# EN KISA YOL PROBLEMİNDE ÇİZGE PARÇALAMA YÖNTEMİ KULLANILARAK YENİ BİR YAKLAŞIM [1]

**Mustafa Kemal BEŞER**
Eskişehir Osmangazi Üniversitesi, İktisadi ve İdari Bilimler Fakültesi

**Özet**
Bu çalışmada ilk olarak çizge kuramının temel kavramları verilmiş, en kısa yol problemi tanıtılmış ve ayrıca çizge parçalama için Kernighan Lin algoritması ele alınmıştır. Asıl amaç olarak, en kısa yol problemi için çizgeyi Kernighan Lin algoritması kurallarına göre işlemcilere ayıran ve böylelikle problem için çizgeyi başlangıç ve bitiş noktalarını ele alan bir zincir çizge formuna dönüştürerek en kısa yolu bulan bir yaklaşım ortaya konulmuştur. Her parça içinde amaç düğümler arasındaki en kısa rotayı bulan parça içi en kısa yollar hesaplanmaktadır.

**ANAHTAR KELİMELER**
Çizge, Çizge Parçalama, En Kısa Yol Problemi



[1] Bu çalışma yazarın D.E.Ü. S.B.E. Ekonometri B.D.'da tamamlanmış yüksek lisans tezinden alınmıştır. Ayrıca bir bölümü YA/EM-2000'de sunulmuştur.

# AN APPROACH FOR THE SHORTEST PATH PROBLEM: DEALING WITH GRAPH PARTITIONING

**Mustafa Kemal BEŞER**
Eskişehir Osmangazi University, Faculty of Economic and Administrative Sciences



## Abstract
As a starting point of this study, basic concepts of graph theory, shortest path problem, and Kernighan Lin algorithm are presented for graph partitioning problem. Main aim is to present an approach which solves the shortest path problem by using the graph partitioning technique in regard with the rules of Kernighan-Lin algorithm; therefore it becomes to a chain graph dealing with the starting and the target nodes of the problem. The shortest paths are calculated in order to provide the shortest route between the objective nodes in all the processors.

## KEY WORDS
Graph, Graph Partitioning, Shortest Path Problem

## 1. GİRİŞ

Geometride (x,y) ikilileri ile belirlenen noktalar bir koordinat sisteminde işaretlenebilir. (x,y) noktalarının değişmesi sonucu ortaya çıkan ya bir doğru ya da bir eğridir. Kısaca çizge (graph) adı verilen bu doğru ya da eğriler, çizge teorisinin temel öğeleridir. Bir doğrusal programlama probleminde, problemin amaç fonksiyonunu en elverişli kılan çözüm kümesinin bulunması ile modelin kaynaklarından en iyi bir düzeyde yararlanmanın sağlanması birbirine eşdeğerdir. Bu yönüyle problemin çözümünde ve geliştirilen kriterlerde çizgeler kuramı, ağ akışları, doğrusal programlama, dağıtım problemi ve matrisler iç içe görülmektedir. Uygulama alanı, sayılamayacak kadar çok olan çizgeler kuramı, bugün, kendine özgü tanım ve teoremleriyle ayrı bir matematik dalı olarak uygulamacıların hizmetindedir.

Ekonomi, işletme, istatistik, kimya, bilgisayar bilimleri, mühendislik, biyoloji gibi pek çok bilim alanındaki çeşitli problemlerde; problemin elemanları ve bunların birbirleriyle olan ilişkileri bir çizgeyle ifade edildiğinde; çözüm çizge teorisi yardımıyla çoğu kez daha kolay bulunabilmektedir.

Bir çizge parçalama probleminde parçalarına ayrılmış gruplar arasındaki, yani işlemciler arasındaki iletişim zamanının az olmasının yanında hesaplama yükünün de dengelenmesi gerekir. Algoritmalar, bir komşu düğümün diğer bir işlemciye aktarılmasıyla çalışır. "Kesim büyüklüğü" işlemciler arasındaki atlama ayrıtlarının ağırlıkları toplamıdır. Bu işlem komşu düğümün diğer işlemciye geçmesi ile kesim büyüklüğünün ne kadar değişeceğinin tespiti içindir. Bu tür ardışık algoritmalarda düğümler işlemciler arasında devamlı yer değiştirdikleri için, çizge, bir hesaplamadan diğerine geçerken artan bir ivme ile değişir. Bu hesaplamalar sırasında işlemcilere düğümler ve dolayısıyla kirişler eklenir ya da çıkarılır. Bu tekrarlı bir işlem izleyen algoritmalara literatürde "dinamik algoritmalar" denir. Bütün bu ardışık algoritmalar istedikleri girdi tipine göre farklılık gösterirler. (Fjalström, 1998)

Konu ile ilgili yazında pek çok öncü çalışma vardır. Örneğin, Fiduccia ve Mattheyses (1982), bir iterasyonun $O(|E|)$ operasyon zamanı aldığı, Kernighan ve Lin'in (KL) algoritmasından esinlenmiş, Fiduccia ve Mattheyses (FM) algoritmasını hazırlamışlardır. KL metodu çizge parçalama bölümünde tam olarak ele alınacaktır. FM metodu da KL metodunda olduğu gibi bir iterasyon sırasında kesim büyüklüğündeki kazancın pozitif olduğu en iyi ikiye parçalama işlemini gerçekleştirir. Ancak, düğüm çiftleri seçmektense FM metodu düğümleri tek olarak seçer. Bu, iterasyonun her adımında, kesim büyüklüğünü en fazla azaltan işaretlenmemiş bir düğümün $N_1$ ve $N_2$'den sırayla seçilmesi içindir. FM

metodu, keyfi parça sayısı ve de ağırlıklı düğümlerle çalışmak üzere geliştirilmiştir.

