Pamukkale Üniversitesi
Mühendislik Bilimleri Dergisi
Cilt 18, Sayı 2, 2012, Sayfa 85-94

Macar Algoritmasının Sıfırları Kapatma Alt Yordamı Üzerine

On a Subroutine for Covering Zeros in Hungarian Algorithm

Murat Erşen BERBERLER*, Onur UĞURLU, Gözde KIZILATEŞ

Dokuz Eylül Üniversitesi, Fen Fakültesi, Bilgisayar Bilimleri Bölümü, 35160, Buca, İzmir

Geliş Tarihi/Received : 25.06.2011, Kabul Tarihi/Accepted : 19.10.2011

ÖZET

Macar algoritması bilgisayar bilimleri literatüründe en çok bilinen yöntemlerden birisidir. Bu yöntem ile maliyet matrisi her adımda sistematik bir şekilde yeni bir indirgenmiş matrise dönüştürülerek atama problemine çözüm getirilmektedir. Algoritmanın alt yordamında matriste sıfır içeren tüm hücreler en az sayıda çizgi ile kapatılmakta ve çizgilerin durumuna göre matris üzerinde işlem yapılmaktadır. Bu makalede literatürdeki en az sayıda çizgi ile kapatma teknikleri incelenecek ve yeni bir yöntem önerisinde bulunularak hesaplama denemelerinin sonuçları tartışılacaktır.

Anahtar Kelimeler: *Macar algoritması, En büyük eşleme, Atama problemi.*

ABSTRACT

The Hungarian algorithm is one of the most well-known methods in computer science literature. By this method, in each step the cost matrix is systematically reduced to a new matrix in order to obtain an optimal solution for the assignment problem. The subroutine of the algorithm includes determining the minimum number of lines needed to cover all zeros in the reduced cost matrix and modifying the matrix according to the number of lines. In this paper, firstly the methods in literature including the covering all zeros with a minimum number of lines are examined, then a new method is proposed and computational experiments are discussed.

Keywords: *Hungarian algorithm, Maximum matching, Assignment problem.*

1. GİRİŞ

Atama problemi, adet kaynağın maliyeti en küçükleyecek şekilde adet hedefe atandığı problem türüdür. Problemin matematiksel modeli aşağıdaki gibidir (Bakır ve Altunkaynak, 2003):

$$Enk\ x_0 = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (1)$$

Kısıtlar :

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1,2,...,n) \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1,2,...,n) \qquad (3)$$

$$x_{ij} = \quad 0 \quad veya \quad 1 \qquad (4)$$

Modelde (2) kısıtı her bir kaynağın sadece bir hedefe, (3) kısıtı ise her bir hedefe sadece bir kaynağın atanmasını zorunlu kılmaktadır. Karar değişkenleri olan x_{ij}'ler ise yalnızca 0 veya 1 değerlerini almaktadır.

Atama problemi, ulaştırma probleminin özel bir durumu olmasına rağmen ulaştırma problemini çözmek için kullanılan simpleks yöntemi atama problemi için etkin bir yöntem değildir. Problemin doğasına uygun olarak Kuhn tarafından 1955'te geliştirilen

* Yazışılan yazar/Corresponding author. E-posta adresi/E-mail address : murat.ersen.berberler@ege.edu.tr (M. E. Berberler)

Macar Algoritması, problemi çözmek için kullanılan en iyi algoritmalardan birisidir. (Kara, 2000; Tütek ve Gümüşoğlu, 2000)

Macar Algoritmasının adımları şöyledir; (Kuhn, 1955).

Adım 1. *nxn* boyutlu *C* maliyet matrisinin her bir satırındaki en küçük eleman bulunur. Bu değerler karşılık gelen satırlardaki her bir maliyetten çıkarılarak yeni bir matris oluşturulur. Yeni matriste bu defa her bir sütunun en küçük değeri karşılık gelen sütunlardaki maliyet değerlerinden çıkarılır. Böylece indirgenmiş maliyet matrisi elde edilir.

Adım 2. İndirgenmiş maliyet matrisindeki tüm sıfır değerli hücrelerden geçecek şekilde en az sayıda yatay ve dikey çizgiler çizilir. Eğer tüm sıfır değerlerini örtecek şekilde *n* sayıda çizgi çizilmişse örtülmüş sıfırlar arasında en iyi çözüm mevcuttur. Aksi durumda sonraki adım olan Adım 3'e geçilir.

Adım 3. Çizgi ile örtülmeyen en küçük eleman bulunur. Bu değer indirgenmiş maliyet matrisindeki örtülmemiş elemanlardan çıkarılır ve iki çizgi ile örtülmüş elemanlara eklenir ve Adım 2 'ye dönülür.

İyi algoritmaların etkin oldukları kadar genel itibariyle basit ve kolay programlanabilir olmaları da gerekir. Ancak Macar algoritmasının adımında maliyet matrisinde bulunan sıfırları örten çizgilerin çizilmesi işlemi kolay programlanabilir değildir. Problemin büyüklüğü arttıkça zorlaşan bu işlemi etkin ve basit bir şekilde gerçekleştirmek için çalışmalar yapılmaktadır. Literatürde bu konuda birçok yöntem mevcuttur.

