T.C.
DOKUZ EYLÜL ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
EKONOMETRİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

BİR ŞEBEKEDE EN KISA YOL PROBLEMİ İÇİN YENİ BİR YAKLAŞIM

M.Kemal BEŞER

Danışman
Yrd.Doç.Dr. Samim DÜNDAR

İZMİR
2000

Yüksek Lisans tezi olarak sunduğum "Bir Şebekede En Kısa Yol Problemi İçin Yeni Bir Yaklaşım" adlı çalışmanın, tarafımdan, bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurmaksızın yazıldığını ve yararlandığım eserlerin bibliyografyada gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve bunu onurumla doğrularım.

Tarih

25 / 08 / 2000

Adı ve Soyadı

M.Kemal Beşer

TUTANAK

Dokuz Eylül Üniversitesi Sosyal Bilimler Enstitüsü'nün 2.. /8... / 2000 tarih ve ....16.. sayılı toplantısında oluşturulan jürimiz tarafından, Lisansüstü Öğretim Yönetmeliğinin .....2.... maddesine göre Enstitümüz EKONOMETRİ Anabilim Dalı Yüksek Lisans öğrencisi M.Kemal BEŞER'in"Bir Şebekede En Kısa Yol Problemi İçin Yeni Bir Yaklaşım" konulu tezi incelenmiş ve aday ..... / ..... / 2000 tarihinde saat ....14.20.....' da jüri önünde tez savunmasına alınmıştır.

Adayın kişisel çalışmaya dayanan tezini savunmasından sonra 60 ..... dakikalık süre içinde gerek tez konusu gerekse tezin dayanağı olan anabilim dallarından jüri üyelerince sorulan sorulara verdiği cevaplar değerlendirilerek tezin

BAŞARILI olduğuna ☒ OY BİRLİĞİ ☐

DÜZELTME yapılmasına ☐ * OY ÇOKLUĞU ☐

RED edilmesine ☐ ** ile karar verilmiştir.

* Bu halde adaya 3 ay süre verilir.

** Bu halde adayın kaydı silinir.

BAŞKAN
Prof.Dr.Ali ŞEN

Üye
Yrd.Doç.Dr. Samim Dündar

Üye
Yrd. Doç. Dr. Ali Kemal ŞEHİRLİOĞLU

	Evet	Hayır
*** Tez, burs, ödül veya Teşvik prog.(Tüba, Fullbright,vb.) aday olabilir.	☐	☐
Tez, mutlaka basılmalıdır.	☐	☐
Tez, mevcut haliyle basılabilir.	☐	☐
Tez, gözden geçirildikten sonra basılabilir.	☐	☐
Tez, basımı gerekliliği yoktur.	☐	☐

YÜKSEKÖĞRETİM KURULU DÖKÜMANTASYON MERKEZİ

TEZ VERİ FORMU

Tez No: Konu Kodu: Üniv. Kodu:

Tezin Yazarının :

Soyadı : BEŞER Adı : M.Kemal

Tezin Türkçe Adı : Bir Şebekede En Kısa Yol Problemi İçin Yeni Bir Yaklaşım

Tezin Yabancı Dildeki Adı : A New Approach For The Shortest Path Problem In A Network

Üniversite : Dokuz Eylül Üni. Enstitü : Sosyal Bilimler Yıl: 2000

Tezin Türü :

Yüksek Lisans: ☐ Dili : Türkçe

Doktora: ☐ Sayfa Sayısı : 50

Tıpta Uzmanlık: ☐ Referans Sayısı : 26

Sanatta Yeterlilik: ☐

Tez Danışmanının:

Ünvanı : Yrd. Doç.Dr. Adı : Samim Soyadı : DÜNDAR

Türkçe Anahtar Kelimeler :	İngilizce Anahtar Kelimeler:
1.Çizge	1. Graph
2.Çizge Parçalanışı	2. Graph Partitioning
3.Gevşek Matris	3. Sparse Matrix
4.Yöneylem Araştırması Uygulamaları	4. Applications Of Operations Of Research
5.Matematiksel Programlama	5. Mathematical Programing

Tarih :25-08-2000

İmza :

ÖNSÖZ

Doğrusal Programlama, Dağıtım Problemi, Ağ Akışları ile iç içe olan Çizge Teorisi konusunda çalışma yapmaya beni yönlendiren, tüm çalışmalarımı titizlikle inceleyip, her konuda değerli görüşlerini aldığım Hocam, Sayın Yrd.Doç. Dr. Samim DÜNDAR'a ve değerli katkılarından dolayı Yrd.Doç.Dr. Pınar Dündar'a teşekkürlerimi iletmeyi bir borç bilirim.

M.Kemal BEŞER

Temmuz,2000,İzmir

ÖZET

Yöneylem araştırması ve endüstri mühendisliğinde bilinen bir problem olan En Kısa Yol Problemi, Çizge Kuramının da uygulamada en çok kullanılan problemlerinden biridir. En Kısa Yol Problemi, verilen iki nokta arasındaki yollar içinden uzunluğu en kısa yolu bulma ya da verilen bir düğümden başlayan ve diğer tüm düğümlere ulaşan yollar içinden en kısa uzunlukta olan yolun bulunması olarak tanımlanır. Çizgenin düğüm sayısının büyük olduğu durumlarda bu çizgeyi alt çizgelere ayırarak problemi çözme, bilinen bir yöntemdir. Verilen çizge, düğümleri sayısı hemen hemen eşit ve kesim kirişlerinin ağırlıkları toplamı minimum olacak biçimde parçalarına ayrılır.

Bu çalışmada ilk olarak çizge kuramının temel kavramları verilmiş, ardından en kısa yol problemi için iki adet algoritma birer uygulama ile sunulmuş, çizge parçalama için Kernighan Lin algoritması ve merkezleştirilmiş ağırlıklı ayrıt algoritmaları ele alınmıştır. Son olarak ise en kısa yol probleminin uygulanacağı çizge önce, Kernighan-Lin algoritması ile yukarıdaki kurallara uygun biçimde parçalarına ayrılmış, sonra yolun başlangıç düğümü ve son düğümü dikkate alınarak, bir zincir çizge haline getirilmiş, her parça içinde amaç düğümlerle ilişkiyi sağlayacak biçimde en kısa yollar hesaplanmış, ardından bunların geçiş kirişleri ile birleştirilmesi yöntemiyle amaç düğümler arasındaki en kısa yolu bulan bir algoritma geliştirilmiştir.

ABSTRACT

Shortest Path Problem, which is well known in Industrial Engineering and Operational Research, is one of the most encountered problems of Graph Theory in application. Shortest Path Problem is defined as finding the shortest way between two given nodes or finding the shortest path from which begins from a given node and arrives to all given nodes. If the node numbers of the graph is so much, then solving problem by separating this graph to processors is an appropriate method. The given graph is separated to parts as the processors are balanced and the cutsize would be minimum.

In this study, first of all basic concepts of Graph Theory is given, then two algorithms for the Shortest Path Problem are presented with an application and dealt with Kernighan Lin Algorithm to partition the graph. Finally, the graph which Shortest Path Problem would be applied on was partitioned into pieces in regard with the rules above with Kernighan-Lin Algorithm, then converted to a chain graph dealing with the starting node and target node of the problem. In all the processors, the shortest paths are calculated in order to provide relation with objective nodes and then an algorithm which finds the shortest path between objective nodes is developed by the method of uniting these with cutting edges.

BİR ŞEBEKEDE EN KISA YOL PROBLEMİ İÇİN YENİ BİR YAKLAŞIM

BİRİNCİ BÖLÜM

LİTERATÜR TARAMASI

İKİNCİ BÖLÜM

ÇİZGELER KURAMINDA ÖNEMLİ KAVRAMLAR

ÜÇÜNCÜ BÖLÜM

EN KISA YOL PROBLEMİ

DÖRDÜNCÜ BÖLÜM

ÇİZGE PARÇALAMA İÇİN ALGORİTMALAR

BEŞİNCİ BÖLÜM

EN KISA YOL PROBLEMİNİN ÇİZGE PARÇALANIŞI İLE ÇÖZÜMÜ

ŞEKİLLER LİSTESİ

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 (graf) adı verilen bu doğru ya da eğriler, çizge teorisinin temel öğeleridir. Çizge teorisi birçok tanım, teorem ve yeni kavramlarla donatılmıştır.

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 hep, 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.

Basit olarak çizgilerle birleştirilmiş noktalar topluluğu olarak tanımlanan çizgeler daha sonra topolojik olarak da incelenmeye başlanmıştır. Çizgelerle ilgili ilk çalışmalar Leonhard Euler (1707 – 1783) tarafından yapılmıştır. Bu çalışmalardan biri olan Königsberg Köprüsü (Dündar, 1998, b) problemi Euler tarafından çizgeler ile gösterilmiştir ve çözüme bu sayede daha kolay ulaşılmıştır.

Ekonomik planlamada ve endüstrinin her kesiminde uygulanan dağıtım şebekeleri, haritalama, üretimin akışı, trafiğin akışı gibi problemler çizgelerle ilgili örneklerdir. Ayrıca, doğrusal programlamanın belirli uygulamalarında ortaya çıkan ve çeşitli merkezlerin belirli çiftlerinin yollarla bağlanması, özel bir çizge yapısındadır. Çizge kuramı ile ilgili yöntemler başlangıçta sadece elektrik devrelerinde kullanılırken zaman içerisinde üretim planlama, doğrusal programlama, transport, v.b. alanlarında etkili bir şekilde kullanılmaya başlanmıştır. Uygulama alanları Bölüm 2.2'de verilmiştir.

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.

Bu çalışmada ilk bölümde çizge parçalaması ile ilgili bir literatür taraması yapılmıştır. İkinci bölümde çizgelerden, çizgeler kuramından ve uygulama alanlarından bahsedilmiştir. Üçüncü bölümde çizgeler kuramının uygulama alanlarından biri olan en kısa yol problemi ele alınmıştır. Bu problem için iki adet genel algoritma birer kısa uygulamalarıyla birlikte sunulmuştur. Dördüncü bölümde ise çizge parçalama için Kernighan ve Lin'in algoritması verilmiştir. Son bölümde ise kendi geliştirdiğimiz 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.

BİRİNCİ BÖLÜM

LİTERATÜR TARAMASI

Bu bölümde çalışma süresince incelenen makaleleri içeren bir literatür taramasının sonuçları verilecektir. Tarama son yıllarda bilgisayar bilimleri alanındaki dergilerde ve makalelerde "çizge, çizge parçalanması" ve ilgili terimleri anahtar kelime olarak içeren makaleler arasında yapılmıştır.

Literatür taraması, ardışık algoritmalar ve paralel algoritmalar olmak üzere iki başlık altında sınıflandırılmıştır.

1.1 Ardışık Algoritmalar:

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)

Fiduccia ve Mattheyses (1982), bir iterasyonun $O(|E|)$ operasyon zamanı aldığı, Kernighan ve Lin'in (KL) algoritmasından esinlenmiş, 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. Herbir 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.

Pek çok araştırmacı çizge parçalama için genetik algoritmalar geliştirmişlerdir. Bunlar arasında Bui ve Moon'un, bir çizgeyi ikiye parçalayan algoritmasından kısaca şöyle söz edilebilir. İlk olarak kirişler yeniden sıraya sokulur. Bu yeni sıra, tesadüfi olarak seçilen bir düğümden başlayarak BFS (Breadth First Search) ile ziyaret edilen sıradır. Sonra başlangıç popülasyonu dengelenmiş iki parçaya ayrılır. Yeni bir jenerasyon oluşturmak için bu iki parçadan bir çift seçilir. Herbir parça kendi kesim kirişleri ağırlığına bağlı olan bir olasılık ile seçilir. Kesim kirişleri ağırlığı ile seçilme

olasılığı ters orantılıdır. Ebeveyn çifti seçilince bir soy yani dengelenmiş iki parça meydana gelir. Sonra KL algoritması uygulanarak soyun kesim kirişleri ağırlığı azaltılmaya çalışılır. Sonuç olarak evlatların yerine mevcut popülasyondaki çözüm konur.

Bui ve Moon kendi algoritmalarını KL (Kernighan-Lin) ve SA (Simulated Annealing) algoritmaları ile deneysel olarak karşılaştırmışlardır ve kendi algoritmalarının diğerlerinden daha kaliteli olduğu sonucuna varmışlardır.

Berger ve Bokhari (1987), bir geometrik algoritmanın en temel örneklerinden biri olan "recursive coordinate bisection" (RSB) algoritmasını tasarlamışlardır. Bir çizgedeki düğümler geometrik olarak da adreslenebilirler. Bunlar geometrik parçalama algoritmaları olarak adlandırılırlar.

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.

"Multilevel recursive spectral bisection" (multilevel-RSB) metodu, Barnard ve Simon (1993) tarafından tasarlanmıştır. Fiedler vektörünün hesap hızını arttırmaya yönelik çokaşamalı bir yaklaşım kullanır. Deneyler multilevel-RSB'nin aynı kalitede parçalama yaptığı ve de önemli ölçüde RSB'den hızlı olduğunu göstermiştir. Bu algoritma 3 safhadan oluşur. "coarsening" (bayağılaştırma), "partitioning"(parçalama), "uncoarsening"(ayrıştırma). Bayağılaştırma aşamasında kiriş ağırlıkları diğer kirişlere göre büyük olan düğümler üst üste çakıştırılır. Böylece çizge daha sade bir hal alır ve

de yüksek miktardaki hesaplama işlemlerinden önemli ölçüde kaçınılmış olur. Parçalama aşamasında bu çizge daha kolay, kaliteli ve de az zamanda parçalanır. Ayrıştırma aşamasında ise işlemcilerdeki çakışık düğümler ayrıştırılır.

"Multilevel KL algorithm", hızlı bir algoritmanın elde edilmesinde çokaşama yaklaşımının nasıl kullanılabileceğine dair başka bir çalışmadır. Bayağılaştırma aşaması esnasında, algoritma, başlangıç çizgesinin ardışık olarak artarak bayağılaşan yaklaşımlarını yaratır. Yeterli bayağılıkta bir graf bulunduğunda iki ya da p-yollu parçalamaya geçilir. Ayrıştırma aşamasında bu parçalama çizge hiyerarşisine yayılır. KL tipi bir algoritmaya parçalamayı gerçekleştirmek için periyodik olarak başvurulur. Bayağılaştırma aşamasında "maximal matching" (maksimal eşleme) yaklaşımı kullanılır. Ancak deneyler "heavy edge matching" (ağırlıklı ayrıt eşlemesi) metodunun en iyi sonuçları verdiğini göstermiştir. (Karypis, Kumar, 1994)

1.2 Paralel Algoritmalar:

Son zamanlarda araştırmacılar çizge parçalama için paralel algoritmalar da geliştirmeye başladılar. Bunun için birçok sebep var. 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 $v \in N_1$ ise v düğümünün bütün komşu düğümleri P_1'in dahil 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. Algoritmaları üç safhadan oluşur: çizge bayağılaştırma, çokaşamalı difüzyon, ve çokaşamalı geliştirme.

Nakhimovski (1997), algoritmasında ilk olarak çizgede mümkün olduğunca az sayıda kirişi kesecek şekilde bir düzlem bulur. Ancak bu düzlemin kaç tane kirişi kestiği kesin olarak tespit edilememekle birlikte bu kesim kirişleri sayısı, düğüm koordinatları sayesinde yaklaşık olarak hesaplanır.

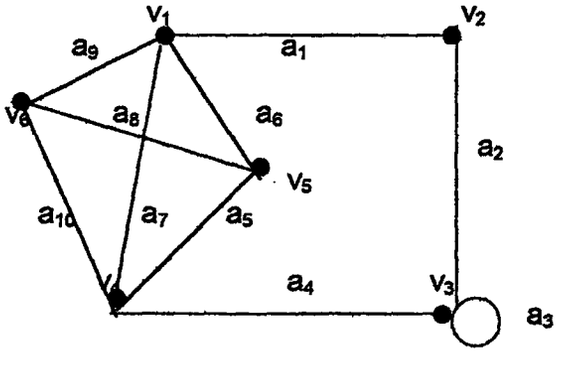
İKİNCİ BÖLÜM

ÇİZGELER KURAMINDA ÖNEMLİ KAVRAMLAR

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 d bağıntısı vardır ve bu $d(a_k)=(v_i,v_j)$ 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.

İki düğüme her zaman bir kiriş eşleme zorunluluğu yoktur. Ancak, bir kirişin varlığı $d(a_k)=(v_i,v_j)$ eşlemesinin varlığına bağlıdır. Bu nedenle uç noktaları olmayan bir kiriş düşünülemez. çizgenin düğümleri noktalarla, kirişleri da açık doğru parçaları açık yay parçalarıyla gösterilir. Örneğin, $N=\{v_1,v_2,...,v_6\}$ ve $E=\{a_1,a_2,...,a_{10}\}$ kümeleriyle ve d bağıntısı; $d(a_1)=(v_1,v_2)$, $d(a_2)=(v_2,v_3)$, $d(a_3)=(v_3,v_3)$, $d(a_4)=(v_3,v_4)$, $d(a_5)=(v_4,v_5)$, $d(a_6)=(v_1,v_5)$, $d(a_7)=(v_1,v_4)$, $d(a_8)=(v_6,v_5)$, $d(a_9)=(v_1,v_6)$, $d(a_{10})=(v_4,v_6)$ biçiminde verilirse bu çizge Şekil 1'deki gibi çizilebilir:



Şekil 1 (Örnek Çizge)

Yukarıdaki şekildeki a_7 ve a_8 kirişleri çapraz kirişlerdir. Böyle kirişlerin içten bağlı oldukları, yani a_7 kirişindeki bir akışın a_8 kirişine geçemeyeceği varsayılmıştır. Kirişleri düğümlerden başka hiçbir noktada kesişmemek üzere düzleme çizilen ağlara topolojik düzlemsel ağlar denir. Topolojik düzlemsel olmayan ağlar düzleme çizildiğinde, çapraz durumda olan kirişlerin arakesiti bir düğüm olarak kabul edilemez.

Bir çizgenin farklı iki düğümü arasında en çok bir yay (her iki yönde) varsa böyle bir çizgeye basit çizge denir. Aksi halde çizge çok katlı çizge adını alır. N kümesinin eleman sayısı çizgenin mertebesini ve bir düğümü uç düğüm kabul eden kirişlerin sayısı o düğümün derecesini tanımlar. Ayrık (izole) düğüm, derecesi sıfır olan düğümdür.

Bir G=(N,E) çizgesinin düğümlerinin herhangi bir dizisi $\{v_0,v_1,...,v_{n-1},v_n\}$ olsun. Eğer her i=1,2,...,n için $(v_{i-1},v_i)\in E$ ise bu diziye v_0 ve v_n düğümlerini birleştiren yol adı verilir.

Bir $\{v_0,v_1,...,v_{n-1},v_n\}$yolunda v_0 ve v_n düğümlerinin dereceleri 1 ve diğer düğümlerin dereceleri 2 ise böyle bir yola elementer yol, $v_0=v_n$ olan elementer yola çevre denir.

Bir çizgenin her farklı düğüm çiftini birleştiren bir yol varsa bu çizgeye birleştirilmiş çizge denir. Herbir düğümü diğer tüm düğümlerle bağlantılı olan çizge tam çizge ve çevre içermeyen birleştirilmiş çizge ağaç olarak tanımlanmaktadır.

Katlı kiriş, bukle ve izole düğüm içermeyen yönsüz çizgeye doğrusal çizge denir.