Rolland, Pirkul, ve Glover (1996), çizgeyi ikiye parçalamada bu metodu başarılı bir şekilde kullanmışlardır. İlk olarak dengelenmiş bir şekilde belirlenmiş iki işlemciden başlanır ve daha iyileri iterasyonlu bir şekilde araştırılır. Her bir iterasyon esnasında bir düğüm seçilir ve diğer işlemciye geçirilir. Geçişten sonra, düğüm "tabu listesi"ne alınır. Geçiş sonunda kesim kirişleri ağırlığı azalıyorsa, bu parçalama o esnadaki en iyisi olarak kayda alınır. Belli sayılardaki geçişlerden sonra daha iyi bir kesim kirişleri ağırlığı toplamı bulunamazsa bu parçalama o esnadaki en iyisi olarak kayda alınır. Belli sayıdaki geçişlerden sonra daha iyi parçalama bulunamaz ise algoritma "dengesizlik faktörü"nü arttırır. Bu faktör iki parça arasındaki esas farkı limitler. Başlangıç olarak bu faktör sıfır olarak kurulur. Düğümleri tabu listesine sokan geçişler o anki en iyi parçalamada bir gelişme sağlarsa, geçişine izin verilir. Algoritma bütün izin verilmiş geçişler için, kesim büyüklüğünde en büyük azalışı sağlayan bu yer değiştirmeleri sağlar. Rolland, Pirkul, ve Glover deneysel olarak kendi algoritmalarını Kernighan-Lin ile karşılaştırmış ve de zaman ve çözüm kalitesi açısından daha iyi sonuç verdiğini bulmuştur.

Bui ve Moon (1996), genetik bir algoritma geliştirmişlerdir. Bir genetik algoritma (GA) popülasyon adı verilen bir kromozomlar (çözümler) kümesi ile başlar. Bu popülasyon bir durma şartı sağlanana kadar birçok jenerasyonlar meydana getirir. Yeni bir jenerasyon mevcut popülasyondan bir ya da daha çok kromozom çiftinin seçilmesiyle elde edilir. Bu seçim olasılıksal seçim şeması tabanlıdır. Çaprazlama operasyonu herbir çiftin evlat meydana getirmesi yani soyunu devam ettirmesi için birleştirilmesidir. Sonuç olarak bir yerleştirme şeması kullanılır. Bu, hangi popülasyon üyeleriyle hangi evlatların yer değiştirdiğini gösteren şemadır.

Bunun yanında pek çok araştırmacı çizge parçalama için genetik algoritmalar geliştirmişlerdir. Bunlar arasında Berger ve Bokhari (1987), bir geometrik algoritmanın en temel örneklerinden biri olan "recursive coordinate bisection" (RSB) algoritmasını tasarlamışlardır
Miller, Teng, Thurston ve Vavasis (1993), d-boyutlu bir çizgeyi (düğümleri d-boyutlu bir uzaya yerleştirilmiş bir çizgeyi) ikiye parçalayan bir algoritma tasarlamışlardır.

Pothen, Simon, ve Liu'nun (1990), geliştirdikleri "Recursive Spectral Bisection" (RSB) metodu, çizgedeki Laplace matrisinin ikinci en küçük özdeğerine uyan özvektörü kullanır (Laplace matrisi L=D-A, D düğüm

derecelerini gösteren diagonal matris, A ise bitişiklik matrisidir). "Fiedler vektörü" adı verilen bu özvektör çizge hakkında önemli bilgileri barındırır. Fiedler vektörünün koordinatları arasındaki fark, ilgili düğümler arasındaki mesafe hakkında bilgi sağlar. Böylece RSB metodu düğümlerin Fiedler vektörü koordinatlarının sınırları bakımından çizgeyi ikiye böler. RSB çizgeyi dörde ya da sekize parçalamayı ve de düğüm ve kiriş ağırlıklarını da gözönüne alarak çözüm verebilecek şekilde tasarlanmıştır.

