MADENCİLİK Aralık December 1991 Cilt Volume XXX Sayı No 4

Dinamik Programlama Tekniğindeki Gelişmeler

Developments in Dynamic Programming Technique

Ercüment YALÇIN (*)

ÖZET

Bu yazıda, optimum nihai açık işletme sınırını bulmak için kullanılan dinamik programlama tekniğiyle yazılmış algoritmalar incelenmiş, Koenigsberg ve Wright tarafından geliştirilen algoritmalar tanıtılmıştır.

ABSTRACT

In this paper, dynamic programming algorithms used to find the optimum ultimate pit limits were discussed and the algorithms developed by Koenigsberg and Wright were explained.

(*) Dr. Maden Y. Müh., Arş. Gör., ODTÜ Maden Müh. Böl., ANKARA

1. GİRİŞ

Açık işletmelerde bilgisayar yardımıyla optimum nihai açık işletme sınırının bulunması, günümüz madenciliğinde üzerinde çalışılan konuların başında gelmektedir. Uzun vadeli açık işletme planlamasında, cevher yatağından en yüksek kârı elde edebilmek için optimum nihai sınırın bilinmesi zorunlu hale gelmiştir. Bu nedenle, son 25 yıl içerisinde bu konuda yapılmış birçok çalışma bulunmaktadır. Bu çalışmalar sırasında geliştirilen tekniklerden Graf tekniği, Lineer programlama tekniği ve Network akım tekniği günümüzde çok az kullanılan tekniklerdir (Kim, 1978). Hareketli koni yöntemi ise yaygın olarak kullanılmasına karşın, çok fazla bilgisayar hafızası ve bilgisayar zamanı gerektirmekte ve gerçek optimumu bulması garanti değildir. Dinamik programlama tekniğiyle yazılmış bilgisayar yazılımlarının yüksek hıza ve hafızaya sahip bilgisayar sistemlerinde kullanılması fazla sorun yaratmamaktadır. Düşük kapasiteli kişisel bilgisayarlarda ise hız ve hafıza sınırlı olduğu için bazı sorunlarla karşılaşılabilmektedjr (Shenggui ve Starfield, 1985; Writght, 1987; Lizotte, 1988; Ünal ve Yalçın, 1989).

Günümüzde yaygın olarak kullanılan ve daha verimli hale getirmek için üzerinde en fazla çalışmanın yapıldığı diğer bir teknik ise dinamik programlama tekniğidir. Bu teknik, kolay anlaşılması, gerekli olan bilgisayar hafıza ve zamanının azlığı ileelde edilen nihai sınırın optimum sınıra çok yakın olması nedeniyle diğerlerine göre daha avantajlı duruma gelmektedir..

Bu yazıda, son yıllarda dinamik programlama tekniği üzerinde yapılmış olan çalışmalar ve bu çalışmalar sonucunda geliştirilmiş yeni algoritmalar incelenecektir.

2. BLOK MODELİ

Optimum nihai sınırı bulmak amacıyla geliştirilen bütün tekniklerde blok modeli kullanılmaktadır. Kullanılan blok modelleri beş gruba ayrılabilir (Kim, 1978).

2.1. Üç Boyutlu Sabit Blok Modeli

Bu modelde cevher yatağı bloklara bölünmekte ve yatağın her yerinde blok boyutlan aynı olmaktpdır. Blok yüksekliği, genellikle basamak yüksekliğine eşit olmaktadır.

2.2. Üç Boyutlu Değişken Blok Modeli

Blok yüksekliği genellikle basamak yüksekliğine eşit olmakta ve yatay yönlerdeki boyutlardan en az birisi bloktan bloka değişmektedir.

2.3. Izgara Düğüm Modeli

Bu blok modelinde, cevher yatağı yüzeyinde yatay yönlerde 2 boyutlu ızgara düğüm noktaları oluşturulmaktadır. Her bir ızgara düğüm noktasının dikey boyutu, örtü tabakası ve cevher kalınlığına bağlı olarak değişmektedir.

2.4. İki Boyutlu Düzensiz Blok Modeli

Cevher yatağından alınan düşey kesitler üzerinde poligonlor ile düzensiz şekiller belirlenmektedir. Düşey kesitler arasındaki uzaklık, cevher yatağının sürekliliğine bağlı olarak sabit ya da değişken olabilmektedir.