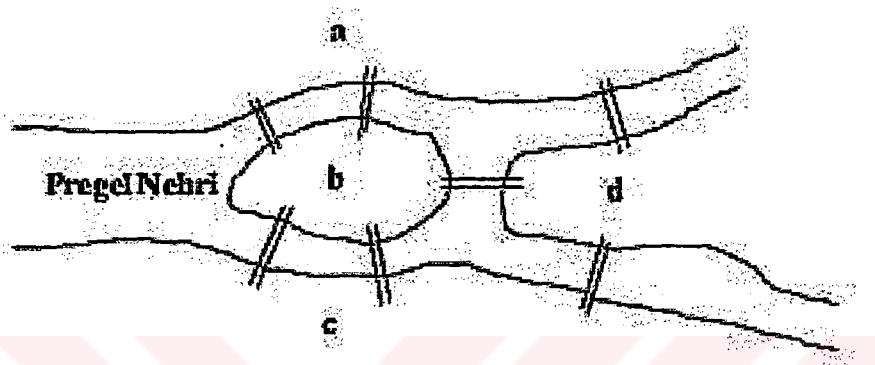
Yol kavramı kirişlerle de tanımlanabilir: çizgenin kirişlerinin herhangi bir $\{a_1,a_2,...,a_m\}$ dizisinde ardışık kirişlerin birer uç noktaları ortak ise bu diziye yol denir. Bir yolda her $i\neq j$ için $a_i\neq a_j$ ise böyle bir yol basit yoldur. (Öztürk, 1994)

Bir G çizgesinin her kirişine $d_k\geq 0$, (k=1,2,...,m) değeri bağlandığında, herhangi v_i ve v_j düğümleri arasında bulunan yollar içinde bu yolların her birinin kirişlerine

bağlanan değerler toplamı minimum olan yola v_i ve v_j düğümleri arasındaki en kısa yol denir.

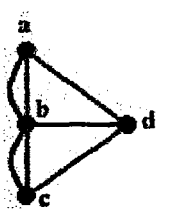
2.1 Königsberg Köprüleri Problemi

18.ci yüzyılın ortalarında Königsberg şehri Pregel nehrinin iki yakasında kurulmuştur ve nehirde iki ada bulunmaktadır. Bu adalardan biri şehrin iki yakasına ikişer köprü, diğeri birer köprü ile bağlanmıştır. Şekil 2'de de görüldüğü gibi ayrıca adalar arasında da bir köprü bulunmaktadır.



Şekil 2 (Königsberg Köprüleri)

Königsberg şehri sakinleri; "herhangi bir noktadan harekete başlayıp, yedi köprünün hepsinden sadece bir kez geçen sürekli bir yol bütün noktaları dolaşıp tekrar başlangıç noktasına dönebilir mi?" sorusuna cevap aramaktadırlar. Euler 1736 yılında bu problemin çözümünün olamayacağını yani bu koşullara uyan yolun bulunamayacağını göstermiştir. Bunu gösterirken, problemi bir çizge üzerinde daha kolay temsil ederek işe başlamıştır. Burada kıyılar a ve c, adalar ise b ve d noktaları ile gösterilmiştir. Bu noktalar düğüm noktaları, kıyılar ile adaları birleştiren yollar da kirişlerdir. Bu probleme ait çizge şekil 3'de görülmektedir.



Şekil 3 (Königsberg Köprülerinin Çizge İle Gösterimi)

Şimdi "herhangi bir düğümden harekete başlayıp, bütün düğümleri ziyaret edip, bütün kirişlerden de sadece birer kez geçerek başlangıçtaki düğüme dönülebilir mi?" sorusuna cevap bulmak için düğümlerin derecelerini ele almak gerekir. Buna göre şekilde a düğümünün derecesi 3, b düğümünün derecesi ise 5'dir.

Probleme geri dönüldüğünde, harekete başlanan düğümün tek dereceli olmaması gerektiği görülür. Çünkü böyle bir düğüm ile başlanırsa diğer düğüme geçebilmek için kirişlerden biri kullanılacağından geriye çift sayıda kiriş kalır ki bu düğümden bir daha geçmek için bir kiriş daha kullanılması sonucu geriye tek sayıda kiriş kalır. Bu kiriş en az bir tane olacaktır. Yani bu durumda en az bir tane kiriş kullanılamayacaktır. Demek ki böyle tek dereceli bir düğüm ile bitiş düğümü olamayacaktır. Aynı zamanda tek dereceli bir düğüm ara düğüm de olamayacaktır. Sonuç olarak bu problemde harekete başlanacak düğüm ve ara düğümler de çift dereceli olmalıdır. Oysa şekildeki çizgede bütün düğümler tek derecelidir. O zaman problemin çözümü yoktur. (Dündar, 1998, b); (Tulunay, 1982)

Euler bu problemle çizge kuramının temellerini atmıştır.

2.2 Çizge Kuramının Uygulamaları

Çizge teorisi kendine özgü tanımları kullanarak fen bilimleri ve sosyal bilimlerin pek çok problemlerini çözmede yaygın olarak kullanılmaktadır. Burada çizge kuramıyla çözülebilen bazı problemlerden söz edilmiştir.

2.2.1 Eşleme Problemi: Bir işletme genişlemeyi ve yeni bölümler açmayı planlıyor. Yeni bölümler için bir desinatör, bir mühendis, bir bilgisayar programcısı, bir veri analisti ve bir yönetici asistanı almayı amaçlıyor. Bu beş iş için herbiri iki ya da işe uygun beş başvuru yapıldığında nasıl bir personel ataması yapmalıdır? Adaylar ve işler bir çizgenin düğümleri olarak alınıp, bir aday bir işe uygunsa bu iki düğüm bir kirişle birleştirilerek problemin çizgesi elde edilir. Bu çizgede işlerin kümesi adayların kümesine eşlenerek problem çözülür. Bu tip problemlere çizge teorisinde eşleme (matching) problemi denir. Çeşitli çözüm algoritmaları vardır. (Christofides, 1986)

2.2.2 Gezgin Satıcı Problemi: Bir boya firması aynı kazan içinde dört farklı renkte boya üretmekte ve bir renk üretildikten sonra diğer renk boyayı üretebilmek için kazanın temizlenmesini beklemek gerekmektedir. Bir gün içinde dört boyanın üretimi sırasında nasıl bir plan izlenmeli ki bekleme süresi minimum olsun? Boya renkleri bir çizgenin düğümleri olarak alınır, ardarda üretilen iki renk varsa bu düğümler bir kirişle birleştirilir ve kirişe bekleme süresi ağırlık olarak atanırsa; ağırlıklandırılmış bir çizge elde edilir. Bu problemde amaç, bir düğümden başlayarak çizgenin tüm düğümlerini tam bir kez geçerek, başlangıç düğümüne mümkün olan en kısa sürede varan bir çevrenin bulunmasıdır. Böyle problemler çizge teorisinde gezgin satıcı (travelling salesman) problemi olarak bilinmektedir. Gezgin satıcı probleminin çözümü için pek çok çizge algoritması üretilmiştir. (Taha, 1997);(Balas, 1989)

2.2.3 En Geniş Bağımsız Küme Problemi: Yürütülmek istenen n tane proje düşünülsün. Her bir x_i projesini geliştirmek için p tane kaynağın bir $R_i \subseteq \{1,2,...,p\}$altkümesine ihtiyaç olsun. Aynı zaman periyodu içinde kaynakları en çok kullanarak, maksimum sayıdaki hangi projeler bitirilebilir? Her bir proje bir çizgenin düğümleri olarak alınır. İki proje aynı kaynağı ortak olarak kullanıyorlarsa bu iki düğüm bir kirişle birleştirilir. Bu durumda problem, bu çizge içinde maksimum sayıda birbirine komşu olmayan düğümlerin bulunması problemidir. Böyle problemler çizge teorisinde en geniş bağımsız küme (maximum independent set problem) problemi olarak bilinir ve çözüm için pek çok algoritma vardır. (Christofides, 1986)

2.2.4 Örten Küme Problemi: Bir üniversite kampüsünde öğrencilerin güvenliğini sağlamak için kampüs içinde belirli yerlere acil telefonlar konularak bir güvenlik merkezi oluşturulmak isteniyor. Kampüs içindeki herbir yola en az bir telefonla hizmet götüren minimum sayıda telefonla bu güvenlik merkezi nasıl kurulabilir? Sokakların kesişimine bir telefon konarak, her telefonun en az iki sokağa hizmet etmesi sağlanabilir. Sokakların kesim noktaları bir çizgenin düğümleri ve kesişen sokaklar da çizgenin kirişleri olarak alınırsa, problemin çizgesi oluşturulur. Bu problem, bir çizgede tüm kirişleri örten minimum sayıdaki düğümlerin bulunmasıdır. Çizge teorisinde bu tip problemler minimum örten küme (set covering problem) problemi olarak bilinir. Çeşitli çözüm yöntemleri bilinmektedir. (Taha, 1997)

2.2.5 Dağıtım Problemi: Stokları bilinen n depodan, talepleri bilinen m tane pazara, i'inci deponun j'nci pazara c_{ij} maliyeti ile nakliyat yaptığı bilindiğinde; minimum maliyetle arz ve talebi karşılayacak bir dağıtım yapılması problemi, dağıtım problemi olarak bilinir. Pazarlar ve depolar bir çizgenin düğümleri olarak alınır ve bir pazar bir depodan mal alıyorsa bu iki düğüm bir kirişle birleştirilir. Bu problem çizge teorisinde, bir çizgede düğümlerin; her kümedeki düğümler birbirine komşu olmayacak şekilde en geniş iki kümeye ayrılması (bipartition set) problemi olarak bilinir. (Taha, 1997)

2.2.6 Merkez Yerleştirme Problemi: Hastane, polis karakolu, itfaiye merkez gibi acil servislerin bir şehirdeki yerleşimindeki amaç, bu merkezlerin şehrin her yerine olabildiğince en kısa sürede (en kısa uzaklıkla) erişebilecek şekilde yerleştirilmesidir. Yerleşim alanları bir çizgenin düğümleri, bir yerleşim alanından diğerine bir yol varsa bu iki düğüm bir kirişle birleştirilerek bir çizge oluşturulur. Problem, bir çizgede en uzak düğüme en kısa sürede (en kısa uzunlukla) ulaşacak şekilde acil merkezlerin yerleştirilmesidir. Bu problem çizge teorisinde, bir çizgenin merkezlerinin bulunması olarak bilinir. Çeşitli çözüm algoritmaları geliştirilmiştir. (Christofides, 1986)

2.2.7 En Kısa Yol Problemi: Bir havayolları işletmesinin uçuş haritası ele alındığında, belirlenmiş iki şehir arasında bir uçuş hattı var mıdır? Birden fazla uçuş hattı varsa en kısa sürede ulaşım yapmak için hangi uçuş yolu seçilmelidir? Şehirler bir çizgenin düğümleri, iki şehir arasında bir uçuş varsa bu iki düğüm bir kirişle birleştirilerek ve iki şehir arası ulaşım süresi kirişe ağırlık olarak bağlanarak bir çizge oluşturulur. Bu çizgede, belirlenmiş iki düğüm arasında en kısa zamanda ulaşım problemi çözülür. Bu tip problemler çizge teorisinde en kısa yol problemi (shortest route problem) olarak bilinir. Bu problemin çözümü ile ilgili birçok çözüm algoritması üretilmiştir. En kısa yol problemi sonraki bölümde ayrıntılı ele alınmıştır. (Christofides, 1986)

2.2.8 Algoritma Tasarımı ve Paralel Hesaplama: Belirli bir programlama dilinde yazılarak, pek çok karmaşık işlemi yapıp problemlerin çözümünü sağlayan bilgisayar programlarının, akış diyagramları birer çizgedir. Mühendislik ve istatistiğin çeşitli uygulamalarında ortaya çıkan büyük boyutlu denklem sistemlerinin parçalarına ayrılarak çözümünün bulunmasında çizgelerden yararlanılmaktadır. (Dündar, 1998, a); (Lovazs, Winkler, 1996)

2.2.9 Ağ Akışı Problemi: Başta internet olmak üzere günlük hayatta kullanılan her tür ağda ki başta bilgisayar ağları, telefon ağları, kanalizasyon ağları ve su ağları gelir. Bu ağlar birer çizge ile sembolize edilip, bunlara ait sorunların çoğu çizge teorisi yöntemleri ile kolayca çözülebilir. (Christofides, 1986)

2.3 Ağ Akışı:

Bir akış (elektrik akımı, sıvı-akımı, trafik akımı, yiyecek-giyecek akımı,... gibi) şebekesini, kirişlerine akışlar karşı getirilmiş bir çizge gibi düşünmek oldukça yararlı sonuçlar vermektedir. Akışın incelenmesinde, buklesiz ve birleştirilmiş yönlü çizgeleri dikkate almak yeterlidir. Bu tip özel çizgelere, yani, kirişlerinde bir akış oluşmuş, buklesiz ve birleştirilmiş yönlü çizgelere ağ (network) denilmektedir.

$v_1, v_2, \ldots, v_n$ gibi n düğüm noktasından ve bu noktaların bazı sıralı çiftlerini birleştiren (v_i, v_j) kirişlerinin bir kümesinden oluşmuş bir ağı gözönüne alalım. Böyle bir ağı $(v_1, v_{i_1}), (v_{i_1}, v_{i_2}), \ldots, (v_{i_k}, v_n)$ kirişlerinden oluşmuş $\mu(v_1, v_{i_1}, v_{i_2}, \ldots, v_{i_k}, v_n)$ yolları boyunca v_1 düğüm noktasından v_n noktasına doğru bir akış bulunsun, v_1 noktasına ağ'ın girişi, v_n noktasına ağ'ın çıkışı denir. Herbir sıralı (v_i, v_j), (i,j=1,2,...,n) nokta çiftine, negatif olmayan ve (v_i, v_j) kirişinin kapasitesi adı verilen bir $d_{ij} \geq 0$ tam sayısı bağlanmış olsun. Bu sayı birim zamanda (v_i, v_j) kirişi boyunca taşınabilmesi mümkün en fazla yük miktarını belirtir. v_r ve v_s düğüm çiftleri bir kirişle birleştirilmemiş ise $d_{rs}=0$'dır.

Kapasite tanımlamasının kirişlere olduğu gibi düğümlere de yapılması mümkündür. Bir v_i düğümüne bu düğümü uç noktası olarak kabul eden bir kiriş bağlantılı ve bu kirişteki akış v_i'den başka bir düğüme gidecek biçimde yönlendirilmişse, v_i'ye kaynak, eğer bu akış v_i'ye gelecek biçimde ise batak (tüketen) adını alır.

Bu açıklamalardan sonra ağ özel bir çizge olmakta ve bu dağıtım, iletişim gibi alanlarda çok kullanılmaktadır. Kısaca, dağıtım problemi kaynaktan tüketene minimum maliyetli ağ akışı problemi olarak ifade edilebilmektedir. Ayrıca, verilen bir ağ akışı üzerinde maksimum dağıtım kapasitesini bulmak yine bir doğrusal programlama problemidir. Bu problemin çözümünde birçok algoritma geliştirilmiştir. Burada bu algoritmalardan iki tanesi olan Dijkstra Algoritması ve Floyd Algoritması ele alınacaktır.

2.4 Çizgenin Matris İle Gösterilişi

Çizgelerin matrislerle olan yakın ilişkisi hem çizge kuramının gelişmesine ve hem de uygulama alanının yaygınlaşmasına neden olmaktadır. Çizge kuramında kirişlerle düğümler arasındaki karşılıklı ilişki, bir çizgenin en temel karakteristiğini oluşturur. Hangi düğümler hangi kirişlerle ya da hangi düğümlerin hangi düğümlerle bağlantılı olduğu açıkça gösterilebilir. Böyle bir gösterim bir matrisle yapılır. Matrisin her satırı bir düğüme ve her sütunu da bir kirişe ya da düğüme karşı getirilir. Bunun sonunda çizgenin $D=[d_{ij}]$ ile gösterilen bir bağlantı matrisi elde edilir. Bu matriste çizgenin j. kirişi i. düğümüyle bağlantılı ise $d_{ij}=1$ ve j. kirişi i. düğümüyle bağlantılı değil ise $d_{ij}=0$ alınır böylece matris belirlenir. Bu tanımdan da anlaşılacağı gibi bir çizgenin bağlantı matrisleri düğüm-düğüm bağlantı matrisi ve düğüm-kiriş bağlantı matrisi olarak iki biçimde oluşturulabilmektedir.

n düğümlü ve m kirişli yönlendirilmiş bir çizgenin $D_y=[d_{ij}]$ düğüm-kiriş bağlantı matrisi, aşağıdaki kurallara göre tanımlanan d_{ij} elemanlarının oluşturduğu (n x m) boyutlu bir matristir.

i) Eğer j'inci kiriş i'inci düğüm ile bağlantılı ve i'inci düğümden uzaklaşacak biçimde yönlendirilmiş ise, $d_{ij}= +1$ dir.

ii) Eğer j'inci kiriş i'inci düğüm ile bağlantılı ve i'inci düğüme yaklaşacak biçimde yönlendirilmiş ise, $d_{ij}= -1$ dir.

iii) Eğer j'inci kiriş i'inci düğüm ile bağlantılı değilse $d_{ij}=0$ dır.

Yönlendirilmemiş ve yönlendirilmiş çizgeler için tanımlanmış olan bu matrislere ayrı ayrı gereksinim duyulmakla beraber bazı uygulamalarda bağlantı matrislerinin başka biçimlerde oluşturulması problemin açıklamasına yararlı olmaktadır. En kısa yol probleminin çözümünde bağlantı matrislerinin başka biçimleri de verilmiştir.

2.5 En Kısa Yol Probleminin İfadesi ve Doğrusal Programlama ile İlişkisi:

Buklesiz ve her kirişine $d_k \geq 0$ biçiminde bir değer bağlanmış olan bir G(N,E) çizgesinde öncelikle şu sorular araştırılır:

i) Bu çizgede herhangi A ve B düğümleri arasında bir yol var mıdır?

ii) Bu çizgede herhangi A ve B düğümleri arasında bulunan yollar içerisinden en kısa olan yol hangisidir?

Burada A ve B noktaları arasında yolların mevcut olduğu varsayılmış ve bunlar içinden en kısa olan yolun bulunması amaçlanmıştır. Bu amaçla G çizgesinde herhangi bir yol μ ise, bu yolun d(μ) uzunluğu

$$d(\mu)=\sum_{v\in\mu} d(v)$$

ile hesaplanır.

Kirişlerine $d_k \geq 0$ maliyet değerleri bağlanmış bir buklesiz çizge G(N,E) olsun. Bu çizgenin düğümleri $N=\{v_1,v_2,...,v_n\}$ ile gösterildiğine göre G'nin herhangi iki v_i,v_j düğümleri arasındaki yollar içinde uzunluğu en kısa olanına en kısa yol denir.

Problem bir başka açıdan ele alınırsa; bir düzlemde $v_1,v_2,...,v_n$ gibi n tane noktanın verildiği varsayılsın. Bu noktaların (v_i,v_j) biçimindeki sıralı ikilileri q_{ij} uzunluğunda (i,j=1,2,...,n) verilen yollar ile birbirine bağlanmış ve böylece yolların birleştirilmiş bir dizisi elde edilmiş olur. Uygulamada bir ağın herhangi bir noktasından bir diğer noktasına en kısa yoldan ulaşmak önemli bir problemdir. Ayrıca, bir ağın herhangi iki sabit noktası arasındaki en kısa yolu bulmak için düzenlenen her problem, bir doğrusal programlama problemidir. (Bakoğlu, 1999)

En kısa yol probleminin bir doğrusal programlama problemi olduğunun ispatı şöyledir: v_1'den v_n'e giden en kısa yol belirlenmek istensin.

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), μ'ye ait ise, n^2 sayıda olan ve $0\leq d_{ij}\leq 1$ (i,j=1,2,...,n) ile tanımlanan d_{ij} sayılarının birleştirilen takımı bire eşitlenir ve aksi halde yani (v_i,v_j) μ'ye ait değilse sıfıra eşitlenir. Bu halde v_1 ve v_n arasındaki en kısa yolu bulma problemi

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

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

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

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

koşulları altında, v_1'den v_n'ye giden yolun uzunluğu olan,

$$z = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} d_{ij}$$

doğrusal ifadesini minimum yapan tam sayılı çözümünü bulmaktır.

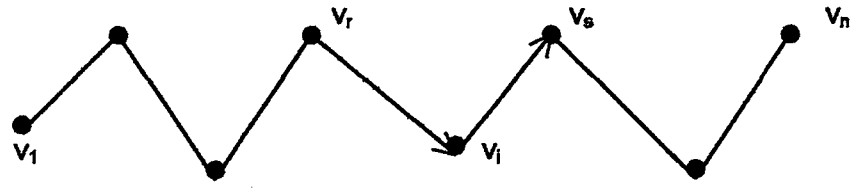
v_1 ve v_n arasındaki herhangi bir $\mu(v_1, v_{i_1}, ..., v_{i_k}, v_n)$ yolunun d_{ij}'ye (i,j=1,2,...,n) karşı gelen belirli kümesi yukarıdaki a), b), c) ve d) koşullarını sağlar. Bunun ispatı şöyledir:

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

b) koşulu, $\sum_{j=1}^{n}(d_{ij} - d_{ji}) = 0$ (i≠1, i≠n) idi. Bu koşul, $(d_{i1} - d_{1i}) + (d_{i2} - d_{2i}) + ... + (d_{in} - d_{ni}) = 0$ 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 d_{ij} ve d_{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 $d_{ij}=d_{ji}=0$ (j=1,2,...,n) olur. Buradan da $\sum_{j=1}^{n}(d_{ij}-d_{ji})=0$ yazılır.