Araştırmacılar çizge parçalama için paralel algoritmalar da geliştirmeye başlamışlardır. Bunun sebebi bazı ardışık algoritmalar yüksek kaliteli parçalar verirler fakat yavaştırlar. Böyle bir algoritmayı paralelize etmek hesaplama hızını arttırabilir. Ancak, çizge çok büyükse veya bir paralel algoritma tarafından meydana getirilmişse ardışık parçalama algoritması etkisiz olacak ve uygulamak mümkün olmayacaktır. (Fjallström, 1998)
Gilbert ve Zmijevski (1987), bir çizgeyi ikiye parçalamak için en eski paralel algoritmalardan birini hazırlamışlardır. Bu algoritma KL algoritması temellidir ve de herbir iterasyonuna $\{N_1,N_2\}$ parçaları girdi teşkil eden birtakım iterasyonlardan oluşur. Bu ikiye parçalama işlemcileri $\{P_1,P_2\}$ şeklinde dengelenmiş bir parçalamadır. Eğer $v \in N_2$ ise v düğümünün bütün komşu düğümleri $P_1$'in dâhil olduğu işlemcide depolanır.

Walshaw, Cross ve Everett (1997), çizgede tekrarlı parçalama yapabilen iterasyonlu bir algoritma vermişlerdir. (JOSTLE-D yazılımı). Bu algoritma ilk olarak komşu parçalar arasında ne kadar ağırlığın transfer yapılacağını belirler. Daha sonra işlemciler arasında iterasyonlu bir düğüm alış-veriş safhasına geçer.

Schloegel, Karypis ve Kumar (1997), çokaşama yaklaşımı temelli ardışık ve paralel tekrarlı parçalamalı algoritmalar sunmuşlardır.

Bu çalışmada önce çizge parçalaması ile ilgili literatür gözden geçirilmiş, ardından çizgeler kuramının uygulama alanlarından biri olan en kısa yol problemi ele alınmıştır. Sonrasında sonuca çizge parçalama metodu kullanarak ulaşan bir en kısa yol problemi algoritması sunulmuştur. Bu algoritmada, zincir çizge yapısı elde edilebildiği durumlarda, bu yapının parçalanmış gruplar arasında geçiş yapılırken sadece ardışık gruplar arası geçişe izin vermesi sayesinde, sadece atlama düğümleri işleme tabi tutulacaktır. Bu algoritma işleme tabi tutulan daha az düğüm sayısından ötürü zaman ve işlem sayısında büyük oranda tasarruf sağlamaktadır.

## 2. ÇİZGE TEORİSİ

N ile E ayrık iki küme ($N \neq \phi$) olmak üzere, bir G çizgesi (N,E) ikilisinden oluşur ve G=(N,E) yazılır. N'nin $v_i$ , (i=1,2,3,...,n) elemanlarına da G'nin düğümleri ve E'nin $a_k$, (k=1,2,3,...,m) elemanlarına da G'nin kirişleri denir. Herbir $a_k$ kirişini iki $v_i$ ve $v_j$ düğümlerine eşleyen bir g bağıntısı vardır ve bu $g(a_k)=(v_i,v_j)$ biçiminde gösterilir. Buradaki $(v_i,v_j)$ bir sıralı ikili ise $a_k$ bir yönlü kiriş, sıralı ikili değil ise $a_k$ bir yönsüz kiriş adını alır. Tüm kirişleri yönlü olan bir çizgeye yönlendirilmiş çizge ve tüm kirişleri yönsüz olan bir çizgeye yönlendirilmemiş çizge denir.

Bir G çizgesinin bir $a_k$ kirişi için $g(a_k)=(v_i,v_j)$ ise $v_i$ ve $v_j$ düğümlerine $a_k$ kirişinin uç düğümleri ya da son noktaları denir. Bir kirişin uç düğümleri durumunda olan iki düğüme bağlantılı düğümler, birer uç düğümleri ortak olan kirişlere bağlantılı kirişler adı verilir. İki uç düğümleri çakışık olan bir kirişe bukle denir.

En kısa yol problemi aslında bir doğrusal programlama problemidir. Bu şöyle ispat edilebilir; $v_1$'den $v_n$'e giden en kısa yol belirlenmek isteniyorsa ve eğer $v_i$ ve $v_j$ noktaları bir kirişle birleştirilmiş ise $q_{ij}$ uzunluklu yollar takımı oluşturulur. Eğer $v_i$ ve $v_j$ noktaları bir kirişle birleştirilmemiş ise $q_{ij}=+\infty$ yazılabilir. Buklesiz herhangi bir ağda $v_1$'den $v_n$'ye giden yollardan herbiri $\mu(v_1, v_{i_1},..., v_{i_k}, v_n)$ biçiminde ifade edilebilir. Eğer $(v_i,v_j)$, $\mu$'ye ait ise, $n_2$ sayıda olan ve $0 \leq x_{ij} \leq 1$ (i,j=1,2,..,n) ile tanımlanan $x_{ij}$ sayılarının birleştirilen takımı bire eşitlenir ve aksi halde yani $(v_i,v_j)$ $\mu$'ye ait değilse sıfıra eşitlenir. Bu halde $v_1$ ve $v_n$ arasındaki en kısa yolu bulma problemi;

a) $0 \leq x_{ij} \leq 1$ (i,j=1,2,...,n),

b) $\sum_{j=1}^{n} (x_{ij} - x_{ji}) = 0$ ;(i≠1, i≠n),

c) $\sum_{j=1}^{n} (x_{1j} - x_{j1}) = 1$

d) $\sum_{j=1}^{n} (x_{nj} - x_{jn}) = -1$

koşulları altında, $v_1$'den $v_n$'ye giden yolun uzunluğu olan, $z=\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_{ij}$ doğrusal ifadesini minimum yapan tam sayılı çözümünü bulmaktır. Yani $\|x_{ij}\|_{n.n}$'yi belirtmektir.