2.5. Üç Boyutlu Düzensiz Blok Modeli

Bu blok modelinde ise, cevher yatağından alınan yatay kesitler üzerindeki düzensiz ve saçınmış haldeki cevher zonları poligonlar vasıtası ile belirtilmektedir. Basamak yüksekliği sabit olmaktadır.

Yukarıda verilen blok modellerinden yaygın olarak kullanılanı 3 boyutlu sabit blok modelidir. Bu blok modeli, masif bakır yatakları için geliştirilmiştir ve sediment cevher yataklarında da kullanılabilmektedir (Kim, 1987).

3. DİNAMİK PROGRAMLAMA ALGORİTMALARI

3.1. İki Boyutlu Dinamik Programlama Algoritması

2 boyutlu dinamik programlama algoritması ilk olarak, Lerchs ve Grossman (1965) tarafından cevher yatağından alınan düşey kesitler üzerinde optimum nihai sınırı bulmak için geliştirilmiştir. Bu tekniğin algoritması aşağıda verilmiştir.

m_{ij}= Blok net değeri,
i = Sıra no, i= 1, 2, ..., I
j = Sütun no, j= 1, 2, ..., J

1. Aşama:

a) $M_{ij} = \sum_{q=1}^{i} m_{qj}$ bütün i, j için

My =j sütununda, i katındaki blok da dahil olmak üzere toplam net değer.

b) Yapay (i = 0) sıracı ilave edilir.
M_{oj}=0, j=1,2,...J

c) j=0, P_{oj}=0

2. Aşama.

Eğer j=J ise 5. aşamaya gidilir.

3. Aşama:

H+1

4. Aşama:

$$P_{ij} = M_{ij} + \max_{r} (P_{i+r, j-1}), \quad r = -1,0,1 \quad i = 1,2,...,I$$

Her i b'oğu için, (i,j) bloğundan en yüksek değere sahip (i+r, j-1) bloğuna giden ok işaretlenir.

Eğer i = I ise 2. aşamaya gidilir.

5. Aşama:

P_0j = Toplam net değer

Eğer P_0j ^ 0 ise cevher yatağı ekonomik değildir.

Eğer POJ>0 ise okları takip ederek optimum nihai sınır bulunur.

Yukarıdaki algoritmanın bütün kesitlerde uygulanmasıyla bulunan optimum nihai sınırlar yan yana getirildiğinde aralarında uyumsuzluklar olmakta ve elle düzeltmeler yapmak gerekmektedir. Bu işlem sonucunda ise bulunan sınır optimumdan uzaklaşmaktadır.

3.2. Uç Boyutlu Dinamik Programlama Algoritması

Johnson ve Sharp (1971), 2 boyutlu dinamik programlama algoritmasını kullanarak 3 boyutlu dinamik programlama algoritmasını geliştirmişlerdir. Tekniğin algoritması ile ilgili geniş bilgi Ünal ve Yalçın (1989) ve Yalçın (1991) kaynaklarında verilmiştir. 3 boyutlu dinamik programlama algoritması genel olarak iki bölümden meydana gelmiştir. Birinci bölümde, kesitlerin her katı için bir optimum sınır ve net değer bulunmakta ve bu, uzunlamasına kesiti temsil eden 2 boyutlu matrikse kaydedilmektedir. İkinci bölümde ise uzunlamasına kesit üzerinde 2 boyutlu dinamik programlama algoritması ile optimum sınır bulunmaktadır.

2 boyutlu dinamik programlama algoritmasında olduğu gibi bu algoritmada da, yan yana olan kesitler için bulunmuş nihai sınırlar arasında uyumsuzluklar olabilmekte ve elle düzeltmeler yapmak gerekmektedir. Dar ve uzun cevher yataklarında ise başarıyla uygulandığı görülmüştür (Shenggui ve Starfield, 1985).

3.3. Koenigsberg'in Üç Boyutlu Dinamik Programlama Algoritması

3 boyutlu dinamik programlama tekniğiyle bulunmuş optimum nihai sınır üzerinde elle düzeltme yapmayı ortadan kaldırmak için, Koenigsberg (1982) tarafından yeni bir algoritma geliştirilmiştir. 2 boyutlu dinamik programlama algoritmasında bir bloktan diğerine geçilirken, bir önceki sütunda bulunan 3 adet komşu blok gözönüne alınırken, Koenigsberg tarafından geliştirilen algoritmada 4 adet komşu sütunda yer alan komşu bloklar gözönüne alınmaktadır,Şekil 1.

