəetmə və bundan başlayaraq bütün qalan addımlarda şərti optimal qazanc müəyyən edilir. Son mərhələdə hər bir addım üçün (şərtsiz) optimal nəzarət müəyyən edilir. İlkin (şərti) optimallaşdırma tərs qaydada addım-addım həyata keçirilir: sonuncu addımdan birinciyə; yekun (şərtsiz) optimallaşdırma - həm də addımlarla, lakin təbii qaydada: ilk addımdan sonuncuya qədər. Optimallaşdırmanın iki mərhələsindən birincisi müqayisəolunmaz dərəcədə daha vacibdir və vaxt aparır. Birinci mərhələ başa çatdıqdan sonra ikinci mərhələnin həyata keçirilməsi heç bir çətinlik yaratmır: birinci mərhələdə artıq hazırlanmış tövsiyələri "oxumaq" qalır [5-10].

Həmçinin deyə bilərik ki, dinamik proqramlaşdırma ideyası optimal alt quruluşdan istifadə edərək qrafikdə bir təpədən digərinə ən qısa yolu tapmaqdır; düz xətt sadə kənarı bildirir; dalğalı xətt birləşdirdiyi təpələr arasında ən qısa yolu göstərir (yolun ara ucları göstərilmir); qalın xətt son ən qısa yolu göstərir.

Nəticə

Baxılan məsələlərin həllində istifadə olunan dinamik proqramlaşdırmanın üstünlüklərini aşağıdakı kimi ümumiləşdirmək olar:

1. Dinamik proqramlaşdırmanın ideyası və metodu iqtisadiyyatdan gələn problemlər olan diskret problemlərə ən uyğunlaşdırılmışıdır.

2. Dinamik proqramlaşdırma metodu istənilən təyinat üsulu və hər hansı icazə verilən vəziyyətlər və idarəetmələr dəsti üçün tətbiq edilir. Klassik optimallaşdırma üsulları və riyazi proqramlaşdırmanın digər hesablama üsulları bu üstünlükdən məhrumdur.

3. Diskret vəziyyətdə dinamik proqramlaşdırma metodunun hesablama sxemləri səmərəlilik göstəricisinin optimal qiymətlərinin sadalanması və vəziyyət dəyişəninin bütün mümkün qiymətləri üçün *i*-ci addımda nəzarət ilə əlaqələndirilir, lakin bu metoddan istifadə edərək hesablamaların məbləği variantların

birbaşa sadalanması ilə müqayisədə xeyli azdır. Bu, onunla əlaqədardır ki, şərti optimallaşdırma mərhələsində uğursuz variantlar dərhal ləğv edilir və bu mərhələdə yalnız şərti optimal olanlar saxlanılır.

4. Dinamik proqramlaşdırma metodu ilkin verilənlər S_i və n dəyişikliklərinə həssaslığı təhlil etməyə imkan verir. Əslində burada bir məsələ həll olunmur, fərqli S_i vəziyyətləri üçün eyni tipli və hər addımda fərqli olan məsələlər toplusu həll edilir. Buna görə də, ilkin məlumatları dəyişdirərkən, problemi yenidən həll etmək deyil, artıq yerinə yetirilən hesablamalara yalnız sadə əlavələr etmək mümkündür, yəni addımların sayını n və ya S_i dəyərlərinin sayını artırmaqla artıq həll edilmiş problem davam etdirilir.

Diskret optimallaşdırma məsələləri orada baş verən prosesləri təhlil etmək üçün riyazi metodlardan istifadə olunduğu müxtəlif sahələrdə geniş istifadə olunur. Bu cür problemlərin həllinə ehtiyac, diskret optimallaşdırmanın tətbiqlərdə yaranan problemlərin həllində istifadəsi ilə bağlı mütəxəssislərin təhsilində mühüm elementə çevrilməsinə səbəb olur. Buna görə də diskret proqramlaşdırma məsələlərinin həlli texnologiyası tətbiqi riyaziyyat mütəxəssisləri üçün müasir riyazi təhsilin mühüm komponentlərindən birinə çevrilməlidir.

ƏDƏBİYYAT SİYAHISI:

1. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Глава 15. Динамическое программирование // Алгоритмы: построение и анализ = Introduction to Algorithms / Под ред. И. В. Красикова. - 2-е изд. - М.: Вильямс, 2005. - 1296 с. - ISBN 5-8459-0857-4

2. Окулов С. М., Пестов О. А.. «Динамическое программирование», БИНОМ, 2015 год, 299 стр.2-е изд.

3. Струченков В. И.. «Динамическое программирование в прикладных задачах специального вида». Прикладная информатика, №3 (87), 2020, стр. 54-68.

4. Di Zhu, Yadong Mei, Xinfa Xu, Zhangjun Liu, Zhenhui Wu, Hao Cai, Optimal Operation of a Parallel Multireservoir System for Flood Control using a Stagewise Compensation Method, Water Resources Management, 10.1007/ s11269-021-02803-9, 35, 6, (1689-1710), (2021).

5. Feng, Zk., Niu, Wj., Jiang, Zq. et al. Monthly Operation Optimization of Cascade Hydropower Reservoirs with Dynamic Programming and Latin Hypercube Sampling for Dimensionality Reduction. Water Resour Manage 34, 2029–2041 (2020).

6. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Глава 15. Динамическое программирование // Алгоритмы: построение и анализ = Introduction to Algorithms / Под ред. И. В. Красикова. - 2-е изд. - М.: Вильямс, 2005. - 1296 с. - ISBN 5-8459-0857-4

7. Sanjoy Dasgupta , Christos H. Papadimitriou, Umesh Vazirani Algorithms = Algorithms. — 1-е изд. — McGraw-Hill Science/Engineering/Math, 2006. — С. 336. — ISBN 0073523402

8. Щербина О. А. Методологические аспекты динамического програм-мирования // Динамические системы, 2007, вып. 22. — с.21-36.

9. Габасов Р., Кириллова Ф.М.Основы динамического программ-мирования.- Мн.: Изд-во БГУ, 1975. - 262 с.

10. Романовская А.М. Динамическое программирование: Учебное пособие. Романовская А.М., Мендзив М.В.– Омск: Издатель Омский институт РГТЭУ, 2010. – 58 с.

Гошгар Сейфулла оглы АЛИЕВ
Западно-Каспийский Университет, Баку, кандидат технических наук

Ляман Ильхам кызы БАШИРЗАДЕ
Западно-Каспийский Университет, Баку, магистрант

ВОПРОСЫ ПРИМЕНЕНИЯ МЕТОДА ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ В УПРАВЛЕНИИ

Резюме. Проблемы оптимизации встречаются практически во всех областях науки, технологии и экономики. Методы оптимизации, составляющие основу математического программирования, широко используются в современной теории управления, что делает представленную тему актуальной. В статье представлены результаты исследования данной проблемы и применения методов динамического программирования в области управления предприятием.
Ключевые слова: динамическое программирование, принятие решений, оптимальное управление.

Goshgar Seyfullah ALIYEV
Western Caspian University, Baku, PhD in technology

Laman Ilham BASHIRZADE
Western Caspian University, master's degree

APPLICATION ISSUES OF DYNAMIC PROGRAMMING METHOD IN MANAGEMENT

Summary: Optimization problems are found in almost all areas of science, technology and economics. Being the basis of mathematical programming, optimization methods are widely used in modern management theory, which makes the presented topic relevant. The article presents the results of the researched problem and application of dynamic programming methods in the field of enterprise management.
Key words: dynamic programming, decision making, optimal management.

Daxil olub: 02.03.2022