İlk olarak Gillett (1976) tarafından, küçük problemler için kolayca uygulanabilen basit ve etkili bir sezgisel yöntem önerilmiştir. En yüksek ağ akış yöntemi ise, Bazaraa v.d., (1990) tarafından önerilen farklı bir yöntemdir. Papadimitrou ve Steiglitz (1982) oldukça karmaşık olan ikiye ayrılabilir çizgelerde ağırlıklı eşleme adlı bir yöntem geliştirmişlerdir. Lotfi (1989), Gillett tarafından geliştirilen yöntemin bazı problemler için çalışmadığını ispatlamış; ayrıca Bazaraa ve diğerleri tarafından geliştirilen yöntem üzerine çalışmış, probleme yeni bir bakış açısı geliştirmiştir. Ancak geliştirdiği bu bakış açısı Gillett yönteminden daha karmaşıktır (Öner ve Ülengin, 2003).

Bu makalede indirgenmiş maliyet matrisindeki tüm sıfırları kapatmayı sağlayan en az çizgi sayısını bulmak için literatürdeki en küçük tepe örtüsü ve en yüksek ağ akış yaklaşımları etraflıca incelenecek ve geri izleme tekniğine dayanan yeni bir algoritma önerilerek hesaplama denemeleri yapılacaktır.

2. EN KÜÇÜK TEPE ÖRTÜSÜ YAKLAŞIMI

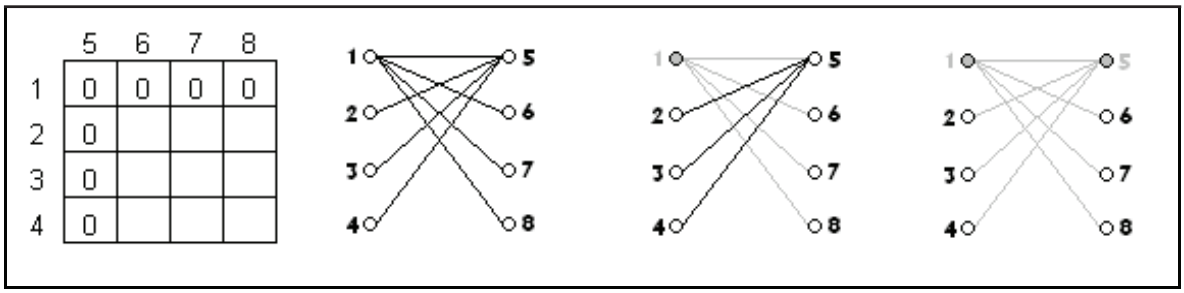
Bu yaklaşımda yönsüz ve iki parçalı bir G=(V_1,V_2,E) çizgesi oluşturulur. Matristeki satır numaraları V_1 tepeler kümesi ile sütun numaraları ise V_2 tepeler kümesiyle gösterilir. İndirgenmiş maliyet matrisinde yer alan her sıfırın bulunduğu satırı temsil eden V_1 kümesindeki tepe ile sütunu temsil eden V_2 kümesindeki tepe çiftini birleştiren bir ayrıt çizilir. Burada maliyet matrisindeki sıfırları kapatma problemi iki parçalı çizgede en küçük tepe örtüsü problemine dönüşür ve *NP-tam* karmaşıklık sınıfından olan en küçük tepe örtüsü problemi iki parçalı çizgelerde tipinde olan en büyük eşleme problemine dönüştüğü için problem belirlenimci (deterministic) tekniklerle çözülür. Aşağıdaki örnek (Şekil 1) bu yaklaşımı göstermektedir; örnekte satır indisleri 'ler aynen kullanılırken sütun indisleri *i*'ler etiketlemede karışmaması için *n* değeri ile toplanır ve *n*+*j* şeklinde kullanılır. $V_1 = \{1,2,3,4\}$, $V_2 = \{5,6,7,8\}$ Örnek çizgede en küçük tepe örtüsü probleminin çözümü $EKTÖ = \{1,5\}$ kümesi olup, bu tepelere bitişik ayrıtları çizgeden sildiğimizde maliyet matrisindeki tüm sıfırları kapatmış oluruz.

En küçük tepe örtüsü yaklaşımına ait hesaplama denemelerini yapmak için aşağıdaki matematiksel model yazılıp test problemleri GAMS (General Algebraic Modeling System) paketinin 22.5 sürümünde çözdürülmüştür.

$$\min Z = \sum_{k=1}^{2n} x_k$$

$$x_i + x_{n+j} \geq 1 \quad \forall (i, n+j) \in E(G) \quad i \in V_1 \quad n+j \in V_2$$

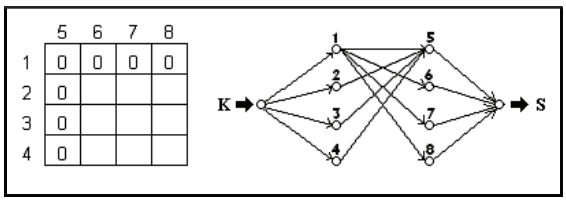
$$x_k \in \{0,1\} \quad k = 1..2n$$



Şekil 1. En küçük tepe örtüsü yaklaşımı.