Blok b i j k ye komşu olan 4 sütunu göstermek için S (Side= Yan, kenar) ve B (Back= Arka) notasyonları kullanılmıştır.

(j-1 ,k) sütununda Si=i'nin yanı;

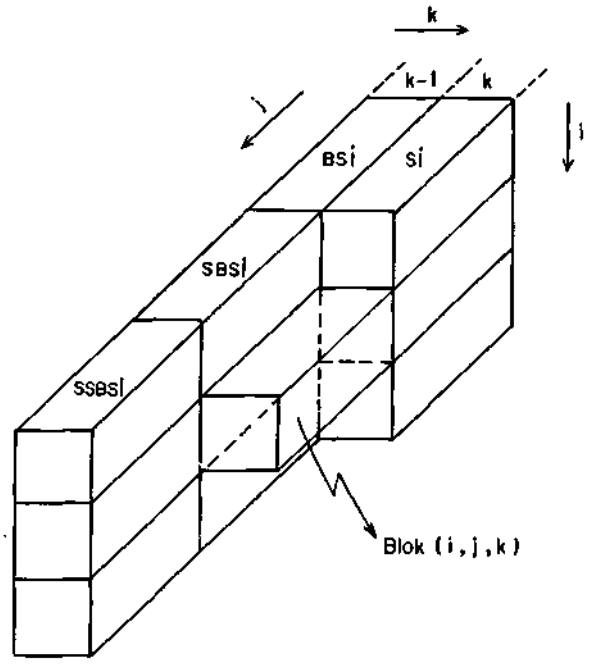
(j-1, k-1) sütununda BSi=i'ı<in yanının arkası;

(j, k-1)sütununr SBSi=ı'.in yanı in arkasının yanı;

(j+1, k-1) sütununda SSBSi=i'nin yanının arkasının yanının yanı.

Eğimin 1:1 olduğu düşünüldüğünde, bu eğimin sağlanabilmesi için aşağıdaki şartların sağlanması gerekmektedir:

li-Sil <1	ISi-BSil < 1	
li-BSil <1	İSi-SBSil <1	O)
li-SBSil < 1	IBSi-SBSil < 1	
li-SSBSil < 1	ISBSi-SSBSil <1	



Şekil 1. Blok (i, j, k)a komşu 12 blokun konumları

Pj j k'nın optimum değeri için, (j-1, k) sütunundaki 3 komşu kattan en az birisi Si'dir, aynı durum (j-1, k-1) sütunu içinde geçerlidir. Blok bj j_k için aşağıdaki tanımlamalar yapılmıştır;

(i, j, k)= blok tanımlayıcı,

Pj j_k = blokun optimum değeri,

Si = G"[1]. k) sütunundaki komşu blok,

BSi = Ü"[1]. k-1) sütunundaki komşu blok,

Bi = SBÎ3İ = Ü' ^"1) sütunundaki komşu blok,

SSBSi= (j+1. k-1) sütunundaki komşu blok.

Bir blok için aynı Pj j_k optimum değerini veren değişik blok setleri olabilir, (j-1 ,k) sütunundaki Si bbklan için P_{Si} j_k değerini ve bu değeri veren blok setinin bilindiği kabul edildiğinde, eğer Si bloklarından birisi Pj $;_k$ optimum değeri için komşu olmak zorunda ise, bu durumunda S(Si) ve BS(Si) blokları ile BSi bloğu arasında şu şartlar sağlanmalıdır:

$$\begin{matrix} IBSi - \widetilde{S(Si)} \quad I \le 1 \\ IBSi - \widetilde{BS(Si)} \quad I \le 1 \end{matrix} \qquad (2)$$

Eğer BSi * $\widetilde{SBS(Si)}$ ise, bu durumda (j-1, k-1) sütununda yukarıdaki şartları sağlayan alternatif blok aranır. Alternatif blokun seçilmesi durumunda, Pj j_k değerinin hesaplanması için Psi, j-1, k değerinden PsBSSi,j-ı,k-ı değeri çıkarılmalı ve Pj j . $^{k-1}$ değeri ilave edilmelidir. Burada i', alternatif olarak seçilmiş olan katı belirtmektedir. Eğer bir blok Pj j_k optimum değerini sağlayan bir önceki blok değilse, bu blokdan önce gelen diğer bloklar sete dahil edilmez.

$$P_{i,j,k} = \sum_{P_{i,j,k}} M_{I,j,k} \qquad (3)$$

Aynı yöntem Pj $_k$ değerini sağlayan SBSi ve SSBSi bloklarının bulunmasında da kullanılır.