$v_1$ ve $v_n$ arasındaki herhangi bir $\mu(v_1, v_{i_1}, ..., v_{i_k}, v_n)$ yolunun $x_{ij}$'ye (i,j=1,2,...,n) karşı gelen belirli kümesi yukarıdaki koşullarını sağlar. Bunun ispatı şöyledir:

($v_i$,$v_j$) düğümleri μ yoluna ait ise $x_{ij}=1$ ve ait değilse $x_{ij}=0$ alınması gereken bir büyüklük olarak tanımlandığından $0\leq x_{ij}\leq 1$ (i,j=1,2,...,n) olup a koşulu sağlanmaktadır.

"b" koşulu, $\sum_{j=1}^{n}(x_{ij}-x_{ji})=0$ (i≠1, i≠n) idi. Bu koşul, $(x_{i1}-x_{1i})+(x_{i2}-x_{2i})+...+(x_{in}-x_{ni})=0$ biçiminde (i=2,3,...,n-1) için ayrı ayrı yazılarak (n-2) tane denklem oluşturur. Bu denklemlerden herbiri μ yolunun ilk ve son noktaları hariç $v_1,v_2,..,v_{n-1},v_n$ takımının sabit bir $v_i$ düğümü için yazılmış ve yine bu denklemlerden herbiri $x_{ij}$ ve $x_{ji}$'leri kapsayan, birinin başlangıcı ve diğerinin sonu $v_i$'de bulunan iki kirişe karşılık gelmektedir.

Eğer $v_i$ noktası μ yoluna ait değil ise, buradan ($v_i$,$v_j$) ve ($v_j$,$v_i$) kirişlerinden hiçbirisi μ'ye ait olamaz ve bu nedenle $x_{ij}=x_{ji}=0$ (j=1,2,...,n) olur. Buradan da $\sum_{j=1}^{n}(x_{ij}-x_{ji})=0$ yazılır.

"c" ve "d" koşulları, μ yolu $v_1$ uç noktalı tüm kirişlerinden sadece birini kapsadığı (yani ilk kiriş μ yoluna ait) ve aynı biçimde yine μ yolu $v_n$ uç noktalı tüm kirişlerinden sadece birini kapsadığı (yani son kiriş μ yoluna ait olduğu) dikkate alınırsa "b" koşuluna benzer düşüncelerin bir sonucu olduğu görülür.

Burada doğrusal programlama probleminin her bir tamsayılı çözümünün belirlediği $x_{1j_1}=x_{j_1j_2}=...=x_{j_{n-1}n}=1$ biçimindeki bir sayı kümesinin bir μ yoluna karşılık olduğunu göstermek gerekir. $\sum_{j=1}^{n}x_{1j}=1+\sum_{j=1}^{n}x_{j1}$ denkleminden de anlaşılacağı gibi en az bir tane $x_{1j}=1$ vardır (Bu ise ilk

kirişin μ yoluna ait bulunduğunun bir ifadesidir). Benzer olarak, $\sum_{j=1}^{n} x_{j_1 j} = \sum_{j=1}^{n} x_{j j_1} \geq 1$ ifadesinden de anlaşılacağı gibi hiç olmazsa $x_{j_1 j_2} = 1$ olan en az bir eleman vardır. Bu şekilde ilerleyerek bir dizi oluşturulur. Bu dizi ise ancak $v_n$ uç noktasında sonuçlandırılır. Ayrıca herhangi bir $v_s$ düğümünde $\sum_{j=1}^{n} (x_{sj} - x_{js}) = 0$ eşitliği elde edilir.

Görüldüğü gibi en kısa yol problemi, bir doğrusal programlama problemine dönüştürülebilmektedir. Ancak bu problemin simpleks yöntemle çözümü, koşulların özelliklerine bağlı olarak, çok uzun işlem zamanı alabilir.

Pratikte bir çizgede istenen bir başka çözüm de, en kısa ikinci, üçüncü yollardır. Bunun sebebi, probleme başka kriterler de katıldığında, bu en kısa yollar arasındaki, bu kriter çerçevesinde en elverişli yolun bulunmak istenmesidir. En kısa yol problemlerini çözen, Ford, Floyd, Bellman-Kalaba, Dijkstra gibi algoritmalar bilinmektedir.

Başlangıç "s" ile "t" düğümleri olarak tanımlanan düğümler arasındaki en kısa yol problemi için en etkili algoritmalardan biri Djkstra algoritmasıdır. Metot düğümlere geçici etiketler vererek çalışır. Bir düğümdeki etiket, s'den o düğüme olan yolun üst sınırıdır. Bu etiketler sürekli olarak iterasyonlu bir prosedür izleyerek üretilirler. Her iterasyonda bir geçici etiket sabit etikete dönüşür ve bu etiket artık üst sınırı değil, s'den o düğüme olan en kısa yolu verir. Sabitlenen etikete "+" üst indisi verilir.