3. EN YÜKSEK AĞ AKIŞ YAKLAŞIMI

Bu yaklaşımda yönlü bir G=(V,E) çizgesi ve ardından bir ağ kurulmaktadır (Şekil 2). Maliyet matrisindeki satırları 1,2,…,m düğümleri; sütunları ise m+1,m+2,…2m düğümleri temsil etmektedir. Eğer indirgenmiş matriste (i,j) elemanı sıfır ise o zaman (i) düğümü ile (m+j) düğümü arasında bir yönlü ayrıt çizilmektedir. Bu düğümlere ek olarak bir K kaynak düğümü ile bir S son düğümü daha ağa eklenmektedir. K düğümünden satırları temsil eden tüm düğümlere ve sütunları temsil eden tüm düğümlerden S düğümüne yönlü ayrıtlar çizilir. Bu şekilde oluşan ağ üzerindeki akış kapasiteleri 1 birim olarak sınırlandırılır. Bu ağda K düğümünden S düğümüne en yüksek akış miktarı indirgenmiş maliyet matrisindeki tüm sıfırları örten en az çizgi adedini vermektedir. Ayrıca bu ağ akış problemi bilinen etiketleme yöntemi ile çözülürse, üzerinden akış geçen düğümler maliyet matrisinde çizgilerin çekileceği satır ve sütunları vermektedir (Bazaraa v.d., 1990; Öner ve Ülengin, 2003).



Şekil 2. En yüksek ağ akış yaklaşımı.

Bu ağdaki en yüksek akış probleminin çözümü $K \to 1 \to 6 \to S$ ve $K \to 2 \to 5 \to S$ akışları olup toplam akış miktarı 2 olduğu için indirgenmiş maliyet matrisindeki sıfırları da 2 çizgiyle kapatmış oluruz.

En yüksek ağ akış yaklaşımına ait hesaplama denemelerini yapmak için Ford-Fulkerson ve Edmond-Karp'ın algoritmaları C dilinde yazılıp test problemleri çözdürülmüştür. Ford-Fulkerson algoritmasının karmaşıklığı $O(|E|f^*)$ olup $|E|$ ayrıtlar kümesinin boyutunu, f^* da en büyük akışı temsil etmektedir, dolayısıyla bu algoritmanın karmaşıklığı sonuca bağımlıdır. Öte yandan Edmond-Karp algoritması $O(|V||E|^2)$ karmaşıklığında olduğundan bu algoritma sonuca bağımlı olmayıp bazı problemler için Ford-Fulkerson algoritmasından daha iyidir (Cormen v.d., 2001).

Ancak, Edmond-Karp algoritmasının kötü yanı çok fazla bellek kullanmasıdır, dolayısıyla problem matrisi belirli bir boyutu aştıktan sonra program bellek hatası vermektedir. Dolayısıyla test problemlerini çözmek için en yüksek ağ akış yaklaşımı için Ford-Fulkerson algoritması kullanılmış olup bu algoritma kullanılarak yazılan MaxFlow.c kaynak kodu Linux ortamında gcc 4.2 derleyicisiyle –O3 optimizasyonu kullanılarak derlenmiştir.

4. GERİ İZLEME TEKNİĞİNE DAYANAN EN AZ ÇİZGİ ALGORİTMASI

Geri izleme tekniği bilgisayar bilimlerinde kullanılan en temel tekniklerden birisidir. Bu teknikte problemin izin verdiği ölçüde aday çözümlerin sayısını küçültecek durumlar sistematik bir şekilde dikkate alınırsa etkin algoritmalar geliştirilebilir. Bu duruma

en güzel örnek 8 vezir bulmacasıdır. Bu bulmacada 8x8 lik satranç tahtası üzerinde 8 vezirin birbirlerini tehdit etmeyecek şekilde yerleştirilmeleri istenmektedir. Burada kaba kuvvet tekniği kullanılırsa $8^8 = 16777216$ durumu değerlendirmek gerekecekken geri izleme mantığı kullanılırsa toplamda 37109 durum değerlendirilerek simetriler dahil 92 adet çözüm bulunmuş olur. 1-1 2-5 3-8 4-6 5-3 6-7 7-2 8-4 vektörü de bu 92 adet çözümden bir tanesidir.

Yukarıdaki paragrafta ipuçları verildiği üzere önerilen algoritmanın çalışma zamanını azaltacak verimli hamleler göz önünde bulundurulmuştur. Bu amaca ulaşmak için ilk olarak indirgenmiş maliyet matrisindeki sıfırların adedi sayılmakta ve $n^2 - 2n + 2$'den fazla sıfır olduğu takdirde sonuç *n* satır (veya *n* sütun) olarak yazılıp algoritma sonlanmaktadır. Bu adımın matematiksel ifadesi Lemma 4.1 ve Teorem 4.2'de verilmektedir.

Lemma 4.1. *k* adet doğru $(k+1)x(k+1)$ boyutlu matris üzerinde sadece yatay ve dikey doğrultularda olmak şartıyla en çok $\left\lfloor \frac{k}{2} \right\rfloor + 1$ farklı şekilde çizilebilir.

İspat : *k* elemanlı bir kümenin tüm alt kümelerini eleman sayılarına göre *k+1* adet gruba ayırabiliriz. *k+1* sayısı Pascal üçgeninin *k*. satırındaki eleman sayısına karşılık gelir. Yani;

$$\underbrace{\binom{k}{0}\ \binom{k}{1}\ \binom{k}{2}\ \cdots\ \binom{k}{k}}_{k+1}$$ 'dir.

Burada, problem; *k* toplam çizgi sayısını, *m*'de dikey çizgi sayısını göstermek üzere, kaç farklı $\binom{k}{m}$ binom değeri olduğunu bulma problemine dönüşür. $\binom{k}{m}$ dikey çizgileri gösterirken $\binom{k}{k-m}$ de yatay çizgileri göstermektedir. $\binom{k}{m} = \binom{k}{k-m}$

'den dolayı simetri vardır ve *k*+1 adet eleman içinden ortanca elemana göre eşit uzaklıktaki binom değerleri eşittir ve ortanca elemanın konumu da $\left\lfloor \frac{k}{2} \right\rfloor + 1$ 'dir. Sonuç olarak $\left\lfloor \frac{k}{2} \right\rfloor + 1$ değeri doğruları çizerken denenebilecek tüm alternatiflerin sayısını verir.