Tersine olarak v_i noktası μ yoluna ait ise, buradan μ yolu v_i noktasını başlangıç ve son noktası olarak içeren iki kirişi kapsar.



Şekil 4 (Örnek Çizge2)

Şekil 4'de görüldüğü gibi bu iki kirişten birinin (v_r,v_i) olduğu ve bu kirişte v_i'nin bir son noktası durumunda, diğerinin (v_i,v_s) olduğu ve burada da v_i'nin bir başlangıç noktası durumunda bulunduğu görülmektedir. Buna göre, $\sum_{j=1}^{n}(d_{ij}-d_{ji})$ toplamındaki tüm d_{ij} (i,j=1,2,...,n)'lerden sadece $d_{ri}=1$ ve $d_{is}=1$ ve diğerleri sıfıra eşittirler.

v_i noktası μ yoluna eşit olduğu zaman, yukarıdaki açıklamalardan sonra, i indisinin bulunduğu yere göre yani, d_{ij}'nin işareti pozitif, d_{ji}'nin işareti negatif olarak saptanır ve böylece $\sum_{j=1}^{n}(d_{ij}-d_{ji})=0$ olduğu görülür. Bu da b) koşulunun sağlandığını ifade eder.

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 alınır ve yukarıdaki şekil görüntüsü altında, b) koşuluna benzer düşüncelerin bir sonucu olduğu görülür.

Burada doğrusal programlama probleminin herbir tamsayılı çözümünün belirlediği

$$d_{1j_1} = d_{j_1j_2} = \ldots = d_{j_{n-1}n} = 1 \qquad (2.1)$$

biçimindeki bir sayı kümesinin bir μ yoluna karşılık olduğunu göstermek gerekir. Gerçekten,

$$\sum_{j=1}^{n} d_{1j} = 1 + \sum_{j=1}^{n} d_{j1}$$

denkleminden de anlaşılacağı gibi en az bir tane $d_{1j}=1$ vardır. (Bu ise ilk kirişin μ yoluna ait bulunduğunun bir ifadesidir.) Benzer olarak,

$$\sum_{j=1}^{n} d_{j_1 j} = \sum_{j=1}^{n} d_{j j_1} \geq 1$$

ifadesinden de anlaşılacağı gibi hiç olmazsa $d_{j_1j_2}=1$ olan en az bir eleman vardır. Bu şekilde ilerleyerek (2.1) gibi bir dizi oluşturulur. Bu dizi ise ancak v_n uç noktasında sonuçlandırılır. Ayrıca herhangi bir v_s düğümünde

$$\sum_{j=1}^{n} (d_{sj} - d_{js}) = 0$$

eşitliği elde edilir.

Görüldüğü gibi en kısa yol problemi, bir doğrusal programlama problemine dönüştürülebilmektedir. Ancak bu problemin simplex yöntemle çözümü, koşulların özelliklerine bağlı olarak, çok uzun zaman alabilir. En kısa yol probleminin çözümü için geliştirilmiş Dijkstra ve Floyd algoritmalarına da üçüncü bölüm'de değinilmiştir.

ÜÇÜNCÜ BÖLÜM

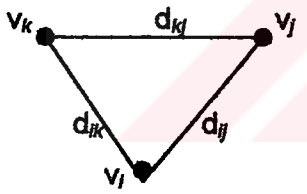
EN KISA YOL PROBLEMİ

Bağlantı matrisi $D=[d_{ij}]$ olarak verilen bir $G=(N,E)$ çizgesinde $d_{ij} \geq o$ olmak üzere, en kısa yol problemi ;

i. Belirli bir başlangıç düğümü s için, s ile bütün diğer düğümler arasındaki $(v_l \in N)$ en kısa yolunun bulunması,
ii. Bütün düğümlerin diğer tüm düğüm çiftleri arasındaki en kısa yolların bulunması.

olarak bilinir.

Problem i'nin çözümü s'den t'ye en kısa yolu bulan metod tarafından iterasyonlar sırasında s'den $(\forall v_l \in N)$'e az bir işlem zamanı olan $O(N^2)$ zamanda bulunabilir. Problem ii ise, problem i'yi çözen bir yöntemle, her iterasyonda farklı bir düğümü başlangıç düğümü olarak alıp, algoritmayı n kez uygulamayla bulunabilir. Burada D matrisinin "üçlü operasyon" adı verilen üçgen kurallarını sağlamadığı varsayılır. Yani bütün i,j ve k'lar için d_{ij}, $d_{ik}+d_{kj}$'den az olmayabilir. $d_{ij}>d_{ik}+d_{kj}$ olabilir. Bu durum şekil 5'de çizge üzerinde gösterilmiştir.



(v_i, v_j) bağlantısı yok ise genelde $d_{ij}=\infty$ alınır.

Şekil 5 (Üçlü Operasyon)

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.

3.1 Dijkstra Algoritması

Başlangıç "*s*" ile hedef "t" düğümleri 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. Algoritma aşağıdaki adımlarla çalışır;

$l(v_i)$, v_i düğümündeki etiket olsun.

Adım1 (başlangıç)

$l(s)=0^+$, etiketi sabitlenir. Bütün $v_i \neq s$ *için* $l(v_i)=\infty$ geçici etiketleri verilir. $p=s$

Adım2 (etiketlerin güncellenmesi)

Her $v_i \in E(p)$ şartını sağlayan v_i'lerden geçici etiketli olanları

$l(v_i) = min\,[l(v_i)\,,\, l(p)+c(p,v_i)]$ ile güncellenir.

Adım3 (bir etiketin sabit etiket olarak işlenmesi)

Bütün geçici etiketli düğümlerde $l(v_i^*) = min[l(v_i)]$ için v_i^*'ı bulunur.

Adım4

v_i^*'ın etiketi sabitlenir. $p=v_i^*$ olarak p de güncellenir.

Adım 5

- *s*'den *t*'ye olan en kısa yol istenirse;
  $p=t$ ise $l(p)$ en kısa yoldur. $p \neq t$ ise adım 2'ye dönülür.
- *s*'den bütün diğer düğümlere olan en kısa yol istenirse:

Eğer bütün düğümler sabit etiketli olursa bu etiketler en kısa yollardır. Durulur. Eğer bazı etiketler geçici ise adım 2'ye gidilir.

Bu algoritmanın en kısa yolu verdiğinin ispatı şöyledir;

Sabit etiketler herhangi bir aşamadaki en kısa yollardır. S_1 bu şekilde etiketleri sabitlenmiş düğümlerin kümesi ve S_2 etiketleri geçici olan düğümlerin kümesi olsun. Her iterasyondaki ikinci adımın sonunda geçici etiket $l(v_i)$, s'den v_i'ye giderken S_1 kümesindeki düğümlerden geçen en kısa yoldur. Çünkü herbir iterasyonda S_1'e sadece bir düğüm girer ve $l(v_i)$'nin güncellenmesi adım2'de verilen formülasyonu gerektirir.

s'den v_i^*'a olan en kısa yol bir tek S_1'den değil, S_2'den de en az bir düğümden geçsin ve $v_j \in S_2$ bu yoldaki ilk düğüm olsun. Yolun v_j'den v_i^*'a giden kısmı mutlaka bir maliyete sahiptir. Bu maliyet de Δ olsun. Böylece $l(v_j) < l(v_i^*)-\Delta < l(v_i^*)$ dır. Bu, $l(v_i^*)$ en küçük geçici etikettir iddiasını yalanlar, böylece v_i^*'a giden en kısa yol S_1'deki düğümlerden geçer ve $l(v_i^*)$ bu yüzden yolun uzunluğudur.

$S_1 = \{s\}$ olarak kurulmuştu ve her iterasyonda v_i^* eklendi. Bütün iterasyonlarda $\forall v_i \in S_1$ için en kısa yol uzunlukları $l(v_i)$'dir varsayımı doğrudur. Böylece tümevarım sayesinde algoritma tarafından üretilen cevap en elverişlidir.

n düğümlü ve bütün düğümleri birbirlerine bağlı bir çizgede, *s* ile bütün diğer düğümler arasındaki en kısa yollar hesaplanmak istenildiğinde algoritma adım2'de $n(n-1)/2$ kadar, ve adım3'de ise aynı bir $n(n-1)/2$ sayıda ekleme ve karşılaştırmayı gerektirir. Ayrıca 2.ve 3. adımda hangi düğümlerin geçici etiketlendiğini bulmak ise ekstra $n(n-1)/2$ karşılaştırma gerektirir. Bu durum herhangi bir çizgede s düğümünden belirlenmiş bir t düğümüne olan en kısa yolun hesaplanmasında gerekli olan işlem sayısının bir üst sınırıdır ve de t son düğüm olarak sabit olarak etiketlendiğinde gerçekleştirilir.

s'den başlayan en kısa yolların uzunlukları son etiket değerlerinden belirlendiği gibi, rotalar da aşağıdaki tekrarlamalı eşitlikten ortaya çıkar. Böylece eğer v_i' bir önceki *s*'den v_i'ye en kısa yol ise, herhangi bir verilen v_i düğümü için v_i' şu şekilde bulunur.

$$l(v_i') + d(v_i', v_i) = l(v_i) \qquad \textbf{(3.1)}$$

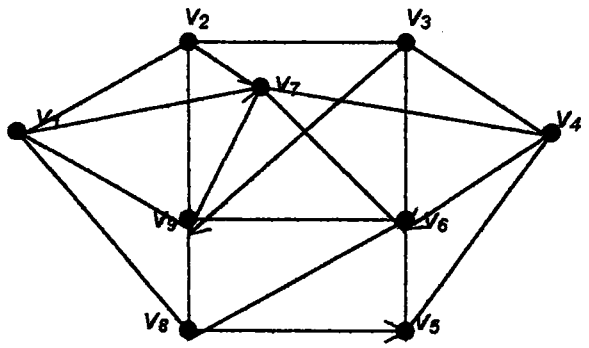
Eğer *s*'den *t*'ye olan en kısa yol tek ise ve (v_i', v_i) bir ağaç gibi düşünülecek olursa, başlangıcı yani kökü *s*'dir. Eğer *s*'den herhangi bir başka düğüme birden fazla en kısa yol varsa yukarıdaki eşitlik bazı v_i'ler için, birden fazla v_i' tarafından sağlanır. *(s-v_i)* arası sadece bir en kısa yol isteniyorsa aralarından biri keyfi seçilir ya da herhangi bir en kısa yoldan bütün görünen (v_i', v_i) bağlantıları ele alınabilir. Ancak bu bağlantılar ağaç oluşturmaz, sadece "temel ilişki" adı verilen genel bir çizge oluşturur. (Christofides, 1986)

3.1.1 Djkstra Algoritması İçin Bir Örnek

Aşağıdaki verilen maliyet matrisi için v_1'den bütün diğer düğümlere olan en kısa yolları bulalım. Sabit etiketlenen düğüme (*) işareti konacaktır.

D=	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
v_1		10					3	6	12
v_2	10		18				2		13
v_3		18		25		20			7
v_4			25		5	16	4		
v_5				5		10			
v_6			20		10		14	15	9
v_7		2		4		14			24
v_8	6				23	15			5
v_9	12	13				9	24		

Bağlantı matrisimize göre çizge şekil 6'daki gibi çizilebilir.



Şekil 6 (Djkstra Algoritması İçin Bir Örnek)

Algoritmanın prosedürü şu şekildedir.

Adım1 $l(v_1)=0^+$, $l(v_i)=\infty \; \forall \; v_i \neq x_1$, $p=v_1$

İlk İterasyon

Adım2 $E(p)= E(v_1)=\{v_2, v_7, v_8, v_9\}$ Başlangıç olarak seçilen düğüme direkt bağlantılı olan bu düğümler $l(v_i) = min\ [l(v_i)\ ,\ l(p)+c(p,v_i)]$ formülasyonuna göre şu etiket değerlerini alırlar . Önce v_2 hesaplansın.

$l(v_2)=min[\infty, 0^+ +10]=10$ olur.

aynı şekilde $l(v_7)=3$, $l(v_8)=6$, $l(v_9)=12$ bulunur.

Adım3 $min[10, 3, 6, 12, \infty]=3$ Bu etiketlerden en küçük değere sahip olan etiket
$v_2\ v_7\ v_8\ v_9$

sabitlenir ve artık bu rakam, başlangıç düğümü olarak seçtiğimiz v_1 düğümünden, v_7 düğümüne olan en kısa yolu verir. Diğer hesaplanan etiketler ise, ilgili düğümlere geçici olarak atanır.

Adım4 v_7'nin etiketi artık sabitlenir. $l(v_7)=3^+$ $p=v_7$

Adım5 Şekil7'de görüldüğü gibi henüz bütün düğümlerin etiketleri sabitlenmedi. Artık diğer iterasyonlara adım2'den başlanır.

İkinci İterasyon

Adım2 $E(p)= E(v_7)=\{v_2, v_4, v_6, v_9\}$ Herbiri yeniden etiketlendirilecekler.

$l(v_2)=min[10, 3^+ +2]=5$

$l(v_4)=min[\infty, 3^+ +4]=7$

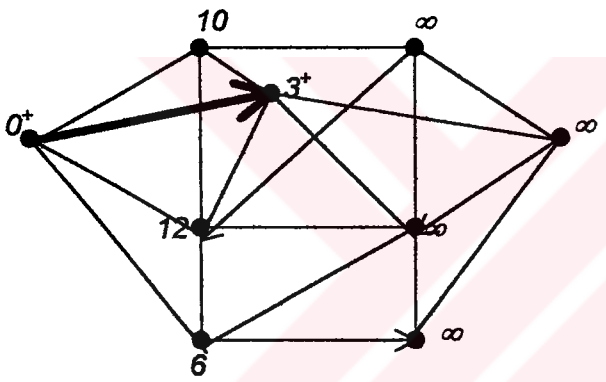
$l(v_6)=min[\infty, 3^+ +14]=17$

$l(v_9)=min[12, 3^+ +24]=12$

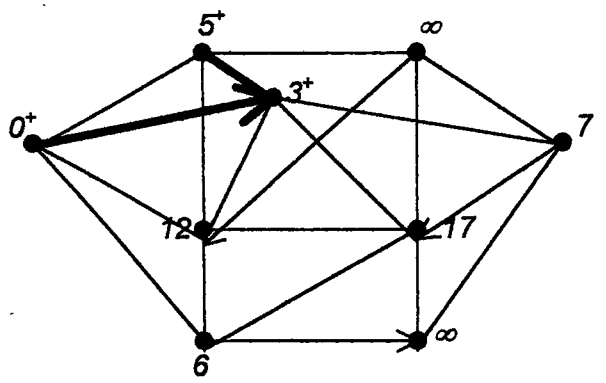
Adım3 $min[5, 7, 17, 6, 12, \infty]=5$ Buradaki 6 değeri birinci iterasyondan gelen v_8 düğümünün geçici etiketidir.

Adım4 $l(v_2)=5^+$ $p=v_2$ Minimum etiketli düğüm olan v_2 artık sabit etiketlendi.

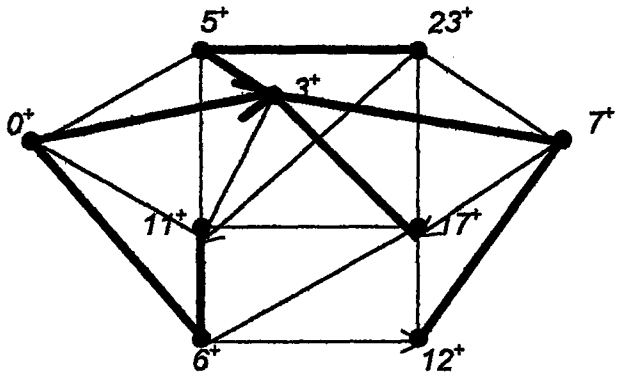
Adım5 Bir sonraki iterasyona geç ve Adım2'den devam et.



Şekil 7 (Djkstra Algoritması Örneği İterasyon1)



Şekil 8 (Djkstra Algoritması Örneği İterasyon2)



Şekil 9 (Djkstra Algoritması Örneği Son Etiketler)

Üçüncü İterasyon

Adım2 $E(p)= E(v_2)=\{v_1, v_3, v_7, v_8\}$ kümesinde sadece $\{v_3, v_8\}$ geçici etiketlidir.

$l(v_3)=min[\infty, 5^+ +18]=23$

aynı şekilde $l(v_8)=12$

Adım3 $min[23, 7, 17, 6, 12, \infty]=6$

Adım4 $l(v_8)=6^+$ $p=v_8$

Adım5 Adım2'ye dön.

Bu şekilde devam ederek şekil 9'da gösterilen son etiketler elde edilir. Başlangıç düğümü v_1'den herhangi bir düğüme (örneğin v_2'ye) olan en kısa yolu bulmak için (3.1) formülü ard arda uygulanır. Böylece $v_i=v_2$ alarak, v_1'den v_2'ye giden en kısa yoldaki v_2'den bir önceki düğüm olan v_2' düğümü şu eşitliği sağlayan düğümdür:

$$l(v_2') + d(v_2', v_2) = l(v_2) = 5$$

Bu eşitliği sağlayan tek v_2' düğümü v_7'dir. (3.1) eşitliğinin tekrar uygulanmasıyla $v_i=v_7$ alınarak, v_1'den v_2'ye giden en kısa yoldaki v_7'den bir önceki düğüm olan v_2' düğümü şu eşitliği sağlayan düğümdür:

$$l(v_7') + d(v_7', v_7) = l(v_7) = 3$$

Bu eşitliği sağlayan tek v_7' düğümü v_7'in kendisidir. Böylece v_1'den v_2'ye giden en kısa yol (v_1, v_7, v_2)'dir. v_1 başlangıçlı bütün en kısa yollar bir ağaç oluşturur ve bu yollar kalın çizgilerle çizgede belirtilmiştir.

3.2 Floyd Algoritması

Floyd algoritması Dijkstra algoritmasından daha genel bir algoritmadır çünkü bir ağda herhangi iki tepe arasındaki rotayı herhangi bir ağaç meydana getirmeden belirler. Halbuki Dijkstra algoritması bir v_1 başlangıç tepesinden başlamak üzere bütün diğer tepelere ulaşan v_1 köklü bir ağaç meydana getirir. Algoritma n düğümlü bir ağı n satır ve n sütunlu bir kare ağırlıklı bitişiklik matrisi olarak ele alır.

Floyd algoritmasının fikri açıktır. Verilen i,j ve k düğümleri için (bkz. Şekil 5) i'den k'ya giderken $d_{ij}+d_{jk}<d_{ik}$ ise j'den geçilir. Yani üçlü operasyon temeline dayalı bir algoritmadır. Bu operasyonlar sistematik olarak ağ içinde şu adımlar kullanılarak uygulanır.