## 3. ÇİZGE PARÇALAMA İÇİN ALGORİTMALAR

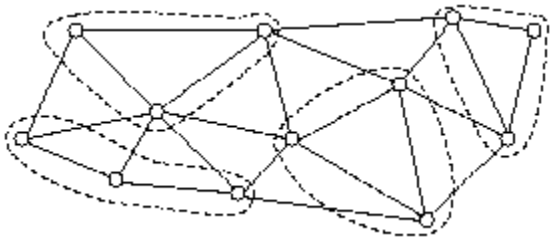
Verilen bir G=(N,E) çizgesinde (N ağırlıklı düğümlerin kümesi, ve E de ağırlıklandırılmış kiriş kümesi olmak üzere) N'nin p alt kümesi $N_1$, $N_2$,...,$N_p$ için

$U_{i=1}^{p} N_i = N$ ve $N_i \cap N_j = \phi$ i≠j ise. Bu bir düğümün birden fazla grup (işlemci) içersinde yer almadığı anlamına gelir.

$W(i) \approx W/p$ i=1,2,..,p $W_i$ ve W sırasıyla $N_i$ ve N'nin düğüm ağırlıkları toplamıdır. Bu, sistemin yükünü işlemciler arasında yaymak ve dengelemek anlamına gelir.

Bu alt kümeler arasındaki geçişi sağlayan kirişlerin (kesim kirişlerinin büyüklüğü) ağırlıkları toplamı minimum olmalıdır. Bu, işlemciler arasındaki iletişimi minimum tutmak anlamına gelir.

Herhangi bir $\{N_i \subseteq N : 1 \leq i \leq p\}$ kümesi, birinci şartı sağlarsa, bu "p-yollu parçalama" olarak adlandırılır. Her bir $N_i$ çizgenin bir parçasıdır. Bir bisection (ikiye parçalama) 2-yollu parçalamadır. İkinci durumu sağlayan parçalamaya da "dengelenmiş parçalama" denir. Şekil-1'de 4-yollu dengelenmiş parçalanma çizge üzerinde gösterilmiştir.



**Şekil-1** (4-yollu Dengelenmiş Parçalama)

G=(N,E) çizgesini ikiye parçalamak iki yolla yapılabilir. Birincisi E'nin en küçük altkümesi olan $E_s$'yi E'den ayırarak, G çizgesini, $N_1 \cup N_2 = N$ olmak üzere her biri $N_1$ ve $N_2$ düğümlerine sahip $G_1$ ve $G_2$ alt çizgesine bölmektir. $N_1$ ve $N_2$ eşit büyüklüktedir. $E_s$ içindeki kirişler $N_1$ içindeki düğümleri $N_2$ içindeki düğümlere bağlarlar. $E_s$ kaldırılırsa çizge ikiye parçalanır. $N_1$ ve $N_2$ arasındaki tüm bağlantılar kopar. $E_s$ kiriş ayrıştırıcısıdır.

Bir çizgeyi parçalamanın diğer bir yolu da $N_s$ kümesini yani düğüm ayrıştırıcılarını bulmaktır. N'nin bir altkümesi olan $N_s$'nin ve tüm bağlı kirişlerinin çizgeden kaldırılması çizgeyi birbirine bağlantısız iki $G_1$ ve $G_2$ alt çizgesine böler. Diğer bir deyişle $N=N_1 \cup Ns \cup N_2$'dir. $N_1$ ve $N_2$ yine eşit büyüklüktedir ve bunları hiçbir kiriş birbirine bağlamaz. Kısaca çizge parçalamak için çizgeden, çizgeyi parçalamak için düğüm grupları ya da kirişler çıkartılır. Eğer kiriş çıkartılarak çizge parçalarına bölünüyorsa bu kirişler kesim büyüklüğü olarak adlandırılır. (Berkeley(1996))

## 4. KERNIGHAN-LIN ALGORİTMASI

Kernighan-Lin ilk çizge parçalama metotlarından birini tasarlamışlardır ve birçok bölgesel gelişim metodu Kernighan-Lin metodunun bir varyasyonudur. Öncelikle çizge rastsal olarak eşit iki parçaya bölünür. Bu iki parça algoritmanın girdileridir. Kernighan-Lin kesim büyüklüğünü azaltmaya yönelik, farklı parçalarda bulunan düğüm çiftlerini ardışık olarak yer değiştirir.