Teorem 4. 2. *nxn* boyutlu indirgenmiş maliyet matrisindeki sıfırların sayısı $n^2 - 2n + 2$ değerinden büyük ise matristeki tüm sıfırları örtmek için adet doğru çizilmelidir.

İspat : *nxn* boyutlu indirgenmiş maliyet matrisinde *n*-1 adet doğru ile örtülebilecek en fazla sıfır adedi $n^2 - 2n + 2$'dir. Bu değere ulaşmak için çizilen doğruların altında kalan hücrelerin hepsinin sıfır içerdiği kabul edilmektedir ve bu durum aşağıdaki tablolarda (Tablo 1) gri renkli hücrelerle gösterilmektedir. Lemma 4.1'de ispatlandığı üzere *n*-1 adet doğru *nxn* boyutlu matriste sadece yatay ve dikey doğrultularda olmak şartıyla en çok $\left\lfloor \frac{n-1}{2} \right\rfloor + 1$ farklı şekilde çizilebilir ve bunların içinden en fazla sıfırı örten çizim modelinde *n*-2 adet doğru birbirine paralel olup kalan doğru ise paralel *n*-2 adet doğruya diktir. Böylece en az kesişim sayısı olan *n*-2'ye sahip çizim modeli elde edilmiş olur (Tablo 1. 2). Geriye kalan $\left\lfloor \frac{n-1}{2} \right\rfloor$ durumda ise ya indirgenmiş maliyet matrisinin gereği olan her satır ve her sütunda en az 1 adet sıfır olması koşulu sağlanmaz (Tablo 1.1 en alt satırda örtülen sıfır yok) ya da doğruların kesişim noktaları *n*-2'den fazla olduğu için örtülen sıfır adedi azalır (Tablo 1. 3 ve Tablo 1. 4). *n*=7 için olası $\left\lfloor \frac{7-1}{2} \right\rfloor + 1 = 4$ durum aşağıda gösterilmiştir.

Sonuç olarak *n*-1 adet doğru ile örtülebilecek en fazla sıfır adedi $n^2 - 2n + 2$ ise, $n^2 - 2n + 2$ 'den fazla sıfır içeren bir matris *n* adet doğru ile örtülmek zorundadır.

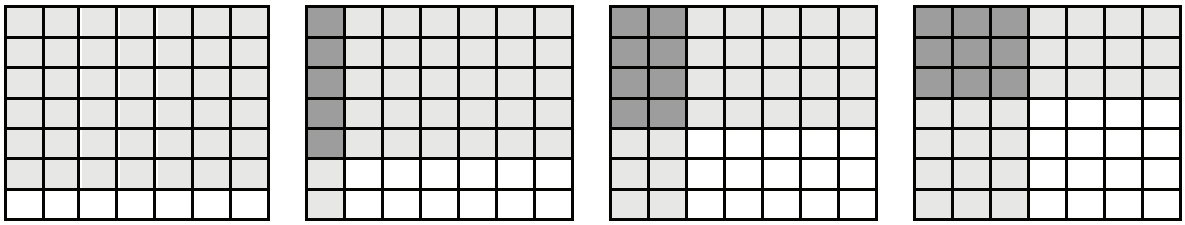
Bir *G* çizgesinin ayrıtlarının herhangi bir alt kümesi *M* olsun. *M*'deki ayrıt çiftlerinin hiçbirisi çizgesinde birbiri ile bitişik değilse *M*'ye *G*'nin bir eşlemesi denir. Eğer, *G* çizgesinin birden fazla eşleme kümesi var ise en çok elemana sahip olan kümeye en büyük eşleme denir. Eğer *G* çizgesinin tüm tepeleri en büyük eşleme içinde yer alıyorsa buna mükemmel eşleme

denir. Önerilen algoritmada en büyük eşleme (mümkünse mükemmel eşleme) geri izleme tekniğiyle bulunmaktadır. *nxn*'lik indirgenmiş maliyet matrisinde sütunlarda yer alan sıfırları baz alarak bir en büyük eşleme bulunmaya çalışılır. Geri izleme tekniğinin denediği durumları azaltmak için aşağıdaki yaklaşımlar önerilmektedir.

4. 1. Sıralama Özelliği

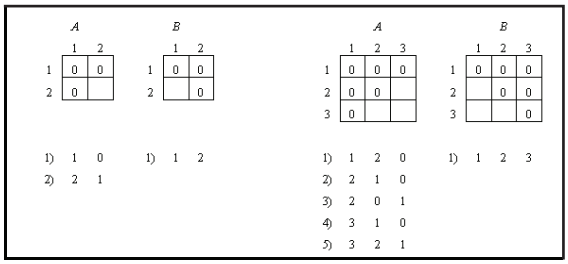
İçerdikleri 0 adetlerine göre tüm sütunlar küçükten büyüğe sıralanır. Dolayısıyla her sütundaki atama, satırları dolaşan bir döngüyle elde edilir. Burada sıralamanın amacı iç içe döngüleri küçük iterasyonlular dışarıda büyük iterasyonlular içeride olacak şekilde ayarlayıp en büyük seçimin en kısa sürede bulunmasını sağlamaktır. Örneğin; i indisi satırları ve j indisi sütunları temsil etsin, $i + j \leq n + 1$ olmak üzere A tipindeki matrisler ile $i \leq j$ olmak üzere B tipindeki matrisler üzerindeki iterasyon sayıları aşağıda (Tablo 2) görülmektedir. A tipindeki matrisler en kötü durumu, B tipindeki matrisler ise sıralama özelliği kullanıldığında önerilen algoritma sonucu oluşan en iyi durumu göstermektedir.