Adım0: D_0 ağırlıklı bitişiklik matrisi olarak tanımlanır ve S_0 matrisi şu kurallara göre oluşturulur. S_0 matrisindeki elemanlar soldan sağa 1'den başlayarak numaralandırılır ancak köşegen elemanlar (-) işaretiyle bloklanmış oldukları belirtilir.

k=1

D_0=

	1	2	...	j	...	n
1	-	d_{12}	...	d_{1j}	...	d_{1n}
2	d_{21}	-	...	d_{2j}	...	d_{2n}
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
i	d_{i1}	d_{i2}	...	d_{ij}		d_{in}
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
n	d_{n1}	d_{n2}	...	d_{nj}	...	-

S_0=

	1	2	...	j	...	n
1	-	2	...	j	...	n
2	i	-	...	j	...	n
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
i	1	2	...	j	...	n
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
n	1	2	...	j	...	-

Genel Adım k:

k'ıncı satır baş satır, k'ıncı sütun da baş sütun olarak tanımlanır. Üçlü operasyon D_{k-1}'deki her d_{ij} elemanına uygulanır. Eğer

$d_{ik}+d_{kj}<d_{ij}$ ($i\neq k$, $j\neq k$ ve $i\neq j$)

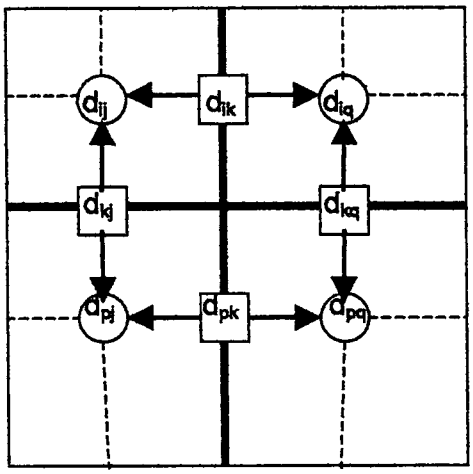
durumu sağlanıyorsa şu değişiklikler yapılır:

(a) D_{k-1}'deki d_{ij}'nin yerine $d_{ik}+d_{kj}$ koyularak D_k oluşturulur

(b) S_{k-1}'deki s_{ij}'nin yerine k konularak S_k oluşturulur. k=k+1 yapılır ve k'ıncı adım tekrarlanır.

.Algoritmanın k'ıncı adımı şekil 10'daki D_{k-1} matrisinin şematize edilmesiyle daha rahat anlaşılabilir. Burada k'ıncı satır baş satır ve k'ıncı sütun baş sütundur. Satır i=1,2,3,...,k-1 ve satır p=k+1,k+2,...,n yine aynı şekilde sütun j=1,2,3,...,k-1'inci sütunlar ve sütun q= k+1,k+2,...,n'inci sütunları göstermektedir. Üçlü operasyon şu şekilde uygulanır;

Baş satır ve baş sütun (kare içinde gösterilenler) elemanları toplamı birleştirilmiş kesişim elemanlarından (yuvarlak içinde gösterilenler) küçükse kesişim uzaklıklarını baş uzaklıklarla değiştirmek optimaldir.



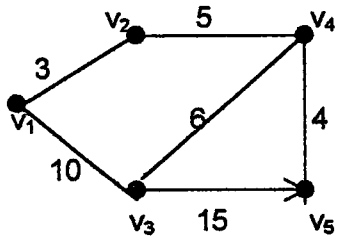
Şekil 10 (Floyd Algoritması Anahtar Satır ve Anahtar Sütunun Seçilmesi)

n adımdan sonra, D_n ve S_n matrislerinden i ve j tepeleri arasındaki en kısa yolun rotası şu kurallar kullanılarak belirlenir

1) D_n matrisinde, d_{ij} i ve j düğümleri arasındaki en kısa yolu verir.
2) S_n matrisinde, aracı düğüm $k=s_{ij}$ belirlenir, bu; rotayı $i \rightarrow k \rightarrow j$ şeklinde yönlendirici düğümdür. Eğer $s_{ik}=k$ ve $s_{kj}=j$ ise durulur; rotanın bütün aracı düğümleri bulunmuştur. Aksi takdirde i ve k düğümleri ve k ve j arasındaki prosedür tekrarlanır.

3.2.1 Floyd Algoritması İçin Bir Örnek

Şekil 11'deki ağ için herbir çift düğüm arasındaki en kısa yolun bulunması ele alınsın. Mesafeler (mil olarak) bağlantılarda verilmiştir. 3. ve 5. düğümü birbirine bağlayan kiriş tek yönlüdür. Bu kirişte 5. düğümden 3. düğüme akış yoktur.



Şekil 11 (Floyd Algoritması İçin Bir Örnek)

Adım 0:

D_0 ve S_0 matrisleri ağ'ın başlangıç gösterimleridir. D_0 , d_{53} hariç simetriktir çünkü 5'inci düğümden 3'üncü düğüme akış verilmemektedir. İlk satır anahtar satır, ilk sütun ise anahtar sütun seçilir. Adım1'de, üçlü operasyonda v_1'in kullanılabileceği, yani d_{ij} değeri ∞ olan tek seçenek d_{23} ve d_{32}'dir

D_0

	v_1	v_2	v_3	v_4	v_5
v_1	-	3	10	∞	∞
v_2	3	-	∞	5	∞
v_3	10	∞	-	6	15
v_4	∞	5	6	-	4
v_5	∞	∞	∞	4	-

S_0

	v_1	v_2	v_3	v_4	v_5
v_1	-	2	3	4	5
v_2	1	-	3	4	5
v_3	1	2	-	4	5
v_4	1	2	3	-	5
v_5	1	2	3	4	-

Adım1:

Üçlü operasyon bu elemana uygulanır.

1) d_{23} elemanı yerine $d_{21}+d_{13}=3+10=13$ koyulur ve $s_{23}=1$ yapılır,
2) d_{32} elemanı yerine $d_{31}+d_{12}=10+3=13$ koyulur ve $s_{32}=1$ yapılır.

Bu değişiklikler D_1 ve S_1 matrislerinde yerlerini alırlar. Şimdi D_1 bağlantı matrisinde ikinci satır anahtar satır, ikinci sütun da anahtar sütun seçilir. Üçlü operasyon uygulanabilecek düğümler v_1, v_3 ve v_4'dür. Bu üç düğümden oluşan ikili düğüm kombinasyonlarından birbirine direkt bağlı olmayan tüm kombinasyonlar yani d_{ij} değeri ∞ olan ikili düğüm kombinasyonlarına, v_2 düğümünden geçmek üzere, üçlü operasyon uygulanacaktır. Burada oluşan ikili düğüm kombinasyonları (v_1,v_3), (v_1,v_4) ve (v_3,v_4)'dür. Algoritmanın bu adımında, bağlantı matrisinde, birbirine direkt bağlantı kurulamamış tek düğüm çifti (v_1,v_4)'dür. Bu düğüm çiftinde bağlantı matrisindeki direkt bağlantı $v_1 \rightarrow v_2 \rightarrow v_4$ ya da $v_4 \rightarrow v_2 \rightarrow v_1$ şeklinde kurulacaktır.

D_1

	v_1	v_2	v_3	v_4	v_5
v_1	-	3	10	∞	∞
v_2	3	-	13	5	∞
v_3	10	13	-	6	15
v_4	∞	5	6	-	4
v_5	∞	∞	∞	4	-

S_1

	v_1	v_2	v_3	v_4	v_5
v_1	-	2	3	4	5
v_2	1	-	1	4	5
v_3	1	1	-	4	5
v_4	1	2	3	-	5
v_5	1	2	3	4	-

Adım 2:

k=2 yapılır ve bu düğüm çiftine üçlü operasyon uygulanarak aralarındaki bağlantı kurulur.

$$d_{41}= d_{42}+d_{21}=5+3=8$$

$$d_{14}= d_{12}+d_{24}=3+5=8$$

Bu sonuçlar aşağıdaki D_2 matrisinde yerlerine yazılır. S_2 matrisi de bu sonuçlar doğrultusunda güncellenir.

D_2

	v_1	v_2	v_3	v_4	v_5
v_1	-	3	10	8	∞
v_2	3	-	13	5	∞
v_3	10	13	-	6	15
v_4	8	5	6	-	4
v_5	∞	∞	∞	4	-

S_2

	v_1	v_2	v_3	v_4	v_5
v_1	-	2	3	2	5
v_2	1	-	1	4	5
v_3	1	1	-	4	5
v_4	2	2	3	-	5
v_5	1	2	3	4	-

Adım 3:

k= 3 yapılır ve güncellemelere devam edilir. Şimdi anahtar satır ve sütun 3. satır ve sütun olarak belirlenir. v_1, v_2, v_4, v_5 düğümlerinden yine öyle düğüm çiftleri alınmalı ki bağlantı matrisindeki değerleri ∞ olsun. Böyle sadece iki tane düğüm çifti vardır. (v_1,v_5) ve (v_2,v_5). Bu düğümlerin birbirlerine geçişlerinde v_3 ara düğüm olarak kullanılır.

D_3

	v_1	v_2	v_3	v_4	v_5
v_1	-	3	10	8	25
v_2	3	-	13	5	28
v_3	10	13	-	6	15
v_4	8	5	6	-	4
v_5	∞	∞	∞	4	-

S_3

	v_1	v_2	v_3	v_4	v_5
v_1	-	2	3	2	3
v_2	1	-	1	4	3
v_3	1	1	-	4	5
v_4	2	2	3	-	5
v_5	1	2	3	4	-

Adım4:

k=4 yapılır ve güncellemelere devam edilir.

D_4

	v_1	v_2	v_3	v_4	v_5
v_1	-	3	10	8	12
v_2	3	-	11	5	9
v_3	10	11	-	6	10
v_4	8	5	6	-	4
v_5	12	9	10	4	-

S_4

	v_1	v_2	v_3	v_4	v_5
v_1	-	2	3	2	4
v_2	1	-	4	4	4
v_3	1	4	-	4	4
v_4	2	2	3	-	5
v_5	4	4	4	4	-

Adım 5:

k=5 yapılır. Ancak matrislerde başka geliştirme yapılamaz. Çünkü son matrisler olan D_4 ve S_4 matrisleri ağdaki herhangi iki tepe arasındaki en kısa rotayı belirlemek için gerekli bilgiyi içermektedirler. Örneğin 1'inci ve 2'inci düğüm arasındaki en kısa yol d_{ij}=12'dir. (v_i,v_j) rotasında s_{ij}=j ise aralarında direkt bağlantı vardır. Aksi takdirde i düğümünden j düğümüne en kısa yoldan gitmek için bir aracı düğüm kullanılmaktadır. Mesela s_{15}=4 ve s_{45}=5 olduğu için rota 1→4→5 olarak belirlenir. Burada henüz s_{14}≠4'dür ve (v_1,v_4) rotası henüz direkt bağlantılı değildir. Aracı tepelerin belirlenmesi gerekmektedir. s_{14}=2 ve s_{24}=4 olduğu için rota 1→4 değil 1→2→4'dür. Sonuçlar birleştirildiğinde optimum rotanın 1→2→4→5 olduğu belirlenir. Rotanın uzunluğu yani v_1'den v_5'e en kısa yol ise 12 mil'dir.

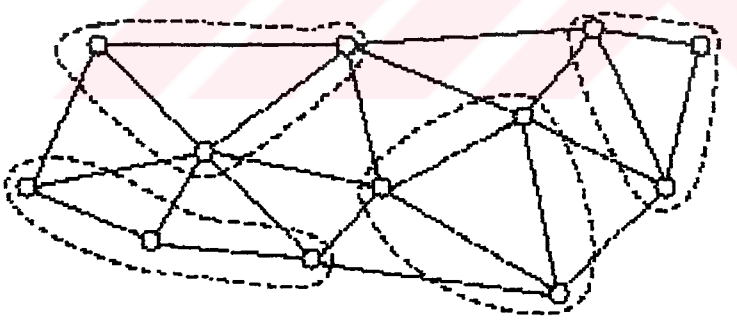
DÖRDÜNCÜ BÖLÜM

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

1. $U_{i=1}^{p} N_i = N$ ve $N_i \cap N_j = \phi$ i≠j ise. Bu bir düğümün birden fazla grup (işlemci) içersinde yer almadığı anlamına gelir.
2. W(i) ≈ 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.
3. 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. Herbir N_i çizgenin bir parçasıdır. Bir bisection (ikiye parçalama) 2-yollu parçalamadır. İkinci durumu sağlayan parçalamaya da "dengelenmiş parçalama" denir. Şekil 12'de 4-yollu dengelenmiş parçalanma çizge üzerinde gösterilmiştir.



Şekil 12 (4-yollu Dengelenmiş Parçalama)

Çizge parçalama problemi VLSI (very large scaled integration) yerleştirme ve rotalama alanlarında , ve de özellikle paralel programlamada kullanılır. Çizge parçalamaya olan ihtiyaç bir ısı iletkenliği probleminde örneklenebilir.

Bir ısı iletkenliği problemini iki boyutlu bir alanda nümerik olarak ele almak için problem farklı bir yaklaşım olan "sonlu eleman metodları" (finite element methods) ile çözülür. (Dündar, 1998, a). Bu metod iletken alanı parçalara bölerek çalışır. Bu parçalar kirişlerden ve düğümlerden oluşur. Bu iletken alanın parçalanması bir çizgeyle gösterilebilir.

Parçalardaki düğümlerin ısıları aşağıdaki doğrusal eşitlik sistemi çözülerek bulunabilir.

$$Ku=f$$

Burada u ve f herbir düğüm için tek elemanlı vektörlerdir ve K ise ağırlıklı bitişiklik matrisidir. İterasyonlu bir metod y vektörünü $Kx=y$ gibi bir eşitlik için çözer. Sistemde i düğümündeki y değerinin hesaplanabilmesi için, i düğümüne bağlanmış olan bütün düğümlerin x değerlerinin bilinmesi gereklidir.

Matris vektör çarpımının paralelize edilebilmesi için herbir düğüm bir işlemciye tayin edilir. Bu, i'inci düğümün x ve y değerlerini ve K'daki i'inci satırdaki sıfırdan farklı elemanların ayrı özel bir işlemciye toparlanması içindir. İşlemcilerdeki düğümlerin haritalanması, bütün işlemcilerin aynı hesaplama operasyonu sayısına denk düşecek şekilde yapılmalıdır. i düğümündeki y'nin değerini bulmak için i düğümünün derecesiyle oransal bir aritmetik operasyon miktarı yapılır.

Hesaplama yükünü dengelemenin yanında iletişim zamanını da azaltmak gerekir. i düğümündeki y'nin değerini bulmak için işlemci, i düğümünün komşularının x değerlerine ihtiyaç duyar. Herhangi bir komşuyu başka bir işlemciye geçirmenin ne kadar zaman alacağına etki eden faktör ise işlemciler arası geçiş kirişlerinin ağırlıkları toplamı olan "kesim büyüklüğü"dür. (Fjalström, 1998)

Daha açık olarak $Kx=y$ doğrusal denklem sisteminde K katsayılar matrisinin elemanlarından çoğunun sıfır olması halinde K matrisi ile elemanlarının hepsi sıfırdan farklı olan x sütun matrisinin çarpımında gereksiz çarpımları ortadan kaldırma K matrisinin boyutunun büyük olduğu sistemlerde önemli bir zaman tasarrufu sağlamaktadır. Bu problemin çözümünde K matrisi sıfırdan farklı elemanlarının yer

alacağı alt matrislere ayrılmakta, alt matrislerin herbiri x sütun matrisinin ilgili elemanları ile çarpılarak birer işlemcide çalıştırılmak yoluyla matris vektör çarpımı paralel hesaplama ile bulunmaktadır. (Dündar, Dündar, 1997)

Yukarıdaki örnekte görüldüğü gibi, çizge parçalama problemi sonlu eleman metodlarını etkili bir biçimde paralelize eden problemlere bir yaklaşımdır. Bu ve diğer sebeplerden dolayı araştırmacılar çizge parçalama için birçok metodlar geliştirmişlerdir. Bu metodların bazıları birinci bölümde verilmiştir.

Son zamanlarda araştırmacılar çizge parçalama için paralel algoritmalar geliştirmeye başladılar. Bunun için pek çok sebep vardır. Bazı ardışık parçalama yapan algoritmalar yüksek kaliteli parçalamalar 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 uygulamak mümkün olmayabilir ya da etkisiz olabilir. Bu tür durumlarda paralel hesaplamaya gerek duyulur. (Fjalström, 1998)

4.1 Kiriş Ve Düğüm Ayrıştırıcıları

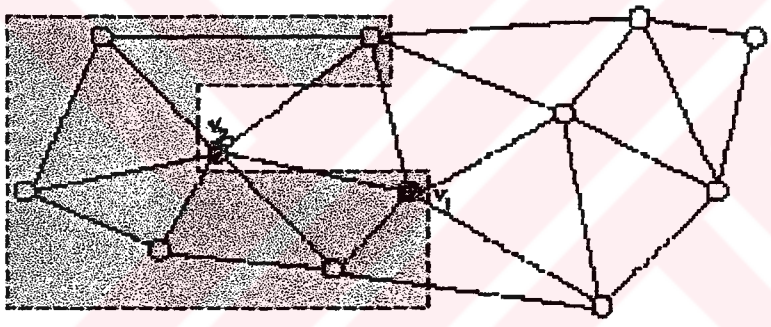
G=(N,E) çizgesini ikiye parçalamak iki yolla yapılabilir. Birincisi E'nin en küçük altkümesi olan E_s'yi E'den ayırarak, G çizgesini, $N_1 \cup N_2 = N$ olmak üzere herbiri N_1 ve N_2 düğümlerine sahip G_1 ve G_2 alt çizgesine bölmektir. N_1 ve N_2 eşit büyüklüktedir. E_s içindeki kirişler N_1 içindeki 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 $N = N_1 \cup N_s \cup N_2$'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) C.S.: "Lectures Graph Partitioning I,II,." HTTP, CS267)

4.2 Kernighan Lin Algoritması

Araştırmacılar çizge parçalama için birçok ardışık algoritmalar geliştirmişlerdir. Bu algoritmalardan Kernighan-Lin algoritması, çizge parçalama için en temel algoritmalardan biri olduğundan dolayı ve beşinci bölümde sunulan en kısa yol problemi algoritması içinde kullanıldığından dolayı, bu bölümde ayrıntılı olarak ele alınmıştır. Çizge parçalama algoritmaları istedikleri girdi tipine göre farklı kategorilere ayrılırlar.

Örneğin, bir bölgesel gelişim algoritması, G'den bir parçayı girdi olarak alır ve kesim büyüklüğünü bölgesel tarama metoduyla azaltmaya çalışır. Diğer bir teknik de birçok rastsal başlangıç parçası türetmek ve algoritmayı her birine uygulayarak en iyi kesim büyüklüğü elde edilen parçayı kullanmaktır. Eğer bölgesel gelişim metodu ikiye parçalanmış bir çizgesi girdi olarak alıyorsa, p ayrışım gerçekleşene kadar her bir parça ardışık olarak ikiye parçalanır. (Fjalström, P., 1998)



Şekil 13 (KL Algoritması Örneği)

Şekil 13, ikiye parçalanmış bir çizgeye örnektir. Koyu alan içindeki düğümler N_1 kümesinin, diğerleri de N_2 kümesinin içerisindedir.

Kernighan-Lin ilk çizge parçalama metodlarından birini tasarlamışlardır ve birçok bölgesel gelişim metodu Kernighan-Lin metodunun bir varyasyonudur.

Öncelikle çizge rassal 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.) Herbir $v \in N$ için şu tanımlanır;

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

$$\text{ext(v)} = \sum_{(v,u)\in E, P(v)\neq P(u)} w(v,u) \qquad (4.1)$$

int(v), v düğümünü kendi işlemcisi içindeki düğümlere bağlayan kirişlerin ağırlıkları toplamıdır. ext(v) ise, v düğümünü diğer işlemcilere bağlayan kirişlerin ağırlıkları toplamıdır. Şekil 13'de gösterilen çizgede (4.1) formülasyonundan $int(v_1)=2$, $ext(v_1)=3$, $g(v_1)=1$, $int(v_2)=0$, $ext(v_2)=6$ olarak görülmektedir. Kesim büyüklüğü=11'dir. (Düğüm ve kiriş ağırlıkları birim olarak varsayılmıştır.)

$P(v)$, 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 (4.2) formülüyle hesaplanır.

$$g(v) = ext(v) - int(v) \qquad (4.2)$$

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;

$$g(v_1,v_2) = \begin{cases} g(v_1)+g(v_2)-2w(v_1,v_2) & eger \ (v_1,v_2)\in E \\ g(v_1)+g(v_2) & d.d. \end{cases} \qquad (4.3)$$

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.

Şekil 13'de deki çizgede bu formülasyon yardımıyla, ele alınan v_1 ve v_2 düğümleri yer değiştirilecek olursa bu değişim sonucundaki kesim kirişleri ağırlıkları toplamındaki kazanç (4.3) formülünden $g(v_1,v_2)$=5 olarak hesaplanır.