Blok bj j_k üzerindeki Pj j_k optimum değeri için aşağıdaki eşitlik yazılabilir:

$$P_{I,j,k}=M_{I,j,k} + \max \begin{cases} P_{Si,\ j-1,\ k} - \widetilde{P_{SBS(Si),\ j-1,\ k-1}} + P_{BSİ,\ j-1,\ k-1} \\ -P_{S(BSi),\ j,k-1} + \widetilde{P_{SBSi,j,k-1}} \\ -P_{S(SBSi),j+1,\ k-1} + \widetilde{P_{SSBSi,\ j+1,\ k-1}} \end{cases} \qquad (4)$$

Burada;

Pj j_k = blok bj j_k için optimum ocak net değeri,

Mj j_k = blok bj j_k için toplam sütun değeri,

^{p}Si,j-ı,k = (M >^{k}) sütunundaki komşu bloklardan birisi için ocak net değeri,

^{p}SBS(Si),j-ı,k-ı = ^{p}Si,j-ı,k değerininhesaplanması sırasında (j-1. k-1) sütununda yer alan optimum komşu blok için ocak net değeri,

P_{BSi}j_!$_{k-1}$ = (j-1, k-1) sütununda yer alan veblokbjj$_{k}$ ve blok b_{BSi} i.$_{1|k}$ ile uyum içinde olan optimum komşu blok bsB^si)$_{H\ k}$.! için ocak net değeridir.

Eşitlik 4'deki terimler de, aynı yolla bulunmuş olan optimum komşu bloklar için ocak net değerleridir. Eşitlikdeki negatif terimler, şev açısını ve bloklar arasındaki uyumu sağlamak için gereken düzeltmelerdir. Buradaki düzeltmeler (k-1) kesitindeki bloklar için bulunmuş olan ocak net değerlerini ifade etmektedir, (k-1) kesitindeki düzeltmeler, (k-2) kesitindeki düzeltmeleri, bu da (k-3) kesitindeki düzeltmeleri gerektirebilmekte ve bu işlem (k=1) oluncaya kadar devam edebilmektedir. Bu durum, 4 tane komşu sütunda seçilen blokların gerekli şev açısını sağlamak için uyum içinde olmalarını engelleyebilmektedir (Wright, 1987).

Koenigsberg'in algoritmasında P, :$_{k}$ değeri kat, sütun ve kesit sırasını takip ederek bütün bloklar için hesaplanır. Bu işlemin sonucunda, bj j$_{k}$ bloğu için şu değerler bilinmektedir:

$$P_{i,j,k},\ \widetilde{Si},\ \widetilde{BSi},\ \widetilde{SBSi}\ \text{ve}\ \widetilde{SSBSi}.$$

^{p}O,J,K değerinden başlayarak geriye doğru gidilir ve belirtilmiş olan komşu bloklar takip edilerek optimum nihai sınır bulunmuş olur. P_{0} j$_{K}$ değeri, bütün şartların yerine getirilmesinden sonra cevher yatağından elde edilen optimum net değerdir.