Tablo 1. n=7 için karşılaşılabilecek 4 farklı durum.



Tablo 1.1 Tablo 1.2 Tablo 1.3 Tablo 1.4

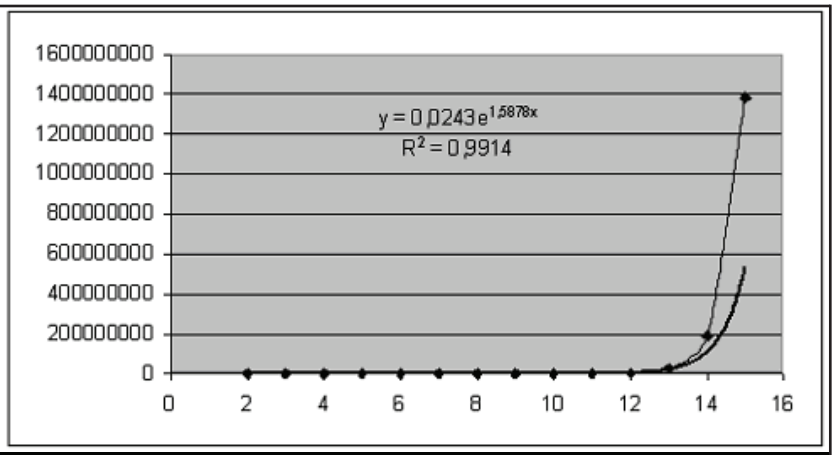
Tablo 2. Sıralama özelliği.



Burada, A tipindeki matrislerin iterasyon sayılarının n'nin büyük değerleri için üstel şekilde artacağı aşağıda (Şekil 3) görülmektedir. Tabloda yer alan iterasyon değerleri literatürde Bell Sayıları veya Üstel Sayılar olarak anılmaktadır. Oysa B tipindeki matrislerde iterasyon sayısı her zaman 1'dir. [http://www.research.att.com/~njas/sequences/].

Sıralama özelliği kullanılmadığında A tipindeki matrisler için n'in küçük değerlerinde bile çok fazla çalışma zamanı ile karşılaşılacağı aşağıda (Tablo 3) görülmekte olup sıralamanın çözüme katkısı vurgulanmaktadır.

n	Iterasyon
2	2
3	5
4	15
5	52
6	203
7	877
8	4140
9	21147
10	115975
11	678570
12	4213597
13	27644437
14	190899322
15	1382958545



Şekil 3. A tipindeki matrislerin iterasyon sayıları.

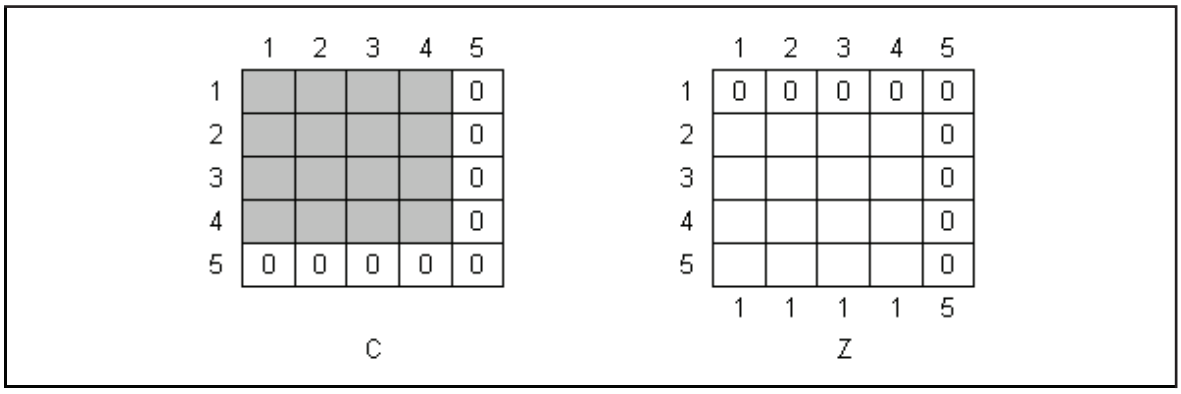
Tablo 3. A tipindeki matrislere ait çalışma süreleri.

n	Süre (sn)
10	0
11	0,04
12	0,27
13	1,83
14	12,72
15	94,69
16	1025,43

4. 2. Sıfırların Konumlarını Değerlendirme Özelliği

Sütunlardaki 0'ları satır bazında dolaşma işlemi 0'lar ardışık olmadığında ekstra maliyet getirmektedir çünkü 0 içermeyen satırların da iterasyonda dikkate alınması geri izleme tekniğinin çözüm süresinin artmasına neden olur. Oysa önerilen algoritmada sütunlardaki sıfırların bulunduğu pozisyonları içeren ikinci bir matris kullanarak 0 içermeyen satırların iterasyonca değerlendirilmesi önlenmiş olur. Bunun algoritmaya yer karmaşıklığı maliyeti olmasına $O(n^2)$ rağmen çalışma zamanından çok büyük oranda tasarruf sağlanmaktadır. En çok kazanç sağlanan durumu irdelemek için aşağıdaki (Tablo 4) C matrisini göz önüne alalım. Eğer C matrisi algoritmada kullanılırsa *nxn*'lik bir matriste $n^2 - 2n + 1$ tane olan gri renkli hücrelerle gösterilen kısımlar döngülerin boşuna dönmesini gerektirecek ve zaman kaybına neden olacaktır. Oysa hangi sütunda kaç tane 0 olduğunu ve bunların hangi hücrelerde bulunduğunu içeren Z matrisi de algoritmada kullanılırsa döngüler en etkin şekilde çalışmış olacaktır.