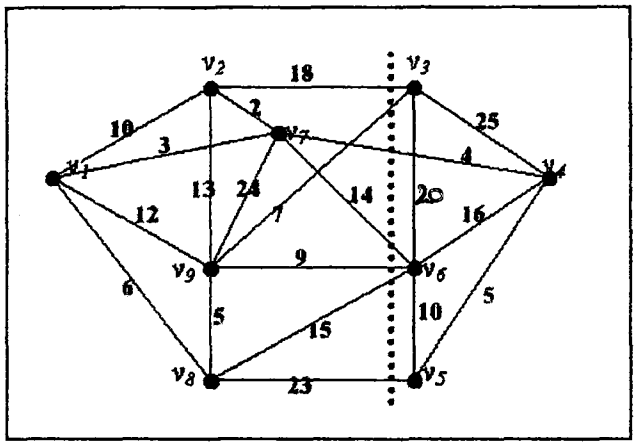
Kernighan-Lin algoritmasının bir iterasyonu şöyledir. İterasayona girdi dengelenmiş (düğüm ağırlıkları eşit) olan $\{N_1,N_2\}$parçalarıdır. İlk olarak, düğümlerin tekrar adlandırılabilmesi için bütün düğümlerin işaretleri kaldırılır. Sonra şu prosedür n kere uygulanır (n = $\min(|N_1|,|N_2|)$).

$g(v_1,v_2)$'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. Bu algoritma için Borland Delphi'de bir program yazılmıştır. Bu program Ek1'de verilmiştir.

Kernighan-Lin metodunun tek bir iterasyonu en fazla $O(|N^3|)$ kadar zamana ihtiyaç duyar (Fjalström, 1998). Bu, algoritmanın karmaşıklığıdır. Yani çizgenin tam çizge (her bir düğümü bütün diğer düğümlere bağlı olan çizge) olması durumunda yapılabilecek maksimum işlem sayısıdır.

4.2.1 KL Algoritması İçin Bir Örnek



Şekil 14 (KL Algoritması Örneği)

Şekil 14'deki çizgeyi Kernighan Lin algoritması ile ikiye parçalama problemini ele alınsın. İlk olarak Dengelenmiş bir şekilde çizge tesadüfi olarak ikiye bölünür. $N_1=\{v_1,v_2,v_7,v_8,v_9\}$ $N_2=\{v_3,v_4,v_5,v_6\}$ kümeleri Şekil 14'de görülmektedir. Başlangıç kesim kirişleri ağırlıkları toplamının 90 olduğu görülmektedir. n=4 defa iterasyon yapılacağı ise minimum eleman sayısına sahip N_2 kümesinin eleman sayısından belirlenir.

İterasyon 1

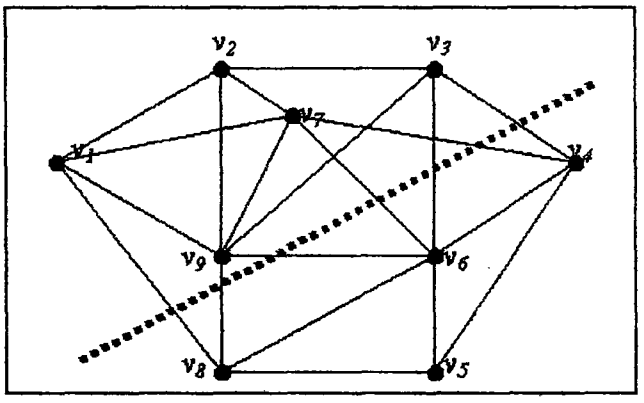
$g(v_1)=-31$ $g(v_2)=-7$ $g(v_3)=-20$ $g(v_4)=-42$ $g(v_5)=8$ $g(v_6)=-8$ $g(v_7)=-11$
$g(v_8)=27$ $g(v_9)=-38$

Burada örneğin $g(v_8)=23+15-6-5=27$ olarak bulunur. Yani v_8 düğümünü diğer işlemciye aktarırsak kesim büyüklüğü 27 br azalır.

$g(v_1,v_3)=-51$ $g(v_1,v_4)=-73$ $g(v_1,v_5)=-23$ $g(v_1,v_6)=-39$
$g(v_2,v_3)=-63$ $g(v_2,v_4)=-48$ $g(v_2,v_5)=+1$ $g(v_2,v_6)=-15$
$g(v_3,v_7)=-31$ $g(v_3,v_8)=+7$ $g(v_3,v_9)=-72$
$g(v_4,v_7)=-70$ $g(v_4,v_8)=-15$ $g(v_4,v_9)=-80$
$g(v_5,v_7)=-3$ $g(v_5,v_8)=-11$ $g(v_5,v_9)=-30$
$g(v_6,v_7)=-47$ $g(v_6,v_8)=+5$ $g(v_6,v_9)=-48$

Burada örneğin $g(v_6,v_9)=-8-38-2*9=-48$ olarak bulunur. v_6 ve v_9'u yer değiştirirsek kesim büyüklüğünde 48 br artar.

Bu ikili kazanç değerleri arasında yer değiştirilmelerinden en fazla kazanç sağlanan iki düğüm çifti (v_3,v_8) çiftidir. Bu düğümler işlemciler arasında yer değiştirilir. Yani kümelerimiz $N_1=\{v_1,v_2,v_7,v_3,v_9\}$ $N_2=\{v_8,v_4,v_5,v_6\}$ şeklinde olur. Bu düğüm çifti bir daha işleme tabi tutulmamak üzere işaretlenir. Artık yeni kesim kirişleri ağırlıkları toplamı 83'e inmiştir. Çizgedeki yeni kiriş ayrıştırıcısı şekil 15'deki gibi oluşur.



Şekil 15 (KL Algoritması Örneği İterasyon1)

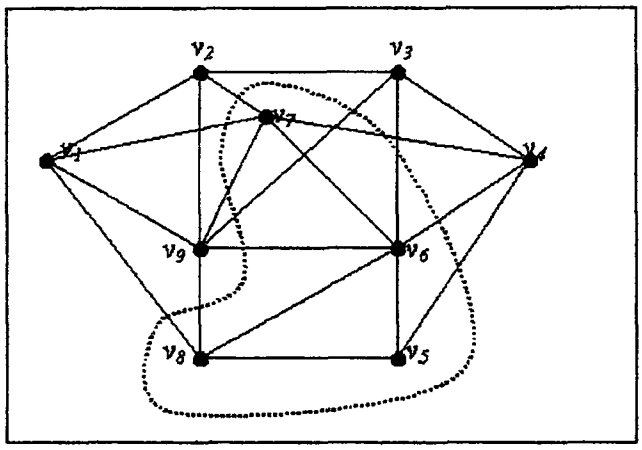
İterasyon 2

$g(v_1)=-19$ $g(v_2)=-43$ $g(v_4)=+8$ $g(v_5)=-38$ $g(v_6)=+2$ $g(v_7)=-11$ $g(v_9)=-42$

$g(v_1,v_4)=-11$ $g(v_1,v_5)=-57$ $g(v_1,v_6)=-20$
$g(v_2,v_4)=-35$ $g(v_2,v_5)=-81$ $g(v_2,v_6)=-41$
$g(v_4,v_7)=-11$ $g(v_4,v_9)=-36$
$g(v_5,v_7)=-49$ $g(v_5,v_9)=-80$
$g(v_6,v_7)=-37$ $g(v_6,v_9)=-65$
MAX=-11 (v_4,v_7) ya da (v_1,v_4) seçilebilir.

En fazla kazancı sağlayan (v_4,v_7) düğüm çifti seçilir ve yer değiştirilir. Bu düğüm çifti sonraki iterasyonlarda tekrar işleme tabi tutulmaması için işaretlenir.

Artık yeni kesim kirişleri ağırlıkları toplamı 94'e yükselmiştir. Çizgedeki yeni kiriş ayrıştırıcısı şekil 16'daki gibi oluşur.



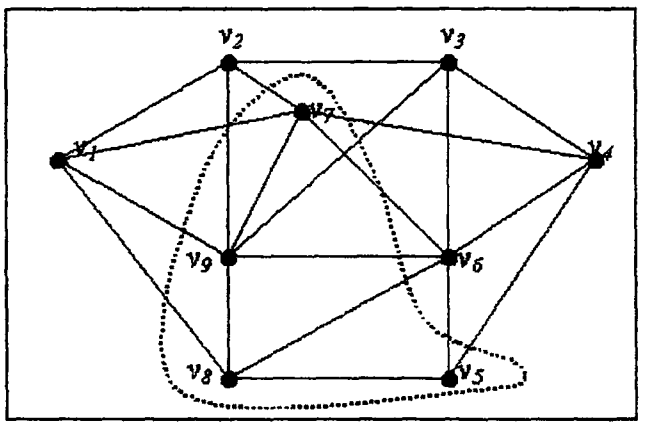
Şekil 16 (KL Algoritması Ömeği İterasyon2)

İterasyon 3

$g(v_1)$=-13 $g(v_2)$=-39 $g(v_5)$=-28 $g(v_6)$=+6 $g(v_9)$=+6

$g(v_1, v_5)$=-41 $g(v_1, v_6)$=-7
$g(v_2, v_5)$=-67 $g(v_2, v_6)$=-33
$g(v_5, v_9)$=-22
$g(v_6, v_9)$=-6

MAX(g)=-6 (v_6, v_9) düğüm çifti seçilir, yer değiştirilir ve işaretlenir. Yeni kesim kirişleri ağırlıkları toplamı artık 100'e çıkmıştır. Çizgedeki yeni kiriş ayrıştırıcısı şekil 17'deki gibi oluşur.



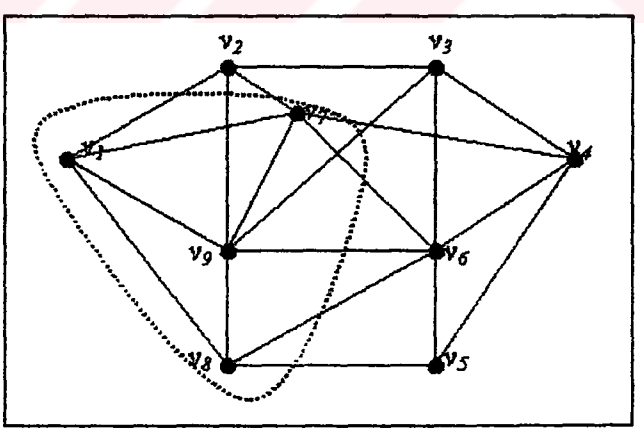
Şekil 17 (KL Algoritması Örneği İterasyon3)

İterasyon 4

$g(v_1)=11$ $g(v_2)=-13$ $g(v_5)=-8$

$g(v_1,v_5)=3$
$g(v_2,v_5)=-21$

MAX(g)=+3 (v_1,v_5) düğüm çifti yer değiştirilir. Bu son iterasyondur. Yeni kesim kirişleri ağırlıkları toplamı 97 olarak hesaplanır. Çizgedeki yeni kiriş ayrıştırıcısı şekil 18'deki gibi oluşur.



Şekil 18 (KL Algoritması Örneği İterasyon4)

Bütün bu iterasyonlar içersinde en az kesim kirişleri ağırlıkları toplamı elde edilen iterasyona kadar o iterasyonda dahil olmak üzere en fazla kazancı sağlayan tepeler yer değiştirilir. Diğerleri yer değiştirilmez. Bu problemde sonucu veren iterasyon birinci iterasyondur. Böylece kesim kirişleri ağırlıkları toplamı en az olacak şekilde bu çizgenin ikiye parçalanması probleminin sonucu Şekil 15'deki gibi olacaktır. Kesim kirişleri ağırlıkları toplamı 83'dür. İterasyonlar sonucunda bu noktaya ulaşılmıştır.

BEŞİNCİ BÖLÜM

EN KISA YOL PROBLEMİNİN ÇİZGE PARÇALANIŞI İLE ÇÖZÜMÜ

Bu bölümde düğümleri sayısı büyük ($|N|$>1000) bir zincir çizgede en kısa yol probleminin çizge parçalanışı kullanılarak çözümü bulunmaya çalışılmıştır.

5.1. Geliştirilen Algoritma

Tanım 5.1: *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 5.2: G bir basit yönlendirilmiş çizge olsun. G'nin A bitişiklik matrisi

$$A = \begin{bmatrix} A_1 & B_1 & 0 & . & . & 0 \\ 0 & A_2 & B_2 & 0 & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & . & . & . & B_{m-1} \\ 0 & 0 & . & . & . & A_m \end{bmatrix}$$

biçiminde 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. (Dündar, Beşer, Dündar, 2000)

Eğer G çizgesinin kirişleri üzerinde ağırlıklar (maliyetler) işaretlenmişse G'nin C maliyetler matrisi:

$$C=\begin{bmatrix} D_1 & E_1 & . & . & . \\ & D_2 & E_2 & . & . \\ . & . & . & . & . \\ . & . & . & . & E_{m-1} \\ . & . & . & . & D_m \end{bmatrix}$$

biçiminde gösterilebilir. 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.

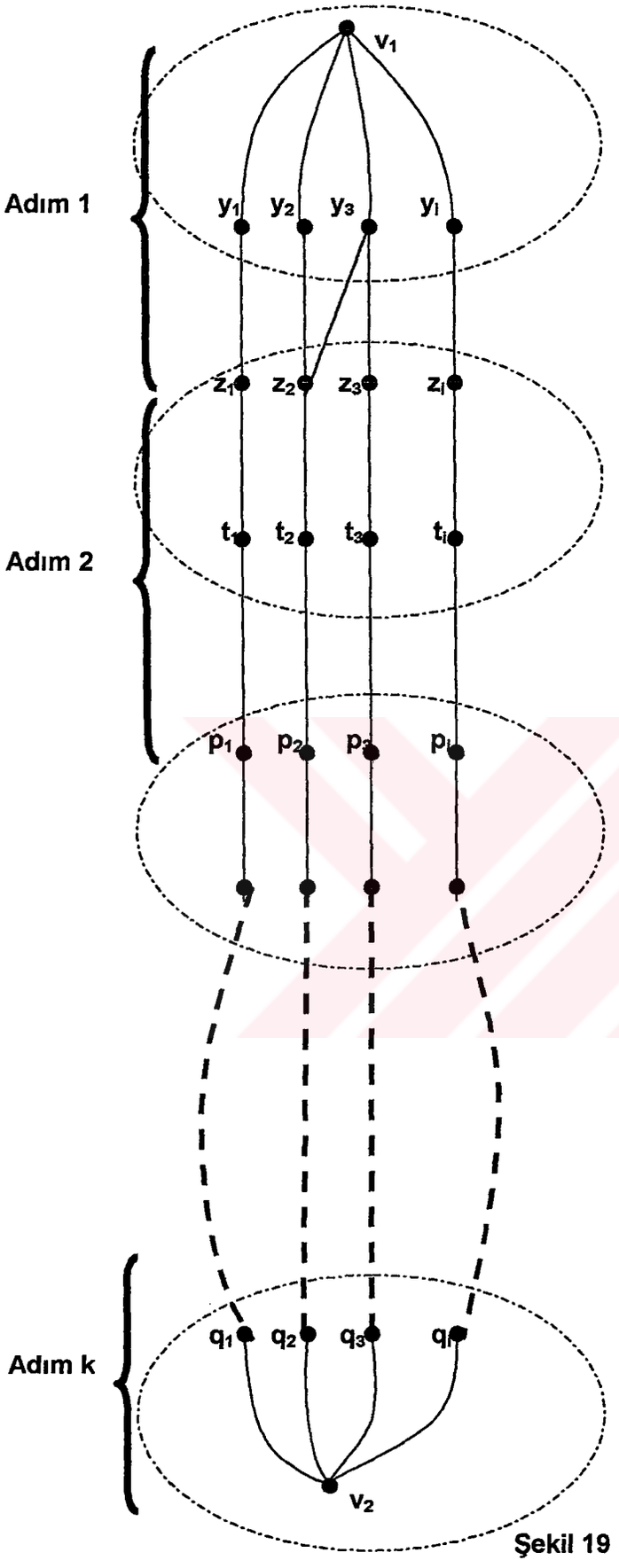
Verilen ağ'ın modeli olan çizge; 2. bölümde verilen KL algoritması ile k adet parçaya ayrılarak bir zincir çizge haline getirildikten sonra; G_1 çizgesindeki bir v_1 düğümünden başlayarak sırasıyla G_2, G_3,...,G_{k-1} alt çizgelerinden geçip G_k çizgesinin bir v_2 tepesinde biten en kısa yolu bulma bu çalışmanın temelidir. Algoritmanın adımları Şekil19 üzerinde anlatılmıştır.

Adım1: 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 herbir bulunan (v_1,y_i) yoluna eklenerek tüm (v_1,z_i) en kısa yolları bulunur.

Adım2: 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 herbir 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.

Adımk+1: v_1'den v_2'e tüm bağlantılı yollar arasındaki $min[(v_1,y_i)+(y_i,z_i)+(z_i,t_i)+(t_i,p_i)+(p_i,...)+(...,q_i)+(q_i,v_2)]$ $(1\le i\le n/k)$ yolu hesaplanır. Bu yol bütün işlemcilerden sırasıyla geçmek şartı ile en kısa yolu verir.



Şekil 19 (Geliştirilen Algoritma)

5.2 Geliştirilen Algoritma İçin Bir Örnek

	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. Hazırladığımız KL algoritması programı çıktıları EK2'de verilmiştir. Çizge 4 parçaya bölündükten sonra bağlantı matrisi (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

5.1'de verilen tüm adımlar ard arda bu çizgeye uygulanırsa şu sonuçlar elde edilir. Hesaplanan yolların uzaklıkları parantez içinde verilmiştir.

Adım 1:

$v_2 \rightarrow v_3$ (18)

$v_2 \rightarrow v_3 \rightarrow v_4$ (43)

Bu yollar birinci işlemci olan N_1 içersindeki Djkstra algoritması ile hesaplanan en kısa yollardır.

$v_2 \rightarrow v_3 \rightarrow v_6$ (43)

$v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_6$ (59)

Bu yollar N_1 içersindeki Djkstra algoritması ile hesaplanan en kısa yollara N_2 işlemcisiyle N_1 işlemcisini birbirine bağlayan kesim kirişlerinin eklenmesiyle elde edilmiş yollardır.

Adım 2:

$v_6 \rightarrow v_8$ (15)

Bu yol N_2 içersindeki Djkstra algoritması ile hesaplanan tek en kısa yoldur.

$v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_7$ (73)

$v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_8 \rightarrow v_1$ (63)

$v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_6 \rightarrow v_8 \rightarrow v_1$ (80)

N_2 ve N_3 arasındaki kesim kirişlerinin önceki hesaplanan yol kombinasyonlarına eklenmesiyle bu yollar bulunur.

Adım3:

$v_1 \rightarrow v_7$ (3)

$v_9 \rightarrow v_7$ (24)

$v_1 \rightarrow v_9 \rightarrow v_7$ (37)

Bu yollar N_3 içerisindeki bütün düğümler arasındaki en kısa yollardır. Bu en kısa yollara N_3 ve N_4 arasındaki tüm kesim kirişlerinin eklenmesiyle aşağıdaki yollar bulunur.

$v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5$ (78)

$v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_8 \rightarrow v_1 \rightarrow v_9 \rightarrow v_7 \rightarrow v_5$ (105)

$v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_6 \rightarrow v_8 \rightarrow v_1 \rightarrow v_9 \rightarrow v_7 \rightarrow v_5$ (119)

Adım4:

$v_5 \rightarrow v_{13}$ (8)

Bu yol N_2 içersindeki Djkstra algoritması ile hesaplanan tek en kısa yoldur. Daha önceki yol kombinasyonlarına bu yolun eklenmesiyle v_{13}'e ulaşan tüm yollar aşağıda gösterilmiştir.

$v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5 \rightarrow v_{13}$ (86)

$v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_8 \rightarrow v_1 \rightarrow v_9 \rightarrow v_7 \rightarrow v_5 \rightarrow v_{13}$ (113)

$v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_6 \rightarrow v_8 \rightarrow v_1 \rightarrow v_9 \rightarrow v_7 \rightarrow v_5 \rightarrow v_{13}$ (127)

Sonuca ulaşılan tüm yollar içinden en kısa yol $v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5 \rightarrow v_{13}$ yoludur. Bu yoldaki uzaklık ise 86 olarak hesaplanır.

5.3. 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 (gevşek, seyrek) 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. Adım sayısının polinomla ifade edildiği algoritmalara polinomsal algoritmalar denir.