3.4. Wilke ve Wrightin Üç Boyutlu Dinamik Programlama Algoritması

Koenigsberg'in algoritmasında yapılması gereken düzeltmeler sırasında optimum nihai sınırdan uzaklaşıldığını belirten Wright (1987), bu problemi ortadan kaldırmak için yeni bir algoritma geliştirmiştir. Bu algoritmada, bütün blokları kapsayan ve bu bloklar üzerinde oluşturulan hareketli koniler şeklindeki 3 boyutlu artırımlar kullanılmaktadır. Wright'in algoritmasında, her blok için ocak net değeri olan Pi, j, k'in hesaplanması için yeni bir eşitlik geliştirmiştir. Bununla beraber, 2 boyutlu dinamik programlama algoritmasına bu algoritmada da sadık kalınmıştır. Blok bj ;$_{k}$ için Pj -,$_{k}$ ocak net değeri, blok b; j$_{k}$ ile üzerindeki konide yer alan bloklardan ve blok b; j$_{k}$ ile uyum sağlayan komşu sınırlardan en iyi net değere sahip olanı ile hesaplanmaktadır. Pj ;$_{k}$ hesaplamaları sırasında bir bloka ait net değer, hesaplamaya birden fazla katılmamalıdır. Bu nedenle, hem Cjj$_{k}$ hareketli konisi içinde hem de komşu açık ocak sınırlarının net değerleri olan Pj ^ ^ k, $P_{ipj\cdot 1k}$ veP_{i+1}j . $_{1\ k}$ içinde yer alan bütün nrijj$_{k}$ blok net değerleriPj j$_{k}$ değerinden çıkarılmalıdır. Bu işlemi yapabilmek için aşağıdaki eşitlik geliştirilmiştir.

$$P_{i,j,k} = C_{i,j,k} + \max \begin{cases} I\,P_{i-1} - I\,M_{i-1} \\ I\,P_{i} - I\,M_{i} \\ I\,P_{i+1} - I\,M_{i+1} \end{cases} \qquad (5)$$

Burada;

Pj j$_{k}$ = blok bj j$_{k}$ için optimum ocak net değeri,

Cj :$_{k}$ = blok bj j$_{k}$ üzerindeki minimum hareketli koni değeri,

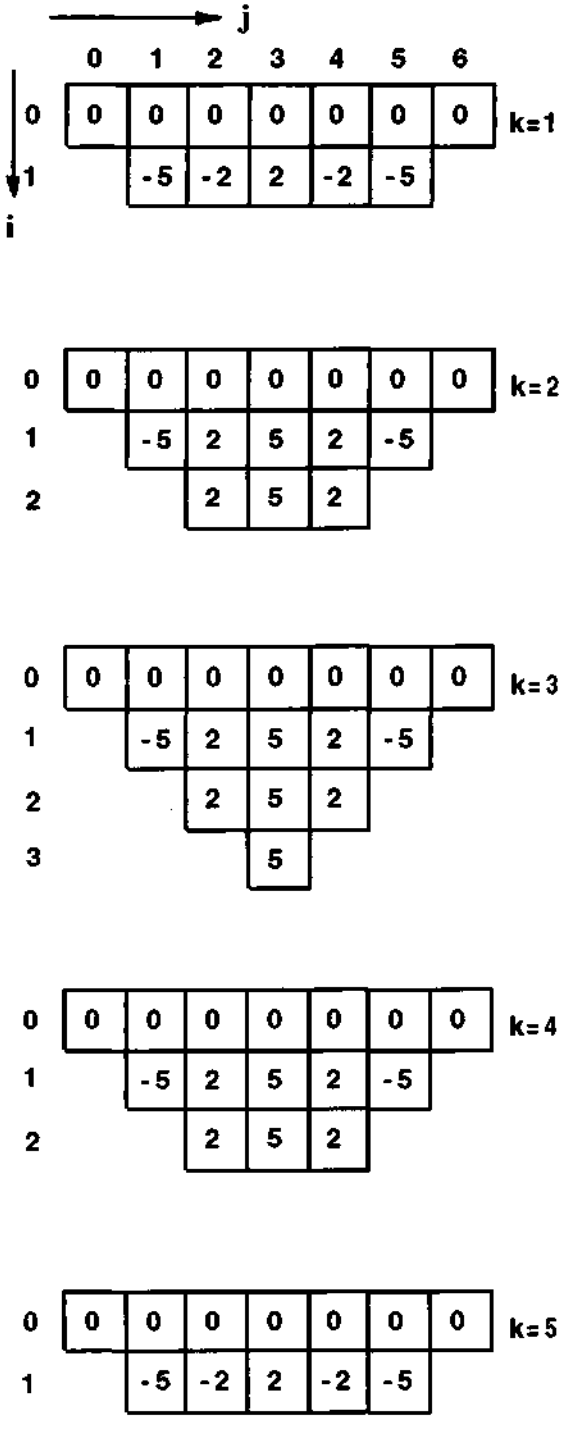
IP$_{L}$ =blok bg.1$_{k}$ için optimum ocak net değeri, L=j-1, i, i+1,