Tablo 4. Sıfırların konumlarını değerlendirme özelliği.



Aşağıdaki tabloda (Tablo 5) Z^-, Z matrisinin kullanılmadığı; Z^+, Z matrisinin kullanıldığı durumlardaki çalışma zamanını; Z^- / Z^+ oranı da problem boyutu olan n 'e bağlı kazancı göstermektedir. Z matrisi kullanıldığında algoritmanın çalışma zamanının ortalama 3 kat azaldığı görülmektedir.

Tablo 5. Sıfırların konumlarını değerlendirme özelliğine ait çalışma süreleri.

	Süre (sn)		
n	Z^-	Z^+	Z^- / Z^+
1000	0,047	0,016	2,9375
1500	0,093	0,031	3
2000	0,188	0,062	3,032258
2500	0,266	0,093	2,860215
3000	0,406	0,125	3,248

4. 3. Erken Çıkış Özelliği

Geri izleme tekniği kullanılarak geliştirilen algoritmada tüm ihtimaller denenmeden ilk en büyük eşlemede çıkılması sağlanarak zamandan çok büyük miktarlarda tasarruf edilmektedir. Bu özellik kullanıldığında elde edilecek en büyük zaman kazancı Tablo 1.2'de bahsedilen problem türünde sağlanır, ayrıca bu özellik kullanılmadığında küçük boyutlu problemlerde bile elde edilen çok yüksek çalışma zamanları aşağıdaki tabloda (Tablo 6) görülmektedir. Oysa erken çıkış özelliği kullanıldığında n en az 4000 seçilirse aşağıdaki en büyük süreye ulaşılmakta olup bu özelliğin ne kadar önemli olduğu bir kez daha görülmektedir.

Tüm bu bilgiler göz önüne alındığında önerilen algoritmanın karmaşıklığı $O(n^3)$'tür. $O(n^2)$

iç döngü olarak çalışmakta ve matris içinde dolaşmada kullanılmaktadır, iç döngüyü kullanan dış döngüde en çok matrisin boyutu olan O(n) kadar çalışacağından (çünkü n boyutlu bir matris en çok n doğru ile kapatılabilir) toplamda karmaşıklık $O(n^2)$. $O(n)=O(n^3)$ bulunur. Önerilen algoritmanın diğerlerinden daha hızlı çalışması yukarıda bahsedilen 4.1., 4.2. ve 4.3. özelliklerini kullanmasından kaynaklanmaktadır.

Tablo 6. Erken çıkış özelliği kullanılmadığında elde edilen çalışma süreleri.

n	Süre (sn)
8	0
9	0,015
10	0,265
11	3,453
12	48,859
13	744,265

5. HESAPLAMA DENEMELERİ

Geri izleme tekniğine dayanan ve C dilinde yazılan mebGO.c programı Linux ortamında gcc 4.2 derleyicisiyle –O3 optimizasyonu kullanılarak derlenmiştir. P1 (% 20), P2 (% 50), P3 (% 80) test problemleri yüzdeleri oranında 0 içerecek ve 0'ların konumları rasgele olacak şekilde (her satır ve sütunda en az bir adet sıfır olma garantisi altında) üretilmiştir. P4 test problemleri 4.1. sıralama özelliğinde bahsedilen A tipindeki matrislerden oluşmakta olup son olarak P5 test problemleri ise Tablo 1.2 de bahsedilen matris tiplerinden oluşmaktadır. Aşağıda (Tablo 7) her bir problem tipinde tüm yaklaşımlar için elde edilen çalışma süreleri görülmektedir.

Tablo 7. P1, P2, P3, P4 ve P5 problem tipleri için çalışma süreleri.

P1 (%20)			
n	Gams	MaxFlow	mebGO
100	0,025	0,01	0
200	0,151	0,08	0,01
300	0,294	0,21	0,03
400	0,551	0,44	0,06
500	0,988	0,84	0,1
600	1,747	1,42	0,21
700	2,752	2,3	0,23
800	3,837	3,46	0,42
900	5,067	4,96	0,44
1000	6,465	6,82	0,58

P2 (%50)			
n	Gams	MaxFlow	mebGO
100	0,086	0,01	0
200	0,404	0,08	0,02
300	0,893	0,26	0,05
400	1,914	0,58	0,13
500	3,171	1,02	0,2
600	4,765	1,78	0,32
700	6,643	2,81	0,51
800	8,905	4,21	1,04
900	11,457	6,01	1,68
1000	14,483	8,24	2,52

P3 (%80)			
n	Gams	MaxFlow	mebGO
100	0,163	0,01	0
200	0,58	0,09	0,03
300	1,659	0,23	0,07
400	3,177	0,5	0,15
500	4,97	0,95	0,28
600	7,376	1,66	0,58
700	10,382	2,58	1,19
800	13,672	3,9	1,82
900	17,614	5,54	2,77
1000	21,83	7,55	4,18