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. (Fjalström, 1998). Sonuçta 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

1. Bakoğlu, H., (1999), "**Doğrusal Programlama**" Ege Üniversitesi, Fen Fakültesi

2. Balas, E. (1989). "The Assimetric Assignment Problem and Some New Facets of The Travelling Salesman Polytope on a Directed Graph". **Siam Journal Discrete Math 2**

3. Barnard, S.T., Simon, H.D. (1993). "A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems". **In Proc. 6th SIAM Conf. Parallel Processing for Scientific Computing**, (711-718)
4. Berger, M.J., Bokhari, S.H. (1987). "A partitioning strategy for non-uniform problems across multiprocessors". **IEEE Transactions on Computers**, C-36:570-580.

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

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

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

8. Christofides, N., (1986). "**Graph Theory An Algorithmic Approach**", Academic Press, London

9. Dündar, P., Dündar, S., (1997), "Matris Vektör Çarpımının Graf Parçalama İle Gerçekleştirimi", **Matematik Sempozyumu, Sarımsaklı'da Matematik Günleri**, 05-08 Haziran, Ayvalık

10. Dündar, S., (a)(1998), "İki Boyutlu Isı Transfer Probleminin Çözümüne Yeni Bir Yaklaşım", **E.Ü. Güneş Enerjisi Enstitüsü Dergisi** Cilt 3 Sayı 1

11. Dündar, S., (b)(1998). "Çizge Teorisi ve Uygulamaları", **M.Ü.İ.İ.B.F. Dergisi**, cilt:XIV, Sayı:2, 113-121.

12. Dündar, S., Beşer, M.K., Dündar, P., (2000), "En Kısa Yol Probleminin Çizge Parçalanışı İle Çözümü", **YA/EM2000 XXI. Ulusal Kongresi**, K.K.T.C.

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

14. Fjalström, P. (1998). "Algorithms for graph partitioning",1998, **Linköping University Electronic Press**

15. 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.

16. Kumar, V., Karypis, G., (1994), "**Introduction to Parallel Computing**", The Benjamin Cumming Pub. Comp., California

17. Lovazs, L., Winkler, P., (1996), "A Note on the Last New Vertex Visited Random Walk" **Journal Of Graph Theory**, vol.17, no.5

18. 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.

19. Nakhimovski, I. (1997). "Bucket-based modification of the parallel recursive coordinate bisection algorithm". **Linköping Electronic Articles in Computer and Information Science,** 2(15).

20. Öztürk, A., (1994), "**Yöneylem Araştırması**", Uludağ Üniversitesi, İktisadi ve İdari Bilimler Fakültesi.

21. 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.

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

23. 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.

24. Taha, A.H. (1997). **"Operations Research An Introduction"**, Printice-Hall, Fayetteville.

25. Tulunay, Y. (1982), **İşletme Matematiği**, İstanbul Üniversitesi, İşletme Fakültesi, Yayın no:233

26. 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.