$\{N_1,N_2\}$ G=(N,E) çizgesinin iki parçası olsun (burada düğüm ağırlıklara 1 olarak varsayılmıştır). Her bir v∈N için şu tanımlamalar yapılır;

$$\text{int(v)}= \sum_{(v,u)\in E, P(v)=P(u)} w(v,u) \quad ; \quad \text{ext(v)} = \sum_{(v,u)\in E, P(v)\neq P(u)} w(v,u)$$

int(v), v düğümünü kendi işlemcisi içindeki düğümlere bağlayan kirişlerin ağırlıkları toplamıdır. ext(v) ise, v düğümünü diğer işlemcilere bağlayan kirişlerin ağırlıkları toplamıdır. P(v), v düğümünün bulunduğu parçadır. w(v,u) ise(v,u) kirişinin ağırlığıdır. Çizgede toplam kesim büyüklüğü $\sum_{v\in N} ext(v)$'dir. v düğümünün bulunduğu parçadan diğer parçaya taşınmasıyla elde edilen kazanç g(v)=ext(v)–int(v) formülüyle hesaplanır. Böylece g(v)>0 olduğunda, v'nin yer değiştirilmesiyle kesim büyüklüğü g(v) kadar azalır. $v_1\in N_1$ ve $v_2\in N_2$ için $g(v_1,v_2)$ $v_1$ ve $v_2$ düğümlerinin $N_1$ ve $N_2$ arasında yer değiştirilmesinden elde edilen kazanç olsun. Bu kazanç;

$$\text{g(v1, v2)} = \begin{cases} g(v_1)+g(v_2)-2w(v_1,v_2) & eger \ (v_1,v_2)\in E \\ g(v_1)+g(v_2) & d.d. \end{cases}$$

ile formülleştirilir. Bu, $v_1$ ve $v_2$ düğümleri yer değiştirilirse, kesim büyüklüğünde elde edilecek azalışı (kazancı) verir. Herhangi bir iterasyonda, bu kazanç değerini yani kesim büyüklüğündeki azalışı en büyükleyen düğüm çifti yer değiştirilir.

Kernighan-Lin algoritmasının bir iterasyonu şöyledir. İterasyona girdi dengelenmiş (düğüm ağırlıkları eşit) olan $\{N_1,N_2\}$parçalarıdır. İlk olarak, düğümlerin tekrar adlandırılabilmesi için bütün düğümlerin işaretleri kaldırılır. Sonra aşağıdaki prosedür n kere tekrarlanır. $(n=\min(|N_1|,|N_2|))$.

$g(v_1,v_2)$'yi pozitif yapan (fakat ister istemez pozitif olmayabilir) $v_1\in N_1$ ve $v_2\in N_2$ işaretsiz düğüm çifti bulunur. $v_1$ ve $v_2$ işaretlenir ve geriye kalan bütün işaretsiz düğümlerin g değerlerini $v_1$ ve $v_2$'yi yer değiştirilmiş gibi güncellenir. Sadece $v_1$ ve $v_2$'nin komşularının g değerlerinin güncellenmesi yeterlidir.

Bu işlemlerden sonra sıraya sokulmuş düğüm çiftleri listesi oluşur. $(v_1^i,v_2^i)$, i=1,2,...,n. $\left[\sum_{i=1}^{j} g(v_1^i,v_2^i)\right]$'yi maksimum yapan j indeksi bulunur. Eğer toplam pozitif ise ilk j adet düğüm çifti yer değiştirilir ve KL algoritmasının başka bir iterasyonuyla devam edilir. Aksi halde algoritma durdurulur. Kernighan-Lin metodunun tek bir iterasyonu en fazla $O(|N^3|)$ kadar zamana ihtiyaç duyar.

## 5. EN KISA YOL PROBLEMİNİN ÇİZGE PARÇALANIŞI İLE ÇÖZÜMÜ

Tanım-1: G=(N,E) birleştirilmiş, katlı kirişsiz, buklesiz ve yönlendirilmiş bir çizge olsun. Böyle bir G çizgesinin bitişiklik matrisi olan A matrisi aşağıdaki biçimde tanımlanır (Buckley, Harary (1990)).

$$a_{ij} = \begin{cases} 1 & i \ tepesi \ j \ tepesine \ bir \ kirisle \ bitisik \\ 0 & i \ tepesi \ j \ tepesine \ bir \ kirisle \ bitisik \ degil \end{cases}$$

Tanım-2: G bir basit yönlendirilmiş çizge olsun. G'nin A bitişiklik matrisi aşağıdaki biçimde yazılabiliyorsa G çizgesine zincir çizge denir. Burada $A_1,A_2,...,A_m$ n boyutlu kare alt matrisler olup sırasıyla G'nin $G_1=(N_1,E_1)$, $G_2=(N_2,E_2),...,G_m=(N_m,E_m)$ alt çizgelerinin bitişiklik matrisleridir. Eğer G çizgesinin kirişleri üzerinde ağırlıklar (maliyetler) işaretlenmişse G'nin C maliyetler matrisi ve A matrisi:

$$A = \begin{bmatrix} A_1 & B_1 & 0 & . & . & 0 \\ 0 & A_2 & B_2 & 0 & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & . & . & . & B_{m-1} \\ 0 & 0 & . & . & . & A_m \end{bmatrix} \qquad C = \begin{bmatrix} D_1 & E_1 & . & . & . \\ & D_2 & E_2 & . & . \\ . & . & . & . & . \\ . & . & . & . & E_{m-1} \\ . & . & . & . & D_m \end{bmatrix}$$

biçiminde gösterilebilir (Dündar, Beşer, Dündar (2000)). Burada $D_1$, $D_2,...,D_m$ sırasıyla $G_1$, $G_2,...,G_m$ alt çizgelerinin ağırlıklar matrislerini ve $E_1$, $E_2,...,E_{m-1}$ bu alt çizgelerin geçiş kirişlerinin ağırlıklarını içerir.