P4			
n	Gams	MaxFlow	mebGO
100	0,081	0,01	0
200	0,424	0,06	0,02
300	1,441	0,24	0,05
400	3,263	0,41	0,09
500	6,521	0,79	0,21
600	12,441	1,36	0,37
700	19,957	2,15	0,61
800	30,361	3,22	1,01
900	43,502	4,6	1,61
1000	60,238	6,29	2,42

P5			
n	Gams	MaxFlow	mebGO
100	0,177	0,01	0
200	0,751	0,08	0,03
300	2,021	0,23	0,1
400	3,868	0,42	0,21
500	6,318	0,81	0,44
600	9,35	1,38	0,9
700	12,714	2,25	1,42
800	16,825	3,38	2,27
900	21,766	4,79	3,49
1000	27,064	6,54	5,18

Hesaplama denemelerinin sonuçlarından görüleceği üzere önerilen algoritma ele alınan diğer tekniklerden daha hızlı çalışmaktadır. Ayrıca bu algoritmayı Macar algoritması içinde bir alt yordam olarak kullanan mebGOX.c programı yazılıp test problemleri üzerinde hesaplama denemeleri yapılmış olup izleyen bölümde detayları anlatılacaktır. Makalede bahsi geçen tüm kaynak kodlar ve test problemleri http://fen.ege.edu.tr/~murateb/maskayu/ adresinde bulunmaktadır.

5. 1. Maksimum İterasyona Sahip Atama Problem Türü

Çözüm için Macar Algoritması kullanıldığında maksimum iterasyonlu problemleri elde etmek için örnek matrisleri A=a[i,j]=i*j ifadesiyle oluşturabiliriz. Bu tip matrislerin özelliği sıfırları kapatırken ilk iterasyonda en az çizgi sayısı olan 2 ile başlayıp, sonraki her bir iterasyonda matrise sadece bir adet 0 ilave etmeyi garanti etmesidir.

Yukarıdaki örnekten (Tablo 8) görüldüğü üzere a[i,j]=i*j tipi matrise sahip problem Macar algoritmasıyla çözülüyorsa iterasyon sayısı problemin boyutu n'e bağlı olarak kolayca hesaplanabilir şöyle ki; matristeki sıfırları kapatan çizgilerin sayısının görülme adedi kapattıkları sıfır sayısının bir eksiği kadardır. Çünkü en büyük seçim olarak belirlenen sıfırların matris üzerindeki ilerleyişi (2,1)-(1,2); (3,1)-(2,2)-(1,3); (4,1)-(3,2)-(2,3)-(1,4) şeklinde olmakta ve bir sonraki en büyük seçimi belirleyen i+j=n özelliğindeki hücrelerden oluşan köşegene geçiş bir önceki adımda köşegen üzerinde bulunan sıfır adedi kadar olmaktadır. Bir başka deyişle (i,1) ve (1,j) hücreleri ilk adımdan itibaren sıfır olduğundan i+j=n köşegeni üzerinde yeni sıfırlar yaratmak için i+j-2 sayısı kadar adım beklenmesi gerekmektedir.

Tablo 8. n=4 için maksimum iterasyona sahip atama problem türü.