EKLER

EK1

```
program kemal1;
uses
  Forms,
  MMAIN in 'MMAIN.pas' {MAINFORM},
  Unit1 in 'Unit1.pas' {ParcaSor};
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TMAINFORM, MAINFORM);
  Application.CreateForm(TParcaSor, ParcaSor);
  Application.Run;
end.
////////////////////////////////////////////////////////////////
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls;
type
  TParcaSor = class(TForm)
    RadioGroup1: TRadioGroup;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  ParcaSor: TParcaSor;
implementation
```

```
{$R *.DFM}
end.
////////////////////////////////////////////////////////////////////////////

unit MMAIN;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, Mask, Menus,Grids, ComCtrls, ExtCtrls,
  ToolWin, Spin,ImgList,Unit1;
type
  TMAINFORM = class(TForm)
    MainMenu1: TMainMenu;
    DOSYA1: TMenuItem;
    ToolBar1: TToolBar;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolButton4: TToolButton;
    ToolButton5: TToolButton;
    ToolButton6: TToolButton;
    Panel1: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    StringGrid1: TStringGrid;
    ToolbarImages: TImageList;
    Panel2: TPanel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Panel5: TPanel;
    Panel6: TPanel;
    AC: TMenuItem;
```

```
    KAYDET: TMenuItem;
    FARKLIKAYDET: TMenuItem;
    YENI: TMenuItem;
    N1: TMenuItem;
    DEGISIM: TMenuItem;
    IKI1: TMenuItem;
    PARCALA: TMenuItem;
    N4: TMenuItem;
    ToolButton7: TToolButton;
    ToolButton8: TToolButton;
    ToolButton9: TToolButton;
    Memos: TMemo;
    Splitter1: TSplitter;
    EDITOR: TMenuItem;
    N2: TMenuItem;
    Kapat1: TMenuItem;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    procedure FormCreate(Sender: TObject);
    PROCEDURE SHOWLOCATER(CONST VALUE:BOOLEAN);
    PROCEDURE OPENFILE(Sender: TObject);
    PROCEDURE SAVEFILE(Sender: TObject);
    PROCEDURE CRNEWFILE(Sender: TObject);
    PROCEDURE OPENEDITMODE(Sender: TObject);
    PROCEDURE CLOSEEDITMODE(Sender: TObject);
    procedure YerGsterici1Click(Sender: TObject);
    procedure ToolButton6Click(Sender: TObject);
    procedure FARKLIKAYDETClick(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure StringGrid1SetEditText(Sender: TObject; ACol, ARow: Integer;const
Value: String);
    procedure GETTOGRID1Click(Sender: TObject);
    procedure ToolButton9Click(Sender: TObject);
    procedure ToolButton5Click(Sender: TObject);
```

```
    procedure EDITORClick(Sender: TObject);
    procedure DEGISIMClick(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure Edit2Change(Sender: TObject);
    procedure StringGrid1SelectCell(Sender: TObject; ACol, ARow: Integer;
      var CanSelect: Boolean);
    procedure Edit3Change(Sender: TObject);
    procedure PARCALAClick(Sender: TObject);
  private
  public
  PROCEDURE DIVIDE;
  PROCEDURE SETDUGUMSAYISI(CONST D:INTEGER);
  PROCEDURE SETMODIFIED(VALUE:BOOLEAN);
  PROCEDURE CLEAR;
  PROCEDURE LOADFILE(CONST FIL:STRING);
  PROCEDURE RECORDFILE(CONST FIL:STRING);
  PROCEDURE GETGAINS(CONST LOW,DIVIDE,HIGH:INTEGER);
  PROCEDURE GETEXGAINS(CONST LOW,DIVIDE,HIGH:INTEGER);
  FUNCTION  CHECK(CONST A:INTEGER):BOOLEAN;
  PROCEDURE EXCHANGE(VAR A,B:INTEGER);
  PROCEDURE OPERATIONMAXSUM(CONST COUNT:INTEGER);
  end;
var
  MAINFORM: TMAINFORM;
  EDITING:BOOLEAN;
  TMP:STRING;
  DUGS:ARRAY[1..1000]OF INTEGER;
  VALUES:ARRAY[1..1000,1..1000]OF INTEGER;
  BORDERSH:ARRAY[1..512]OF INTEGER;
  BORDERSL:ARRAY[1..512]OF INTEGER;
  GAINS:ARRAY[1..1000]OF INTEGER;
  EXCHI,EXCHJ,SUMOFEXCH:ARRAY[1..500]OF INTEGER;
  WORKINGFILE:STRING;
  NEWFILE,MODIFIED:BOOLEAN;
  DUGUMSAYISI:INTEGER=0;
```

```
  PARCASAYISI:INTEGER=0;
  CONT1:BOOLEAN=FALSE;
  CONT3:BOOLEAN=FALSE;

  CON2:BOOLEAN=FALSE;
  ITERATION1,ITERATION2,MAXSUMAT:INTEGER;
implementation
{$R *.DFM}
{GRAPH PARÇALAMA ALGORITMASI}
PROCEDURE TMAINFORM.DIVIDE;
      TYPE
         PMBORDERS=^TMBORDERS;
         TMBORDERS=RECORD
                NEXT:PMBORDERS;
                PREV:PMBORDERS;
                L:INTEGER;
                H:INTEGER;
                END;
     VAR
          BORDERSENTRY,P11:PMBORDERS;
          BORDERSLEAVE:PMBORDERS;
          BORDERSCOUNT:INTEGER;
          I11,III,JJJ:INTEGER;
          S:STRING;
          PROCEDURE CREATEBORDERS(CONST N,D:INTEGER);
           VAR   P1,P2:PMBORDERS;SIZE,I:INTEGER;
           BEGIN
//               MEMOS.LINES.ADD('');
//               MEMOS.LINES.ADD('CREATE BORDERS FOR '+ INTTOSTR(N)+'
PARTS & '+INTTOSTR(D)+' DUGUMS!');
                SIZE:=SIZEOF(TMBORDERS);
                GETMEM(P1,SIZE);
                P1^.PREV:=NIL;
                P1^.L:=1;
                P1^.H:=D;
```

```
                BORDERSENTRY:=P1;
                FOR I:=2 TO N DO
                BEGIN
                 GETMEM(P2,SIZE);
                 P1^.NEXT:=P2;
                 P2^.PREV:=P1;
                 P1:=P2;
                END;
                P1^.NEXT:=NIL;
                BORDERSLEAVE:=P1;
                BORDERSCOUNT:=N;
//               MEMOS.LINES.ADD('CREATE BORDERS DONE COOL');
          END;
          PROCEDURE FREEBORDERS;
           VAR P1,P2:PMBORDERS;I:INTEGER;
          BEGIN
//          MEMOS.LINES.ADD('');
//          MEMOS.LINES.ADD('FREE UP '+ INTTOSTR(BORDERSCOUNT)+'
BORDERS!');
           P1:=BORDERSENTRY;
           FOR I:=1 TO BORDERSCOUNT DO
           BEGIN
            P2:=P1^.NEXT;
            FREEMEM(P1);
            P1:=P2;
           END;
           BORDERSENTRY:=NIL;
           BORDERSLEAVE:=NIL;
//          MEMOS.LINES.ADD('FREE UP  BORDERS DONE COOL');
          END;
 PROCEDURE SETBORDERS(CONST N,D:INTEGER);
 VAR
  LAST,P1,P2:PMBORDERS;
  J,DL,N2:INTEGER;
 BEGIN
```

```
//      MEMOS.LINES.ADD('');
//      MEMOS.LINES.ADD('SET BORDERS FOR '+ INTTOSTR(N)+' PARTS &
'+INTTOSTR(D)+' DUGUMS!');

      ITERATION1:=0;
      J:=N;
      N2:=0;
      WHILE J>1 DO
      BEGIN
       J:=J DIV 2;
       N2:=N2+1;
      END;
      LAST:=BORDERSLEAVE;
      BORDERSCOUNT:=N;
      DL:=1;
      FOR ITERATION1:=1 to N2 do
       begin
         P1:=BORDERSENTRY;
         for j:=1 to DL do
         begin
           P2:=LAST;LAST:=P2^.PREV;
           P2^.H:=P1^.H;
           P1^.H:=((P1^.H-P1^.L+1) DIV 2)-1+P1^.L;
           P2^.L:=P1^.H+1;
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD('ITERATION '+INTTOSTR(ITERATION1));
//memos.lines.add('ITERATE : '+'[ '+inttostr(p1^.l)+' , '+inttostr(p1^.h)+' ]'+' & '+'[
'+','+inttostr(p1^.h+1)+' , '+inttostr(p2^.h)+' ]');
MEMOS.LINES.ADD('SUB ITERATION COUNT: '+INTTOSTR(P1^.H-P1^.L+1));
MEMOS.LINES.ADD('----------');
FOR ITERATION2:=1 TO (P1^.H-P1^.L+1) DO
BEGIN
MEMOS.LINES.ADD('');
memos.lines.add('      ITERATION :
'+INTTOSTR(ITERATION1)+'.'+INTTOSTR(ITERATION2));
```

```
MEMOS.LINES.ADD('      ----------------------------------------------------');

GETGAINS(P1^.L,P1^.H,P2^.H);
GETEXGAINS(P1^.L,P1^.H,P2^.H);
MEMOS.LINES.ADD('');
memos.lines.add('      ITERATION :
'+INTTOSTR(ITERATION1)+'.'+INTTOSTR(ITERATION2)+' DONE!');
END;
OPERATIONMAXSUM(P1^.H-P1^.L+1);
MEMOS.LINES.ADD('');
memos.lines.add('      ITERATION : '+INTTOSTR(ITERATION1)+' DONE');
MEMOS.LINES.ADD('');
            P2^.NEXT:=P1^.NEXT;
            P1^.NEXT:=P2;
            P1:=P2^.NEXT;
          end;
        DL:=DL*2;
//memos.lines.add('SET BORDERS DONE COOLL');
        end;
END;
BEGIN
MEMOS.ENABLED:=FALSE;
MEMOS.VISIBLE:=FALSE;
MEMOS.LINES.CLEAR;
CREATEBORDERS(PARCASAYISI,DUGUMSAYISI);
SETBORDERS(PARCASAYISI,DUGUMSAYISI);
P11:=BORDERSENTRY;
FOR I11:=1 TO BORDERSCOUNT DO BEGIN
                        BORDERSH[I11]:=P11^.H;
                        BORDERSL[I11]:=P11^.L;
                        P11:=P11^.NEXT;
                       END;
FREEBORDERS;
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD('');
```

```
MEMOS.LINES.ADD('_____ FOUND GROUPS :');
FOR III:=1 TO PARCASAYISI DO
BEGIN
S:='  N'+INTTOSTR(III)+' = { ';
//S:='[ '+INTTOSTR( BORDERSL[III])+'..'+INTTOSTR(BORDERSH[III])+' ] : '+'
N'+INTTOSTR(III)+' = { ';
FOR JJJ:=BORDERSL[III] TO BORDERSH[III] DO S:=S+'
V'+INTTOSTR(DUGS[JJJ])+' ,';
DELETE (S,LENGTH(S),1);
S:=S+' }';
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD(S);
END;
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD('CALCULATION COMPLETED.....');
CLEAR;
MEMOS.ENABLED:=TRUE;
MEMOS.VISIBLE:=TRUE;
END;
PROCEDURE TMAINFORM.GETEXGAINS(CONST LOW,DIVIDE,HIGH:INTEGER);
VAR
I,J,MAX,EXGAIN,MAXI,MAXJ,TMP,L,START1,START2:INTEGER;S:STRING;EXISTS:
BOOLEAN;
BEGIN
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD('      GETTING EXGAINS...');
//MEMOS.LINES.ADD('      GET EXGAINS FOR
:'+inttostr(LOW)+','+inttostr(DIVIDE)+','+inttostr(HIGH));
MEMOS.LINES.ADD('');
FOR I:=1 TO DIVIDE DO BEGIN IF ( NOT CHECK(I)) THEN BEGIN
START1:=I;BREAK;END;END;
FOR I:=DIVIDE+1 TO HIGH DO BEGIN IF ( NOT CHECK(I)) THEN BEGIN
START2:=I;BREAK;END;END;
EXGAIN:=GAINS[START1]+GAINS[START2]-2*VALUES[START1,START2];
MAXI:=START1;
```

```
MAXJ:=START2;
MAX:=EXGAIN;
S:='      EXGAIN ( ' +        INTTOSTR(START1) +        ' , ' +
INTTOSTR(START2) +        ' ) : ' +        INTTOSTR(GAINS[START1]) +        '+' +
INTTOSTR(GAINS[START2]) +        ' - 2 * ' +
INTTOSTR(VALUES[START1,START2]) +        ' = ' +        INTTOSTR(EXGAIN);
S:=S+'      <------NEW MAX';MEMOS.LINES.ADD(S);
FOR I:=START1 TO DIVIDE DO
BEGIN
      IF CHECK(I) THEN CONTINUE;
      FOR J:=START2 TO HIGH DO
      BEGIN
      IF CHECK(J) THEN CONTINUE;
       EXGAIN:=GAINS[I]+GAINS[J]-2*VALUES[I,J];
       S:='      EXGAIN ( ' +        INTTOSTR(I) +        ' , ' +        INTTOSTR(J) +
' ) : ' +        INTTOSTR(GAINS[I]) +        '+' +        INTTOSTR(GAINS[J]) +        ' - 2
* ' +        INTTOSTR(VALUES[I,J]) +        ' = ' +        INTTOSTR(EXGAIN);
       IF MAX<EXGAIN THEN
       BEGIN MAX:=EXGAIN;MAXI:=I;MAXJ:=J;S:=S+'      <------NEW MAX';END;
       MEMOS.LINES.ADD(S);
      END;
END;
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD('      MAX AT :  ' +INTTOSTR(MAXI)+' , '+INTTOSTR(MAXJ)+ '
GONNA CHANGE THEM! ');
EXCHI[ITERATION2]:=MAXI;
EXCHJ[ITERATION2]:=MAXJ;
SUMOFEXCH[ITERATION2]:=SUMOFEXCH[ITERATION2-1]+MAX;
IF ITERATION2=1 THEN MAXSUMAT:=1 ELSE BEGIN IF
SUMOFEXCH[ITERATION2]>SUMOFEXCH[MAXSUMAT] THEN
MAXSUMAT:=ITERATION2;END;
///SIL
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD('     FMXS : '+INTTOSTR(ITERATION2));
S:='      EXCH ROWS     : ';
```

```
FOR I:=1 TO ITERATION2 DO
BEGIN
S:=S+' , '+INTTOSTR(EXCHI[I]);
END;
DELETE(S,1,1);MEMOS.LINES.ADD(S);
S:='      EXCH COLUMNS  : ';
FOR I:=1 TO ITERATION2 DO
BEGIN
S:=S+' , '+INTTOSTR(EXCHJ[I]);
END;
DELETE(S,1,1);MEMOS.LINES.ADD(S);
S:='      CUMULATIVE  GAIN  : ';
FOR I:=1 TO ITERATION2 DO
BEGIN
S:=S+' , '+INTTOSTR(SUMOFEXCH[I]);
END;
DELETE(S,1,1);MEMOS.LINES.ADD(S);MEMOS.LINES.ADD('');
////////////////////////////////////////////////////////
EXCHANGE(MAXI,MAXJ);
//SIL
//FOR I:=1 TO DUGUMSAYISI DO
//BEGIN
//STRINGGRID1.CELLS[I,0]:='V '+INTTOSTR(DUGS[I]);
//STRINGGRID1.CELLS[0,I]:='V '+INTTOSTR(DUGS[I]);
//FOR J:=1 TO DUGUMSAYISI DO
//BEGIN
//IF VALUES[I,J]<>0 THEN STRINGGRID1.CELLS[I,J]:=INTTOSTR(VALUES[I,J])
ELSE STRINGGRID1.CELLS[I,J]:='';
//IF VALUES[J,I]<>0 THEN STRINGGRID1.CELLS[J,I]:=INTTOSTR(VALUES[J,I])
ELSE STRINGGRID1.CELLS[I,J]:='';
//END;
//END;
END;
PROCEDURE TMAINFORM.GETGAINS(CONST LOW,DIVIDE,HIGH:INTEGER);
VAR I,J,L,INTOP,EXTOP:INTEGER;S:STRING;EXISTS:BOOLEAN;
```

```
BEGIN
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD('      GET GAINS ...');
//MEMOS.LINES.ADD('      GET GAINS FOR
:'+inttostr(LOW)+','+inttostr(DIVIDE)+','+inttostr(HIGH));
MEMOS.LINES.ADD('');
INTOP:=0;EXTOP:=0;
FOR I:=1 TO DIVIDE DO
BEGIN
      IF CHECK(I) THEN CONTINUE;
      S:='        INT SUM  ('+INTTOSTR(I)+') : ';
      FOR J:=1 TO DIVIDE DO BEGIN S:=S+' + '+INTTOSTR(VALUES[I,J]);
      INTOP:=INTOP+VALUES[I,J];END;
      S:=S+' = '+INTTOSTR(INTOP);DELETE(S,1,1);
      MEMOS.LINES.ADD('');
      MEMOS.LINES.ADD(S);
      S:='        EXT SUM  ('+INTTOSTR(I)+') : ';
      FOR J:=DIVIDE+1 TO HIGH DO BEGIN S:=S+' +
'+INTTOSTR(VALUES[I,J]);EXTOP:=EXTOP+VALUES[I,J];END;
      S:=S+' = '+INTTOSTR(EXTOP);DELETE(S,1,1);
      MEMOS.LINES.ADD(S);
      GAINS[I]:=EXTOP-INTOP;INTOP:=0;EXTOP:=0;
      MEMOS.LINES.ADD('       GAIN    '+INTTOSTR(I)+' :  '+INTTOSTR(GAINS[I]));
END;
FOR I:=DIVIDE+1 TO HIGH DO
BEGIN
     IF CHECK(I) THEN CONTINUE;
      S:='        INT SUM  ('+INTTOSTR(I)+') : ';
      FOR J:=DIVIDE+1 TO HIGH DO BEGIN S:=S+' +
'+INTTOSTR(VALUES[I,J]);INTOP:=INTOP+VALUES[I,J];END;
      S:=S+' = '+INTTOSTR(INTOP);DELETE(S,1,1);
      MEMOS.LINES.ADD('');
      MEMOS.LINES.ADD(S);
      S:='        EXT SUM ('+INTTOSTR(I)+') : ';
```

```
      FOR J:=1 TO DIVIDE DO BEGIN
S:=S+'+'+INTTOSTR(VALUES[I,J]);EXTOP:=EXTOP+VALUES[I,J];END;
      S:=S+' = '+INTTOSTR(EXTOP);DELETE(S,1,1);
      MEMOS.LINES.ADD(S);

      GAINS[I]:=EXTOP-INTOP;INTOP:=0;EXTOP:=0;
      MEMOS.LINES.ADD('       GAIN       : '+INTTOSTR(GAINS[I]));
END;
END;
PROCEDURE TMAINFORM.SHOWLOCATER(CONST VALUE:BOOLEAN);
BEGIN
PANEL2.VISIBLE:=VALUE;
TOOLBUTTON6.DOWN:=VALUE;
EDITOR.CHECKED:=VALUE;
END;
procedure TMAINFORM.YerGsterici1Click(Sender: TObject);
begin
SHOWLOCATER(NOT EDITOR.CHECKED);
end;
procedure TMAINFORM.ToolButton6Click(Sender: TObject);
begin
SHOWLOCATER(ToolButton6.DOWN);
end;
PROCEDURE TMAINFORM.OPENFILE(Sender: TObject);
BEGIN
IF OPENDIALOG1.EXECUTE THEN
BEGIN
CLEAR;
LOADFILE(OPENDIALOG1.FILENAME);
MODIFIED:=FALSE;
NEWFILE:=FALSE;
KAYDET.ENABLED:=FALSE;
WORKINGFILE:=OPENDIALOG1.FILENAME;
PANEL5.CAPTION:='  '+WORKINGFILE;
END;
```

```
END;
PROCEDURE TMAINFORM.SAVEFILE(Sender: TObject);
BEGIN
IF NEWFILE THEN
BEGIN
SAVEDIALOG1.FILENAME:=WORKINGFILE;
 IF SAVEDIALOG1.EXECUTE THEN
 BEGIN
  RECORDFILE(SAVEDIALOG1.FILENAME);
  MODIFIED:=FALSE;
  NEWFILE:=FALSE;
  WORKINGFILE:=SAVEDIALOG1.FILENAME;
  PANEL5.CAPTION:='  '+WORKINGFILE;
 END
 ELSE
 BEGIN
  RECORDFILE(WORKINGFILE);
  MODIFIED:=FALSE;
 END;
KAYDET.ENABLED:=FALSE;
END;
END;
PROCEDURE TMAINFORM.CRNEWFILE(Sender: TObject);
VAR I:INTEGER;S:STRING;
BEGIN
CLEAR;
S:='20';
IF InputQuery('Yeni Graph Yapısı...','Düğüm Sayısı :',S)
THEN
BEGIN
 FOR I:=0 TO DUGUMSAYISI DO STRINGGRID1.Rows[I].CLEAR;
 SETDUGUMSAYISI(STRTOINT(S));
 FOR I:=1 TO DUGUMSAYISI DO DUGS[I]:=I;
 NEWFILE:=TRUE;
 MODIFIED:=FALSE;
```

```
 WORKINGFILE:='ADSIZ.KML';
 PANEL5.CAPTION:='   '+WORKINGFILE;
 OPENEDITMODE(SELF);
END;
END;
procedure TMAINFORM.FARKLIKAYDETClick(Sender: TObject);
begin
NEWFILE:=TRUE;
SAVEFILE(SELF);
end;
PROCEDURE TMAINFORM.OPENEDITMODE(Sender: TObject);
begin
STRINGGRID1.COLOR:=CLWINDOW;
DOSYA1.ENABLED:=FALSE;
EDITOR.ENABLED:=FALSE;
EDIT3.READONLY:=FALSE;
EDIT3.TABSTOP:=TRUE;
TOOLBUTTON1.ENABLED:=FALSE;
TOOLBUTTON2.ENABLED:=FALSE;
TOOLBUTTON6.ENABLED:=FALSE;
TOOLBUTTON7.ENABLED:=FALSE;
TOOLBUTTON9.ENABLED:=FALSE;
STRINGGRID1.OPTIONS:=STRINGGRID1.OPTIONS + [GOEDITING];
EDITING:=TRUE;
end;
PROCEDURE TMAINFORM.CLOSEEDITMODE(Sender: TObject);
begin
STRINGGRID1.COLOR:=CLBTNFACE;
DOSYA1.ENABLED:=TRUE;
EDITOR.ENABLED:=TRUE;
EDIT3.READONLY:=TRUE;
EDIT3.TABSTOP:=FALSE;
TOOLBUTTON1.ENABLED:=TRUE;
TOOLBUTTON6.ENABLED:=TRUE;
TOOLBUTTON7.ENABLED:=TRUE;
```

```
TOOLBUTTON9.ENABLED:=TRUE;

KAYDET.ENABLED:=MODIFIED;
TOOLBUTTON2.ENABLED:=MODIFIED;
STRINGGRID1.OPTIONS:=STRINGGRID1.OPTIONS - [GOEDITING];
EDITING:=FALSE;
end;
procedure TMAINFORM.BitBtn1Click(Sender: TObject);
begin
//GRIDTOMEM
CLOSEEDITMODE(SELF);
end;
procedure TMAINFORM.BitBtn2Click(Sender: TObject);
begin
//MEMTOGRID
CLOSEEDITMODE(SELF);
end;
PROCEDURE TMAINFORM.SETDUGUMSAYISI(CONST D:INTEGER);
VAR J:INTEGER;
BEGIN
STRINGGRID1.COLCOUNT:=D+1;
STRINGGRID1.ROWCOUNT:=D+1;
DUGUMSAYISI:=D;
FOR J:=1 TO DUGUMSAYISI DO
BEGIN
STRINGGRID1.CELLS[0,J]:=' V'+INTTOSTR(J);
STRINGGRID1.CELLS[J,0]:=' V'+INTTOSTR(J);
END;
END;
procedure TMAINFORM.StringGrid1SetEditText(Sender: TObject; ACol,
  ARow: Integer; const Value: String);
begin
IF CON2 THEN EXIT;
IF ACOL=AROW THEN
BEGIN
```

```
CON2:=TRUE;
STRINGGRID1.CELLS[ACOL,AROW]:='';
CON2:=FALSE;
EXIT;
END;
IF STRTOINTDEF(VALUE,-1)>0 THEN
BEGIN
SETMODIFIED(TRUE);
CON2:=TRUE;
STRINGGRID1.CELLS[AROW,ACOL]:=VALUE;
CON2:=FALSE;
END
ELSE
BEGIN CON2:=TRUE;
STRINGGRID1.CELLS[ACOL,AROW]:='';STRINGGRID1.CELLS[AROW,ACOL]:='';
CON2:=FALSE;
END;
end;
procedure TMAINFORM.GETTOGRID1Click(Sender: TObject);
VAR I,J:INTEGER;
begin
FOR I:=1 TO DUGUMSAYISI DO
 BEGIN
 FOR J:=1 TO DUGUMSAYISI DO
  BEGIN
  STRINGGRID1.CELLS[I,J]:=INTTOSTR(VALUES[I,J]);
  END;
 END;
end;
PROCEDURE TMAINFORM.CLEAR;
VAR I,J:INTEGER;
BEGIN
//MEMOS.LINES.ADD('');
//MEMOS.LINES.ADD('CLEARING VALUES ['+INTTOSTR(DUGUMSAYISI)+' ,
'+INTTOSTR(DUGUMSAYISI)+' ]');
```

```
//MEMOS.LINES.ADD('');
//MEMOS.LINES.ADD('CLEARING GAINS ['+INTTOSTR(DUGUMSAYISI)+' ]');
FOR I:=1 TO DUGUMSAYISI DO
BEGIN
//GAINS
GAINS[I]:=0;
FOR J:=1 TO DUGUMSAYISI DO VALUES[I,J]:=0;
END;
//BORDERS
//MEMOS.LINES.ADD('');
//MEMOS.LINES.ADD('CLEARING BORDERSH & BORDERSL
['+INTTOSTR(PARCASAYISI)+' ]');
FOR I:=1 TO PARCASAYISI DO
BEGIN
BORDERSH[I]:=0;
BORDERSL[I]:=0;
END;
END;
PROCEDURE TMAINFORM.LOADFILE(CONST FIL:STRING);
VAR S,SS:STRING;I,J,PCOL:INTEGER;PACOL:BOOLEAN;F:SYSTEM.TEXTFILE;
BEGIN
 ASSIGNFILE(F,FIL);
 RESET(F);
 READLN(F,S);
 FOR I:=0 TO DUGUMSAYISI DO STRINGGRID1.ROWS[I].CLEAR;
 SETDUGUMSAYISI(STRTOINT(S));
 FOR I:=1 TO DUGUMSAYISI DO DUGS[I]:=I;
 STRINGGRID1.CELLS[0,DUGUMSAYISI]:=' V'+INTTOSTR(DUGUMSAYISI);
 STRINGGRID1.CELLS[DUGUMSAYISI,0]:=' V'+INTTOSTR(DUGUMSAYISI);
 FOR J:=1 TO DUGUMSAYISI-1 DO
  BEGIN
   READLN(F,S);
   SS:='';
   PACOL:=TRUE;
   FOR I:=1 TO  LENGTH(S) DO
```

```
  BEGIN
   IF S[I]<>' ' THEN SS:=SS+S[I] ELSE
                        BEGIN
                         IF PACOL THEN
                          BEGIN
                           PCOL:=STRTOINT(SS);
                           SS:='';
                           PACOL:=FALSE;
                           CONTINUE;
                          END ELSE
                          BEGIN
                           PACOL:=TRUE;
                           VALUES[PCOL,J]:=STRTOINT(SS);
                           VALUES[J,PCOL]:=STRTOINT(SS);
                           STRINGGRID1.CELLS[J,PCOL]:=SS;
                           STRINGGRID1.CELLS[PCOL,J]:=SS;
                           SS:='';
                           CONTINUE;
                          END;
                        END;
    END;
    STRINGGRID1.CELLS[0,J]:=' V'+INTTOSTR(J);
    STRINGGRID1.CELLS[J,0]:=' V'+INTTOSTR(J);
  END;
  CLOSEFILE(F);
END;
      PROCEDURE TMAINFORM.RECORDFILE(CONST FIL:STRING);
      VAR I,K,J:INTEGER;F:SYSTEM.TEXTFILE;S:STRING;
      BEGIN
        ASSIGNFILE(F,FIL);
        REWRITE(F);
        S:=INTTOSTR(DUGUMSAYISI);
        WRITELN(F,S);
        FOR I:=1 TO DUGUMSAYISI-1 DO
        BEGIN
```

```
        K:=I+1;S:='';
        FOR J:=K TO DUGUMSAYISI DO
        BEGIN
         IF (STRINGGRID1.CELLS[J,I]<>'') THEN S:=S+INTTOSTR(J)+'
'+STRINGGRID1.CELLS[J,I]+' ';
        END;
        WRITELN(F,S);
        END;
       CLOSEFILE(F);
      END;
FUNCTION TMAINFORM.CHECK(CONST A:INTEGER):BOOLEAN;
VAR I:INTEGER;S:STRING;
BEGIN
RESULT:=FALSE;
FOR I:=1 TO ITERATION2-1 DO IF ((A=EXCHI[I]) OR (A=EXCHJ[I])) THEN
RESULT:=TRUE;
IF RESULT THEN
BEGIN
S:=INTTOSTR(A)+' EXCHANGED BEFORE , SKIPING IT!';
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD(S);
END;
END;
PROCEDURE TMAINFORM.EXCHANGE(VAR A,B:INTEGER);
VAR I,T:INTEGER;
BEGIN
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD('EXCHANGE '+INTTOSTR(A)+'<-->'+INTTOSTR(B));
MEMOS.LINES.ADD('');
IF A<B THEN BEGIN T:=A;A:=B; B:=T;END;
FOR I:=1 TO DUGUMSAYISI DO
BEGIN
IF ((I=A) OR (I=B)) THEN CONTINUE;
T:=VALUES[B,I];
VALUES[B,I]:=VALUES[A,I];
```

```
VALUES[I,B]:=VALUES[A,I];
VALUES[A,I]:=T;
VALUES[I,A]:=T;
END;
T:=DUGS[A];
DUGS[A]:=DUGS[B];
DUGS[B]:=T;
END;
PROCEDURE TMAINFORM.OPERATIONMAXSUM(CONST COUNT:INTEGER);
VAR I:INTEGER;
BEGIN
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD('----------');
MEMOS.LINES.ADD('MAXIMUM SUM AT :'+INTTOSTR(MAXSUMAT));
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD('WE-VE EXCHANGED:');
MEMOS.LINES.ADD('');
FOR I:=1 TO COUNT DO
BEGIN
MEMOS.LINES.ADD(INTTOSTR(EXCHI[I])+'<->'+INTTOSTR(EXCHJ[I])+'
MAKINGSUM:'+INTTOSTR(SUMOFEXCH[I]));
END;
MEMOS.LINES.ADD('');
MEMOS.LINES.ADD('WE-LL UNDO:');
MEMOS.LINES.ADD('');
FOR I:=COUNT DOWNTO MAXSUMAT+1 DO
BEGIN
MEMOS.LINES.ADD(INTTOSTR(EXCHI[I])+'<->'+INTTOSTR(EXCHJ[I]));
END;
FOR I:=COUNT DOWNTO MAXSUMAT+1 DO
 BEGIN
 EXCHANGE(EXCHI[I],EXCHJ[I]);
 END;
END;
PROCEDURE TMAINFORM.SETMODIFIED(VALUE:BOOLEAN);
```

```
BEGIN
IF MODIFIED=VALUE THEN EXIT;
MODIFIED:=VALUE;
END;
procedure TMAINFORM.ToolButton5Click(Sender: TObject);
begin
IF EDITING THEN
 BEGIN
  CLOSEEDITMODE(SELF);
  ToolButton5.DOWN:=FALSE;
  DEGISIM.CAPTION:='Değişime Aç';
 END
 ELSE
 BEGIN
  OPENEDITMODE(SELF);
  ToolButton5.DOWN:=TRUE;
  DEGISIM.CAPTION:='Değişime Kapat';
 END;
end;
procedure TMAINFORM.ToolButton9Click(Sender: TObject);
begin
if ParcaSor.showmodal=mrok then
PARCASAYISI:=strtoint(ParcaSor.RadioGroup1.items[ParcaSor.RadioGroup1.itemind
ex]) else exit;
DIVIDE;
end;
procedure TMAINFORM.EDITORClick(Sender: TObject);
begin
SHOWLOCATER(EDITOR.CHECKED);
end;
procedure TMAINFORM.DEGISIMClick(Sender: TObject);
begin
IF EDITING THEN
 BEGIN
  CLOSEEDITMODE(SELF);
```

```
  ToolButton5.DOWN:=TRUE;
  DEGISIM.CAPTION:='Değişime Aç';
 END
 ELSE
 BEGIN
  OPENEDITMODE(SELF);
  ToolButton5.DOWN:=FALSE;
  DEGISIM.CAPTION:='Değişime Kapat';
 END;
end;
procedure TMAINFORM.FormCreate(Sender: TObject);
 VAR
 I:INTEGER;
BEGIN
  DUGUMSAYISI:=20;
 FOR I:=1 TO 20 DO
          BEGIN
          STRINGGRID1.CELLS[I,0]:='V '+INTTOSTR(I);
          STRINGGRID1.CELLS[0,I]:='V '+INTTOSTR(I);
          END;
END;
procedure TMAINFORM.Edit1Change(Sender: TObject);
VAR I:INTEGER;
begin
IF CONT1 THEN EXIT;
CONT1:=TRUE;
I:=StrToIntDef(EDIT1.TEXT, 0);
IF ((I>0) AND (I<DUGUMSAYISI+1)) THEN STRINGGRID1.ROW:=I;
I:=StrToIntDef(EDIT2.TEXT, 0);
IF ((I>0) AND (I<DUGUMSAYISI+1)) THEN STRINGGRID1.COL:=I;
CONT1:=FALSE;
CONT3:=TRUE;
EDIT3.TEXT:=STRINGGRID1.CELLS[STRINGGRID1.COL,STRINGGRID1.ROW];
CONT3:=FALSE;
end;
```

```
procedure TMAINFORM.Edit2Change(Sender: TObject);
VAR I:INTEGER;
begin
IF CONT1 THEN EXIT;
CONT1:=TRUE;
I:=StrToIntDef(EDIT1.TEXT, 0);
IF ((I>0) AND (I<DUGUMSAYISI+1)) THEN STRINGGRID1.ROW:=I;
I:=StrToIntDef(EDIT2.TEXT, 0);
IF ((I>0) AND (I<DUGUMSAYISI+1)) THEN STRINGGRID1.COL:=I;
CONT1:=FALSE;
CONT3:=TRUE;
EDIT3.TEXT:=STRINGGRID1.CELLS[STRINGGRID1.COL,STRINGGRID1.ROW];
CONT3:=FALSE;
end;
procedure TMAINFORM.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
IF CONT1 THEN EXIT;
CONT1:=TRUE;
EDIT1.TEXT:=INTTOSTR(AROW);
EDIT2.TEXT:=INTTOSTR(ACOL);
CONT3:=TRUE;
EDIT3.TEXT:=STRINGGRID1.CELLS[ACOL,AROW];
CONT3:=FALSE;
CONT1:=FALSE;
end;
procedure TMAINFORM.Edit3Change(Sender: TObject);
begin
IF CONT3 THEN EXIT;
StringGrid1SetEditText(SELF,STRTOINT(EDIT1.TEXT),STRTOINT(EDIT2.TEXT),INT
TOSTR(STRTOINTDEF(EDIT3.TEXT,0)));
StringGrid1SetEditText(SELF,STRTOINT(EDIT2.TEXT),STRTOINT(EDIT1.TEXT),INT
TOSTR(STRTOINTDEF(EDIT3.TEXT,0)));
end;
procedure TMAINFORM.PARCALAClick(Sender: TObject);
```

```
begin
if ParcaSor.showmodal=mrok then
PARCASAYISI:=strtoint(ParcaSor.RadioGroup1.items[ParcaSor.RadioGroup1.itemind
ex]) else exit;
DIVIDE;
end;
End.
////////////////////////////////////////////////////////////////////////
```

EK2

ITERATION 1

SUB ITERATION COUNT: 6

ITERATION : 1.1

GET GAINS ...