$^{1\ M}$L = ^{p}L,j-ı, k $^{n\ c}$i,j, k> $^{blok\ b}$L,j-ı, k üzerdeki ocak sınırı ile b,;$_{k}$ üzerindeki koninin kesişim bölgesinde yer alan blokların net değerinin toplamı, L = i-1, i,i+1 ,

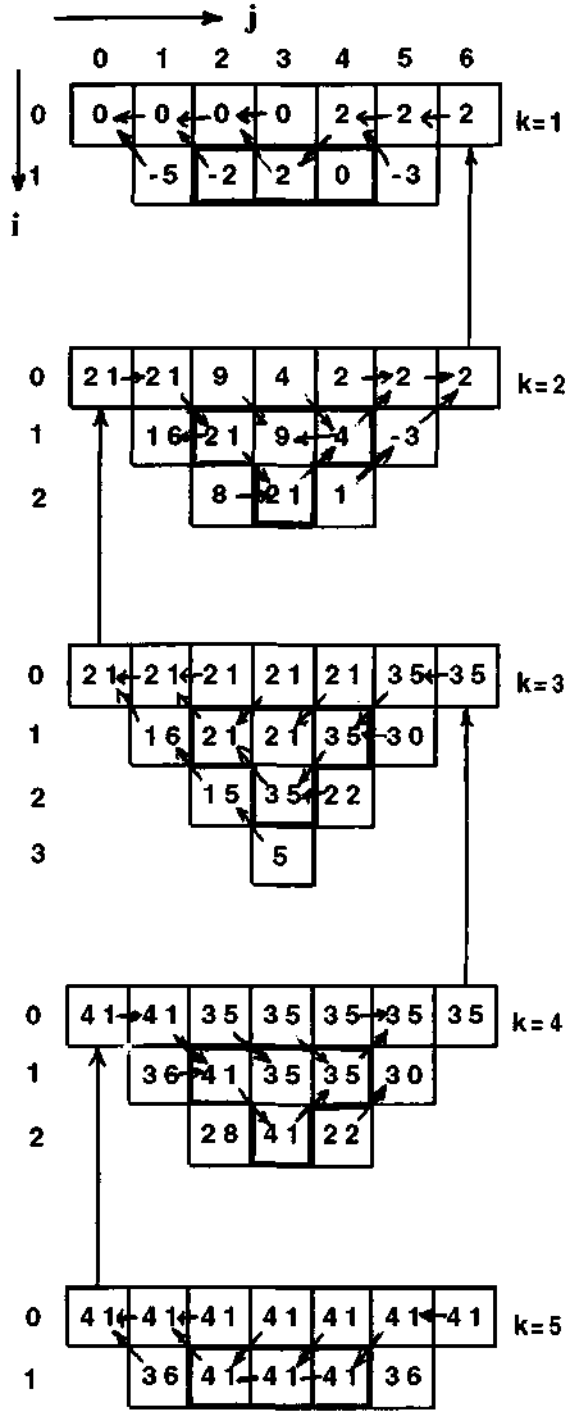
L= i+1 olduğunda $P_{i+1ij\cdot 1>k}$n C_{ijk} = C_{IJk}dir.

Bu durumda eşitlik 5 aşağıdaki gibi yazılabilir:

$$P_{i,j,k} = C_{i,j,k} + \max \begin{cases} I\,P_{i-1} - I\,M_{i-1} \\ I\,P_{i} - I\,M_{i} \\ I\,P_{i+1} - C_{i,j,k} \end{cases} \qquad (6)$$



Şekil 2. Örnek cevher yatağı ve blok net değerleri (Wright, 1987)



Şekil 3. Optimum nihai açık işletme sınırları (Wright, 1987)

Şekil 2'de, 5 kesitten oluşan bir örnek cevher yatağı ve blok net değerleri verilmiştir. Her blok için, Eşitlik 6 kullanılarak P y $_k$ değeri hesaplanmış ve bulunan optimum nihai sınır Şekil 3 üzerinde kalın çizgilerle gösterilmiştir.