Verilen ağ'ın modeli olan çizge; KL algoritması ile k adet parçaya ayrılarak bir zincir çizge haline getirildikten sonra; $G_1$ çizgesindeki bir $v_1$ düğümünden başlayarak sırasıyla $G_2$, $G_3,..,G_{k-1}$ alt çizgelerinden geçip $G_k$ çizgesinin bir $v_2$ tepesinde biten en kısa yolu bulma bu çalışmanın temelidir.

**Adım-1:** $G_1$ çizgesinin verilen bir $v_1$ düğümü ile başlayan ve $G_1$ ile $G_2$ çizgesinin $(y_i,z_i)$ kesim kirişlerinin $(y_i)$ düğümleri ile biten işlemci içi tüm $(v_1,y_i)$ en kısa yolları Dijkstra algoritması ile hesaplandıktan sonra bütün bu yolları $G_2$ işlemcisine bağlayan $(y_i,z_i)$ kesim kirişleri her bir bulunan $(v_1,y_i)$ yoluna eklenerek tüm $(v_1,z_i)$ en kısa yolları bulunur.

**Adım-2:** $G_2$ çizgesinde $z_i$ düğümleri ile başlayıp $G_2$ ile $G_3$ çizgesinin $(t_i,p_i)$ kesim kirişlerinden $t_i$ düğümleri ile biten işlemci içi tüm $(z_i,t_i)$ en kısa yolları Dijkstra algoritması ile hesaplandıktan sonra bütün bu yolları $G_3$ işlemcisine bağlayan $(t_i,p_i)$ kesim kirişleri her bir bulunan $(z_i,t_i)$ yoluna eklenerek $(z_i,p_i)$ en kısa yolları bulunur.

**Adım-k:** $G_{k-1}$ ile $G_k$ işlemcilerinin kesim ayrıtlarının son düğümleri olan $q_i$ düğümleri ile $v_2$ arasındaki tüm $(q_i,v_2)$ yolları bulunur.

**Adım-k+1:** $v_1$'den $v_2$'e tüm bağlantılı yollar arasındaki $\min[(v_1,y_i)+(y_i,z_i)+(z_i,t_i)+(t_i,p_i)+(p_i,..)+(...,q_i)+(q_i,v_2)]$ $(1\leq i\leq n/k)$ yolu hesaplanır. Bu yol bütün işlemcilerden sırasıyla geçmek şartı ile en kısa yolu verir. Geliştirilen algoritma için bir örnek olarak aşağıdaki çizge ele alınırsa, çizge parçalama yöntemi ile en kısa yol bu algoritma ile şu şekilde hesaplanabilecektir. Bu hesaplamalar Borland Delphi yardımıyla yazılan program ile elde edilmiştir.

| | v1 | v2 | v3 | v4 | v5 | v6 | v7 | v8 | v9 | v10 | v11 | v12 | v13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| v1 | x | 10 | | | | | 3 | 6 | 13 | | | | |
| v2 | 10 | X | 18 | | | | | | | | | | |
| v3 | | 18 | X | 25 | | 25 | | | | | | | |
| v4 | | | 25 | X | 5 | 16 | | | | | | | |
| v5 | | | | 5 | X | 10 | 5 | | | | | | 8 |
| v6 | | | 25 | 16 | 10 | X | 14 | 15 | | | | 20 | |
| v7 | 3 | | | | 5 | 14 | X | | 24 | 13 | | | |
| v8 | 6 | | | | | 15 | | X | | | | | |
| v9 | 13 | | | | | | 24 | | X | | | | |
| v10 | | | | | | | 13 | | | X | 12 | | |
| v11 | | | | | | | | | | 12 | X | 15 | |
| v12 | | | | | | 20 | | | | | 15 | X | 10 |
| v13 | | | | | 8 | | | | | | | 10 | X |

Verilen bu çizgede $v_2$ düğümü ile $v_{13}$ düğümü arasındaki en kısa yolu bulma problemini ele alalım. Bu çizge önce Kernighan-Lin algoritmasıyla 4 parçaya bölünmüş, ardından geliştirdiğimiz algoritma bu çizgeye uygulanmıştır. Çizge 4 parçaya bölündükten sonra ağırlıklı bitişiklik matrisi aşağıdaki şekilde oluşur.