i/j	1	2	3	4
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12
4	4	8	12	16

0	1	2	3
0	2	4	6
0	3	6	9
0	4	8	12

0	0	0	0
0	1	2	3
0	2	4	6
0	3	6	9

0	0	0	0
1	2	3	**0**
2	4	6	0
3	6	9	0

1	0	**0**	0
0	**0**	1	2
0	1	3	5
0	2	5	8

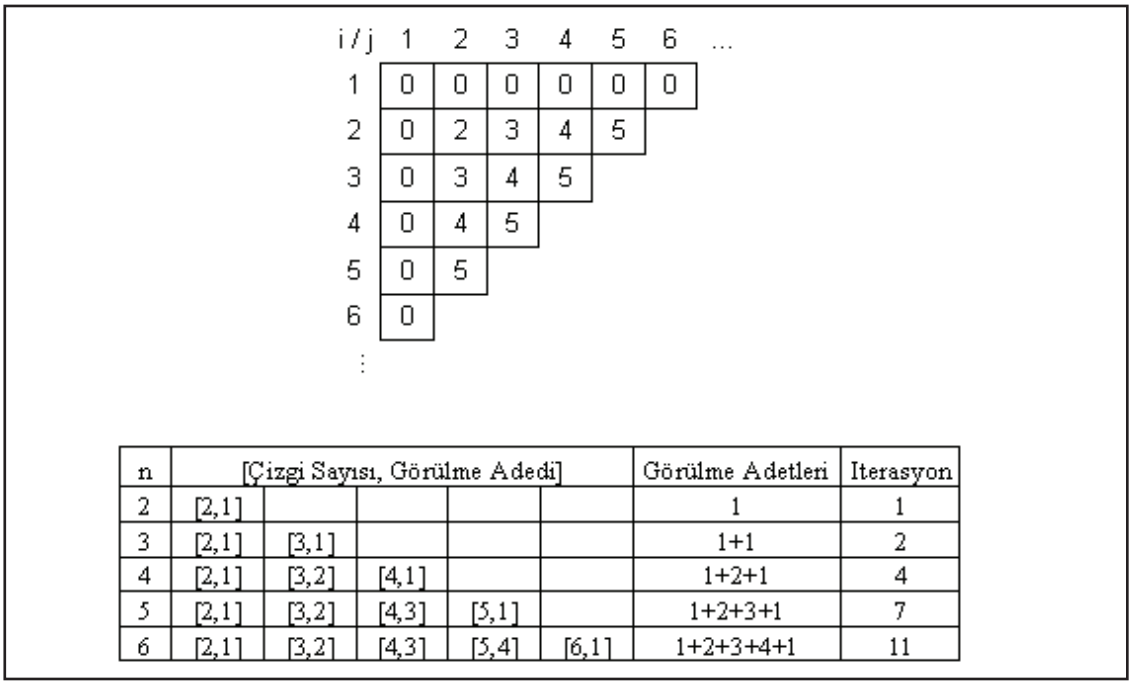
2	0	**0**	0
1	**0**	1	2
0	0	2	4
0	1	4	7

3	1	0	**0**
1	0	**0**	1
0	**0**	1	3
0	1	3	6

Tablodan (Tablo 9) görüldüğü üzere n'e bağlı iterasyon sayını veren formülü $\frac{(n-2)(n-1)}{2}+1$, çizgi sayılarının görülme adetlerini toplayarak bulabiliriz. Ayrıca bu tip matrislerin çözümünde elde edilen en iyi değerleri bulmak için de $\frac{n^3+3n^2+2}{6}$ formülü kullanılabilir.

Sonuç olarak atama problemini Macar algoritmasıyla çözerken en çok iterasyona sahip matris türü tespit edilmiş ve mebGO.c sıfırları kapatma programı Macar Algoritmasına uyarlanarak mebGOX.c programı yazılmıştır. Aşağıdaki Tablo 10'da en çok iterasyona sahip matrisi içeren farklı boyutlardaki problemlerin mebGOX, WinQSB 2.0 ve Lingo 8.0 paketleriyle çözüm süreleri görülmektedir.

Tablo 9. n'e bağlı iterasyon sayıları.



i / j	1	2	3	4	5	6	...
1	0	0	0	0	0	0	
2	0	2	3	4	5		
3	0	3	4	5			
4	0	4	5				
5	0	5					
6	0						
⋮							

n	[Çizgi Sayısı, Görülme Adedi]					Görülme Adetleri	Iterasyon
2	[2,1]					1	1
3	[2,1]	[3,1]				1+1	2
4	[2,1]	[3,2]	[4,1]			1+2+1	4
5	[2,1]	[3,2]	[4,3]	[5,1]		1+2+3+1	7
6	[2,1]	[3,2]	[4,3]	[5,4]	[6,1]	1+2+3+4+1	11

Tablo 10. En çok iterasyona sahip matris için çalışma süreleri.

		mebGOX		WinOSB 2.0		Lingo 8.0	
n	Obj	t (sn)	İterasyon	t (sn)	iterasyon	t (sn)	iterasyon
10	220	0	37	0,070	19	0,50	47
20	1540	0	172	0,511	39	0,75	26
30	4960	0	407	1,983	59	1,41	37
40	11480	0,01	742	5,428	79	2,43	49
50	22100	0,02	1177	12,118	99	4,44	58
60	37820	0,05	1712	23,664	119	2,18	68
70	59640	0,07	2347	41,910	139	2,80	79
80	88560	0,12	3082	68,909	159	3,80	88
90	125580	0,17	3917	107,425	179	5,11	98
100	171700	0,29	4852	155,063	199	4,62	108

6. SONUÇLAR

Macar algoritmasında yer alan sıfırları kapatma alt yordamına geri izleme tekniği ile yeni bir algoritma önerisi getirilmiştir. Önerilen algoritma C dilinde programlanarak literatürdeki mevcut yöntemlerle çeşitli test problemleri üzerinde kıyaslanmış ve bu örnekler üzerinde ele alınan yöntemlerden daha iyi olduğu görülmüştür. İleriye dönük olarak akademik çevrelerden gelecek geri bildirimler doğrultusunda algoritma geliştirilerek paket program haline getirilecektir.

7. KAYNAKLAR

Bakır, M. A., Altunkaynak, B. 2003. Tamsayılı Programlama: Teori, Modeller ve Algoritmalar, Nobel Yayın Dağıtım, Ankara.

Bazaraa, M., Jarvis, J., Sherali, H. 1990. Linear Programming and Network Flows, 2nd Ed., John Wiley & Sons, NewYork, USA.

Cormen T. H., Leiserson, C. E., Rivest R. L., Stein C. 2001. Introduction to Algorithms (second ed.). MIT Press and McGraw–Hill.

Gillett, B. E. 1976. Introduction to Operations Research: A Computer-Oriented Algorithmic Approach, McGraw-Hill, NewYork, USA.

Kara, İ. 2000. Doğrusal Programlama, Bilim Teknik Yayınevi, Ankara.

Kuhn, H.W. 1955. The Hungarian Method for the Assignment Problem, Naval Research Logistics Quarterly. (2), 1-2.

Lotfi, V. 1989. A Labeling Algorithm to Solve the Assignment Problem, Computers and Operations Research. (16), 397-408.

Öner, A., Ülengin, F. 2003. Atama problemi için yeni bir çözüm yaklaşımı, itüdergisi/d mühendislik, Cilt 2 (1), 73-79.

Papadimitrou, C.H., Steiglitz, K. 1982. Combinatorial Optimization, Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, N. J., USA.

Tütek, H.H., Gümüşoğlu, Ş. 2000. Sayısal Yöntemler Yönetsel Yaklaşım, Beta Basım A.Ş., İstanbul.