Optimum nihai açık işletme sınırlarını bulmak için geliştirilmiş olan bu yeni 3 boyutlu dinamik programlama algoritması, bulunan sınır üzerinde elle düzeltme yapılmasını ve Koenigsberg'in algoritmasındaki düzeltmeleri ortadan kaldırmaktadır. Aynı zamanda, bulunan sınır sürekli optimumdur (Wright, 1987). Algoritmanın tek dezavantajı ise, bloklar için bulunan ocak sınırları ile bloklar üzerindeki hareketli konilerde yer alan blokların kontrolünden dolayı gerekli olan bilgisayar hafızası ve zamanının fazla oluşudur. Kontrol işleminin yapılmaması durumunda gerekli olan bilgisayar zamanı oldukça azalmaktadır.

4. SONUÇ

Optimum nihai açık işletme sınırlarını bulmak için kullanılan dinamik programlama algoritmaları içinde, Koenigsberg ve Wright tarafından geliştirilmiş olanlarının, bulunan optimum nihai sınırlar üzerinde yapılması gereken elle düzeltmeyi en az düzeye indirdiği, hatta tamamen ortadan kaldırdığı ve yapılması gereken düzeltmelerden dolayı optimumdan uzaklaşılması problemi ile karşılaşılmadığı belirtilmektedir. Buna karşılık, bu algoritmaların cevher yataklarına uygulanmaları sırasında ne derece başarılı oldukları, performansları ve hesaplama hızı hakkında bilgi veren kaynak sayısı yok denecek kadar azdır.

Dinamik programlama tekniği, optimum nihai açık işletme sınırını bulmak amacıyla geliştirilen teknikler içinde en kolay ve hızlı olmasına karşın, açık işletmenin değişik bölgelerinde değişik şev açılarına göre optimum nihai sınır bulmak bu teknikle henüz olanaklı değildir. Bunedenle 3 boyutlu dinamik programlama tekniği üzerindeki çalışmaların bundan sonra da devam edeceğini ve yeni algoritmaların geliştirileceğini şimdiden söylemek olanaklıdır.

KAYNAKLAR

JOHNSON, T.B. ve SHARP, R.W., 1971; "A Three-Dimensional Dynamic Programming Methods for Optimal Ultimate Open Pit Design", U.S. Bur. Min., RI 7553.

KIM, Y.C., 1978; "Ultimate Pit Design Methodologies Using Computer Models - The State of the Art", Min. Eng., Vol. 30, pp 1454-1459.

KOENİGSBERG, E., 1982; "The Optimum Contours of an Open Pit Mine: An Application of Dynamic Programming", Proa. 17 th APCOM Symp. Soc. Min. Eng., AIME, pp 274-287.

LERCHS, H. ve GROSSMAN, I. F., 1965, "Optimum Design Of Open Pit Mines", Can. Inst. Min, Bull., Vol. 58, pp 47-54.

LIZOTTE, Y., 1988; "The Economics of Computerized Open-pit Design", Int. J. of Sur. Min., Vol. 2, pp 59-78.

SHENGGUI, Z. ve STARFIELD, A. M., 1985; "Dynamic Programming with Colour Graphics Smoothing for Open-pit Design on a Personal Computer", Int. J. of Min. Eng., Vol.3, pp 27-34.

ÜNAL, A. ve YALÇIN, E., 1989; "Açık Ocak Nihai Sınırlannın Bilgisayar Destekli Tasanmı ve Batı Kef Krom Yatağına Uygulanması", Türkiye Mad. Bil. Tek. 11. Kong., s 1-19.

WRIGHT, E. A., 1987; "The Use of Dynamic Programming for Open Pit Mine Design: Some Practical Implications", Min. Sci. and Tech., Vol. 6, pp 79-104.

YALÇIN, E., 1991; "Açık İşletme Dizaynı için Üç Boyutlu Di namik Programlama Tekniği", Madencilik, Cilt 30, Sayı: 2, s.17.

SONDAJ YÖNTEMLERİ VE UYGULAMALARI

Atilla YALÇIN
Maden Yüksek Mühendisi

TMMOB MADEN MÜHENDİSLERİ ODASI YAYINI

Fiyatı
Üye :25.000.-TL
Öğrenci :20.000.-TL
Diğer :50.000.-TL

İsteme Adresi: TMMOB Maden Mühendisleri Odası
Selanik Caddesi 19/3, 06650 Kızılay-Ankara