| | v4 | v2 | v3 | v8 | v10 | v6 | v7 | v1 | v9 | v5 | v11 | v12 | v13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| v4 | X | | 25 | | | 16 | | | | 5 | | | |
| v2 | | X | 18 | | | | | 10 | | | | | |
| v3 | 25 | 18 | X | | | 25 | | | | | | | |
| v8 | | | | X | | 15 | | 6 | | | | | |
| v10 | | | | | X | | 13 | | | | 12 | | |
| v6 | 16 | | 25 | 15 | | X | 14 | | | 10 | | 20 | |
| v7 | | | | | 13 | 14 | X | 3 | 24 | 5 | | | |
| v1 | | 10 | | 6 | | | 3 | X | 13 | | | | |
| v9 | | | | | | | 24 | 13 | X | | | | |
| v5 | 5 | | | | | 10 | 5 | | | X | | | 8 |
| v11 | | | | | 12 | | | | | | X | 15 | |
| v12 | | | | | | 20 | | | | | 15 | X | 10 |
| v13 | | | | | | | | | | 8 | | 10 | X |

## 6. SONUÇ

Çizge Kuramı kendine özgü tanımları kullanarak fen bilimleri ve sosyal bilimlerin pek çok problemlerini çözmede yaygın olarak kullanılmaktadır. Problemlerin çizgelerle sembolize edilmeleri ile problem daha sade bir görünüme kavuşur, böylece çözüme daha kolay ulaşılabilir. Önceki çalışmalarda; diferansiyel denklemlerin sonlu farklarla çözümleri gibi bazı, kare matrislerle çözülen problemlerin çözümlerinde sparse matrislerle karşılaşılmakta; bu matrislerle yapılacak işlemlerin süresini kısaltmak amacıyla matris yoğun alt matrislerine parçalanmaktadır.

Bu çalışmada, en kısa yol probleminin bağlantı matrisinin sparse olduğu durumda - ki bu zincir çizgelerde karşımıza çıkmaktadır, matris, yoğun alt matrislerine parçalanmış ve en kısa yol probleminin çözümü için bir algoritma üretilmiştir. Bir problemin çözümünde kullanılan algoritmanın karmaşıklığı, onun, uygulanması mümkün olan maksimum adım sayısıdır. Bu, işlem süresi ile doğru orantılıdır.

Çizgedeki düğüm sayısının büyük olduğu durumlarda (N>1000), çizgeye Djkstra Algoritması uygulandığında, en kısa yol probleminin çözümü için yapılabilecek maksimum işlem sayısı N(N-1)/2 olup, karmaşıklığı $N^2$ mertebesindedir. Kernighan-Lin Algoritmasının karmaşıklığının $N^3$ olduğu kaynaklardan bilinmektedir. Bu çalışmada önerilen algoritma $N^5$ karmaşıklıkta polinomsal bir algoritmadır. Düğüm ve parça sayısının büyük olduğu durumlarda çizge kolaylıkla zincir çizge formuna getirilebilmekte ve işlem sayısı önerilen algoritma yardımıyla azalmaktadır.

**KAYNAKLAR**

**Berger, M.J., Bokhari, S.H. (1987),** A Partitioning Strategy for Non-Uniform Problems across Multiprocessors, IEEE Transactions on Computers, C-36:570-580.

**Berkeley, C.S. (1996),** "Lectures Graph Partitioning I,II,." HTTP, CS267, Ders Notları

**Buckley, F., Harary, F. (1990),** *Distance in Graphs*, California: Addison-Wesley Pub.

**Bui, T.N., Moon, B.R. (1996),** Genetic algorithm and graph partitioning, IEEE Transactions on Computers, 45(7):841-855.

**Dündar, S., Beşer, M.K., Dündar, P. (2000),** En Kısa Yol Probleminin Çizge Parçalanışı ile Çözümü, Yöneylem Araştırması ve Endüstri Mühendisliği XXI. Ulusal Kongresi Bildiriler Kitabı, 118–121

**Fjalström, P. (1998),** *Algorithms for Graph Partitioning*, Linköping University Electronic Press

**Fiduccia, C., Mattheyses, R. A (1982),** Linear time heuristic for improving network partitions, 19th IEEE Design Automation Conference

**Gilbert, J.R., Zmijewski, E. (1987),** A parallel graph partitioning algorithm for a message-passing multiprocessor". *International J. of Parallel Programming* 16(1), 427-449.

**Miller, G.L., Teng, S.H., Thurston, W., Vavasis, S.A. (1993),** *Automatic mesh partitioning*, In A. George, .R. Gilbert, and J.W.H. Liu, editors, Graph Theory and Sparse Matrix Computation, volume 56 of The IMA Volumes in Mathematics and its Applications, 57-84. SpringerVerlag.

**Pothen, A. Simon, H.D., Liu, K.P. (1990),** Partitioning sparse matrices with eigenvectors of graphs, *SIAM J. on Matrix Analysis and Applications* 11(3), 430-452.

**Rolland, H. Pirkul, Glover, F. (1996),** Tabu search for graph partitioning, *Ann. Oper. Res.*, 63, 209-232.

**Schloegel, K., Karypis, G., & Kumar, V. (1997),** Parallel multilevel difusion schemes for repartitioning of adaptive meshes. Technical Report 97-014, University of Minnesota, Department of Computer Science.

**Walshaw, C., Cross, M., & Everett, M.G. (1997),** Parallel dynamic graph-partitioning for unstructured meshes. Technical Report 97/IM/20, University of Greenwich, Centre for Numerical Modelling and Process Analysis.