INT SUM (1) : + 0 + 10 + 0 + 0 + 0 + 0 = 10

EXT SUM (1) : + 3 + 6 + 13 + 0 + 0 + 0 + 0 = 22

GAIN 1 : 12

INT SUM (2) : + 10 + 0 + 18 + 0 + 0 + 0 = 28

EXT SUM (2) : + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0

GAIN 2 : -28

INT SUM (3) : + 0 + 18 + 0 + 25 + 0 + 25 = 68

EXT SUM (3) : + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0

GAIN 3 : -68

INT SUM (4) : + 0 + 0 + 25 + 0 + 5 + 16 = 46

EXT SUM (4) : + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0

GAIN 4 : -46

INT SUM (5) : + 0 + 0 + 0 + 5 + 0 + 10 = 15

EXT SUM (5) : + 5 + 0 + 0 + 0 + 0 + 0 + 8 = 13

GAIN 5 : -2

INT SUM (6) : + 0 + 0 + 25 + 16 + 10 + 0 = 51

EXT SUM (6) : + 14 + 15 + 0 + 0 + 0 + 20 + 0 = 49

GAIN 6 : -2

INT SUM (7) : + 0 + 0 + 24 + 13 + 0 + 0 + 0 = 37

EXT SUM (7) : +3+0+0+0+5+14 = 22

GAIN : -15

INT SUM (8) : + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0

EXT SUM (8) : +6+0+0+0+0+15 = 21

GAIN : 21

INT SUM (9) : + 24 + 0 + 0 + 0 + 0 + 0 + 0 = 24

EXT SUM (9) : +13+0+0+0+0+0 = 13

GAIN : -11
INT SUM (10) : + 13 + 0 + 0 + 0 + 12 + 0 + 0 = 25
EXT SUM (10) : +0+0+0+0+0+0 = 0
GAIN : -25
INT SUM (11) : + 0 + 0 + 0 + 12 + 0 + 15 + 0 = 27
EXT SUM (11) : +0+0+0+0+0+0 = 0
GAIN : -27
INT SUM (12) : + 0 + 0 + 0 + 0 + 15 + 0 + 10 = 25
EXT SUM (12) : +0+0+0+0+0+20 = 20
GAIN : -5
INT SUM (13) : + 0 + 0 + 0 + 0 + 0 + 10 + 0 = 10
EXT SUM (13) : +0+0+0+0+8+0 = 8
GAIN : -2
GETTING EXGAINS...
EXGAIN (1 , 7) : 12+-15 - 2 * 3 = -9 <-------NEW MAX
EXGAIN (1 , 7) : 12+-15 - 2 * 3 = -9
EXGAIN (1 , 8) : 12+21 - 2 * 6 = 21 <-------NEW MAX
EXGAIN (1 , 9) : 12+-11 - 2 * 13 = -25
EXGAIN (1 , 10) : 12+-25 - 2 * 0 = -13
EXGAIN (1 , 11) : 12+-27 - 2 * 0 = -15
EXGAIN (1 , 12) : 12+-5 - 2 * 0 = 7
EXGAIN (1 , 13) : 12+-2 - 2 * 0 = 10
EXGAIN (2 , 7) : -28+-15 - 2 * 0 = -43
EXGAIN (2 , 8) : -28+21 - 2 * 0 = -7
EXGAIN (2 , 9) : -28+-11 - 2 * 0 = -39
EXGAIN (2 , 10) : -28+-25 - 2 * 0 = -53
EXGAIN (2 , 11) : -28+-27 - 2 * 0 = -55
EXGAIN (2 , 12) : -28+-5 - 2 * 0 = -33
EXGAIN (2 , 13) : -28+-2 - 2 * 0 = -30
EXGAIN (3 , 7) : -68+-15 - 2 * 0 = -83
EXGAIN (3 , 8) : -68+21 - 2 * 0 = -47
EXGAIN (3 , 9) : -68+-11 - 2 * 0 = -79
EXGAIN (3 , 10) : -68+-25 - 2 * 0 = -93
EXGAIN (3 , 11) : -68+-27 - 2 * 0 = -95
EXGAIN (3 , 12) : -68+-5 - 2 * 0 = -73

```
EXGAIN ( 3 , 13 ) : -68+-2 - 2 * 0 = -70
EXGAIN ( 4 , 7 ) : -46+-15 - 2 * 0 = -61
EXGAIN ( 4 , 8 ) : -46+21 - 2 * 0 = -25
EXGAIN ( 4 , 9 ) : -46+-11 - 2 * 0 = -57
EXGAIN ( 4 , 10 ) : -46+-25 - 2 * 0 = -71
EXGAIN ( 4 , 11 ) : -46+-27 - 2 * 0 = -73
EXGAIN ( 4 , 12 ) : -46+-5 - 2 * 0 = -51
EXGAIN ( 4 , 13 ) : -46+-2 - 2 * 0 = -48
EXGAIN ( 5 , 7 ) : -2+-15 - 2 * 5 = -27
EXGAIN ( 5 , 8 ) : -2+21 - 2 * 0 = 19
EXGAIN ( 5 , 9 ) : -2+-11 - 2 * 0 = -13
EXGAIN ( 5 , 10 ) : -2+-25 - 2 * 0 = -27
EXGAIN ( 5 , 11 ) : -2+-27 - 2 * 0 = -29
EXGAIN ( 5 , 12 ) : -2+-5 - 2 * 0 = -7
EXGAIN ( 5 , 13 ) : -2+-2 - 2 * 8 = -20
EXGAIN ( 6 , 7 ) : -2+-15 - 2 * 14 = -45
EXGAIN ( 6 , 8 ) : -2+21 - 2 * 15 = -11
EXGAIN ( 6 , 9 ) : -2+-11 - 2 * 0 = -13
EXGAIN ( 6 , 10 ) : -2+-25 - 2 * 0 = -27
EXGAIN ( 6 , 11 ) : -2+-27 - 2 * 0 = -29
EXGAIN ( 6 , 12 ) : -2+-5 - 2 * 20 = -47
EXGAIN ( 6 , 13 ) : -2+-2 - 2 * 0 = -4
MAX AT : 1 , 8 GONNA CHANGE THEM!
FMXS : 1
EXCH ROWS : , 1
EXCH COLUMNS : , 8
CUMULATIVE GAIN : , 21
EXCHANGE 1<-->8
ITERATION : 1.1 DONE!
ITERATION : 1.2
-----------------------------------------------------
GET GAINS ...
EXCHANGED BEFORE , SKIPING IT!
INT SUM (2) : + 0 + 0 + 18 + 0 + 0 + 0 = 18
EXT SUM (2) : + 0 + 10 + 0 + 0 + 0 + 0 + 0 = 10
```

```
    GAIN    2 :  -8
    INT SUM  (3) :  + 0 + 18 + 0 + 25 + 0 + 25 = 68
    EXT SUM  (3) :  + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0
    GAIN    3 :  -68
    INT SUM  (4) :  + 0 + 0 + 25 + 0 + 5 + 16 = 46
    EXT SUM  (4) :  + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0
    GAIN    4 :  -46
    INT SUM  (5) :  + 0 + 0 + 0 + 5 + 0 + 10 = 15
    EXT SUM  (5) :  + 5 + 0 + 0 + 0 + 0 + 0 + 8 = 13
    GAIN    5 :  -2
    INT SUM  (6) :  + 15 + 0 + 25 + 16 + 10 + 0 = 66
    EXT SUM  (6) :  + 14 + 0 + 0 + 0 + 0 + 20 + 0 = 34
    GAIN    6 :  -32
    INT SUM  (7) :  + 0 + 3 + 24 + 13 + 0 + 0 + 0 = 40
    EXT SUM (7) : +0+0+0+0+0+5+14 = 19
    GAIN        : -21
8 EXCHANGED BEFORE , SKIPING IT!
    INT SUM  (9) :  + 24 + 13 + 0 + 0 + 0 + 0 + 0 = 37
    EXT SUM (9) : +0+0+0+0+0+0+0 = 0
    GAIN        : -37
    INT SUM  (10) :  + 13 + 0 + 0 + 0 + 12 + 0 + 0 = 25
    EXT SUM (10) : +0+0+0+0+0+0+0 = 0
    GAIN        : -25
    INT SUM  (11) :  + 0 + 0 + 0 + 12 + 0 + 15 + 0 = 27
    EXT SUM (11) : +0+0+0+0+0+0+0 = 0
    GAIN        : -27
    INT SUM  (12) :  + 0 + 0 + 0 + 0 + 15 + 0 + 10 = 25
    EXT SUM (12) : +0+0+0+0+0+0+20 = 20
    GAIN        : -5
    INT SUM  (13) :  + 0 + 0 + 0 + 0 + 0 + 10 + 0 = 10
    EXT SUM (13) : +0+0+0+0+0+8+0 = 8
    GAIN        : -2
  GETTING EXGAINS...
1 EXCHANGED BEFORE , SKIPING IT!
  EXGAIN ( 2 , 7 ) : -8+-21 - 2 * 0 = -29      <-----NEW MAX
```

EXGAIN (2 , 7) : -8+-21 - 2 * 0 = -29
8 EXCHANGED BEFORE , SKIPING IT!
EXGAIN (2 , 9) : -8+-37 - 2 * 0 = -45
EXGAIN (2 , 10) : -8+-25 - 2 * 0 = -33
EXGAIN (2 , 11) : -8+-27 - 2 * 0 = -35
EXGAIN (2 , 12) : -8+-5 - 2 * 0 = -13 <------NEW MAX
EXGAIN (2 , 13) : -8+-2 - 2 * 0 = -10 <------NEW MAX
EXGAIN (3 , 7) : -68+-21 - 2 * 0 = -89
8 EXCHANGED BEFORE , SKIPING IT!
EXGAIN (3 , 9) : -68+-37 - 2 * 0 = -105
EXGAIN (3 , 10) : -68+-25 - 2 * 0 = -93
EXGAIN (3 , 11) : -68+-27 - 2 * 0 = -95
EXGAIN (3 , 12) : -68+-5 - 2 * 0 = -73
EXGAIN (3 , 13) : -68+-2 - 2 * 0 = -70
EXGAIN (4 , 7) : -46+-21 - 2 * 0 = -67
8 EXCHANGED BEFORE , SKIPING IT!
EXGAIN (4 , 9) : -46+-37 - 2 * 0 = -83
EXGAIN (4 , 10) : -46+-25 - 2 * 0 = -71
EXGAIN (4 , 11) : -46+-27 - 2 * 0 = -73
EXGAIN (4 , 12) : -46+-5 - 2 * 0 = -51
EXGAIN (4 , 13) : -46+-2 - 2 * 0 = -48
EXGAIN (5 , 7) : -2+-21 - 2 * 5 = -33
8 EXCHANGED BEFORE , SKIPING IT!
EXGAIN (5 , 9) : -2+-37 - 2 * 0 = -39
EXGAIN (5 , 10) : -2+-25 - 2 * 0 = -27
EXGAIN (5 , 11) : -2+-27 - 2 * 0 = -29
EXGAIN (5 , 12) : -2+-5 - 2 * 0 = -7 <------NEW MAX
EXGAIN (5 , 13) : -2+-2 - 2 * 8 = -20
EXGAIN (6 , 7) : -32+-21 - 2 * 14 = -81
8 EXCHANGED BEFORE , SKIPING IT!
EXGAIN (6 , 9) : -32+-37 - 2 * 0 = -69
EXGAIN (6 , 10) : -32+-25 - 2 * 0 = -57
EXGAIN (6 , 11) : -32+-27 - 2 * 0 = -59
EXGAIN (6 , 12) : -32+-5 - 2 * 20 = -77
EXGAIN (6 , 13) : -32+-2 - 2 * 0 = -34

```
    MAX AT :  5 , 12  GONNA CHANGE THEM!
    FMXS : 2
    EXCH ROWS    : , 1 , 5
    EXCH COLUMNS  : , 8 , 12
    CUMULATIVE  GAIN  : , 21 , 14
EXCHANGE 5<-->12
    ITERATION : 1.2 DONE!
    ITERATION : 1.3
    ----------------------------------------------------
    GET GAINS ...
1 EXCHANGED BEFORE , SKIPING IT!
      INT SUM  (2) :  + 0 + 0 + 18 + 0 + 0 + 0 = 18
      EXT SUM  (2) :  + 0 + 10 + 0 + 0 + 0 + 0 + 0 = 10
      GAIN    2 :  -8
      INT SUM  (3) :  + 0 + 18 + 0 + 25 + 0 + 25 = 68
      EXT SUM  (3) :  + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0
      GAIN    3 :  -68
      INT SUM  (4) :  + 0 + 0 + 25 + 0 + 0 + 16 = 41
      EXT SUM  (4) :  + 0 + 0 + 0 + 0 + 0 + 5 + 0 = 5
      GAIN    4 :  -36
5 EXCHANGED BEFORE , SKIPING IT!
      INT SUM  (6) :  + 15 + 0 + 25 + 16 + 20 + 0 = 76
      EXT SUM  (6) :  + 14 + 0 + 0 + 0 + 0 + 10 + 0 = 24
      GAIN    6 :  -52
      INT SUM  (7) :  + 0 + 3 + 24 + 13 + 0 + 5 + 0 = 45
      EXT SUM (7) : +0+0+0+0+0+0+14 = 14
      GAIN       : -31
8 EXCHANGED BEFORE , SKIPING IT!
      INT SUM  (9) :  + 24 + 13 + 0 + 0 + 0 + 0 + 0 = 37
      EXT SUM (9) : +0+0+0+0+0+0+0 = 0
      GAIN       : -37
      INT SUM  (10) :  + 13 + 0 + 0 + 0 + 12 + 0 + 0 = 25
      EXT SUM (10) : +0+0+0+0+0+0+0 = 0
      GAIN       : -25
      INT SUM  (11) :  + 0 + 0 + 0 + 12 + 0 + 0 + 0 = 12
```

```
        EXT SUM (11) : +0+0+0+0+15+0 = 15
        GAIN         : 3
12 EXCHANGED BEFORE , SKIPING IT!
        INT SUM  (13) :  + 0 + 0 + 0 + 0 + 0 + 8 + 0 = 8
        EXT SUM (13) : +0+0+0+0+10+0 = 10
        GAIN         : 2
      GETTING EXGAINS...
1 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 2 , 7 ) : -8+-31 - 2 * 0 = -39       <------NEW MAX
      EXGAIN ( 2 , 7 ) : -8+-31 - 2 * 0 = -39
8 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 2 , 9 ) : -8+-37 - 2 * 0 = -45
      EXGAIN ( 2 , 10 ) : -8+-25 - 2 * 0 = -33      <------NEW MAX
      EXGAIN ( 2 , 11 ) : -8+3 - 2 * 0 = -5         <------NEW MAX

12 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 2 , 13 ) : -8+2 - 2 * 0 = -6
      EXGAIN ( 3 , 7 ) : -68+-31 - 2 * 0 = -99
8 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 3 , 9 ) : -68+-37 - 2 * 0 = -105
      EXGAIN ( 3 , 10 ) : -68+-25 - 2 * 0 = -93
      EXGAIN ( 3 , 11 ) : -68+3 - 2 * 0 = -65
12 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 3 , 13 ) : -68+2 - 2 * 0 = -66
      EXGAIN ( 4 , 7 ) : -36+-31 - 2 * 0 = -67
8 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 4 , 9 ) : -36+-37 - 2 * 0 = -73
      EXGAIN ( 4 , 10 ) : -36+-25 - 2 * 0 = -61
      EXGAIN ( 4 , 11 ) : -36+3 - 2 * 0 = -33
12 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 4 , 13 ) : -36+2 - 2 * 0 = -34
5 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 6 , 7 ) : -52+-31 - 2 * 14 = -111
8 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 6 , 9 ) : -52+-37 - 2 * 0 = -89
```

```
    EXGAIN ( 6 , 10 ) : -52+-25 - 2 * 0 = -77
    EXGAIN ( 6 , 11 ) : -52+3 - 2 * 0 = -49
12 EXCHANGED BEFORE , SKIPING IT!
    EXGAIN ( 6 , 13 ) : -52+2 - 2 * 0 = -50
    MAX AT :  2 , 11  GONNA CHANGE THEM!
    FMXS : 3
    EXCH ROWS    : , 1 , 5 , 2
    EXCH COLUMNS  : , 8 , 12 , 11
    CUMULATIVE  GAIN  : , 21 , 14 , 9
EXCHANGE 2<-->11
    ITERATION : 1.3 DONE!
    ITERATION : 1.4
    ----------------------------------------------------
    GET GAINS ...
1 EXCHANGED BEFORE , SKIPING IT!
2 EXCHANGED BEFORE , SKIPING IT!

     INT SUM  (3) :  + 0 + 0 + 0 + 25 + 0 + 25 = 50
     EXT SUM  (3) :  + 0 + 0 + 0 + 0 + 18 + 0 + 0 = 18
     GAIN    3 :  -32
     INT SUM  (4) :  + 0 + 0 + 25 + 0 + 0 + 16 = 41
     EXT SUM  (4) :  + 0 + 0 + 0 + 0 + 0 + 5 + 0 = 5
     GAIN    4 :  -36
5 EXCHANGED BEFORE , SKIPING IT!
     INT SUM  (6) :  + 15 + 0 + 25 + 16 + 20 + 0 = 76
     EXT SUM  (6) :  + 14 + 0 + 0 + 0 + 0 + 10 + 0 = 24
     GAIN    6 :  -52
     INT SUM  (7) :  + 0 + 3 + 24 + 13 + 0 + 5 + 0 = 45
     EXT SUM (7) : +0+0+0+0+0+14 = 14
     GAIN       : -31
8 EXCHANGED BEFORE , SKIPING IT!
     INT SUM  (9) :  + 24 + 13 + 0 + 0 + 0 + 0 + 0 = 37
     EXT SUM (9) : +0+0+0+0+0+0 = 0
     GAIN       : -37
     INT SUM  (10) :  + 13 + 0 + 0 + 0 + 0 + 0 + 0 = 13
```

```
        EXT SUM (10) : +0+12+0+0+0+0 = 12
        GAIN         : -1
11 EXCHANGED BEFORE , SKIPING IT!
12 EXCHANGED BEFORE , SKIPING IT!
        INT SUM  (13) :  + 0 + 0 + 0 + 0 + 0 + 8 + 0 = 8
        EXT SUM (13) : +0+0+0+0+10+0 = 10
        GAIN         : 2
      GETTING EXGAINS...
1 EXCHANGED BEFORE , SKIPING IT!
2 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 3 , 7 ) : -32+-31 - 2 * 0 = -63      <------NEW MAX
      EXGAIN ( 3 , 7 ) : -32+-31 - 2 * 0 = -63
8 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 3 , 9 ) : -32+-37 - 2 * 0 = -69
      EXGAIN ( 3 , 10 ) : -32+-1 - 2 * 0 = -33      <------NEW MAX
11 EXCHANGED BEFORE , SKIPING IT!
12 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 3 , 13 ) : -32+2 - 2 * 0 = -30       <------NEW MAX
      EXGAIN ( 4 , 7 ) : -36+-31 - 2 * 0 = -67

8 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 4 , 9 ) : -36+-37 - 2 * 0 = -73
      EXGAIN ( 4 , 10 ) : -36+-1 - 2 * 0 = -37
11 EXCHANGED BEFORE , SKIPING IT!
12 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 4 , 13 ) : -36+2 - 2 * 0 = -34
5 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 6 , 7 ) : -52+-31 - 2 * 14 = -111
8 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 6 , 9 ) : -52+-37 - 2 * 0 = -89
      EXGAIN ( 6 , 10 ) : -52+-1 - 2 * 0 = -53
11 EXCHANGED BEFORE , SKIPING IT!
12 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 6 , 13 ) : -52+2 - 2 * 0 = -50
      MAX AT :  3 , 13  GONNA CHANGE THEM!
```

```
    FMXS : 4
    EXCH ROWS    : , 1 , 5 , 2 , 3
    EXCH COLUMNS : , 8 , 12 , 11 , 13
    CUMULATIVE  GAIN  : , 21 , 14 , 9 , -21
EXCHANGE 3<-->13
    ITERATION : 1.4 DONE!
    ITERATION : 1.5
    ----------------------------------------------------
    GET GAINS ...
1 EXCHANGED BEFORE , SKIPING IT!
2 EXCHANGED BEFORE , SKIPING IT!
3 EXCHANGED BEFORE , SKIPING IT!
      INT SUM  (4) :  + 0 + 0 + 0 + 0 + 0 + 16 = 16
      EXT SUM  (4) :  + 0 + 0 + 0 + 0 + 0 + 5 + 25 = 30
      GAIN    4 :  14
5 EXCHANGED BEFORE , SKIPING IT!
      INT SUM  (6) :  + 15 + 0 + 0 + 16 + 20 + 0 = 51
      EXT SUM  (6) :  + 14 + 0 + 0 + 0 + 0 + 10 + 25 = 49
      GAIN    6 :  -2
      INT SUM  (7) :  + 0 + 3 + 24 + 13 + 0 + 5 + 0 = 45
      EXT SUM (7) : +0+0+0+0+0+14 = 14
      GAIN       : -31
8 EXCHANGED BEFORE , SKIPING IT!
      INT SUM  (9) :  + 24 + 13 + 0 + 0 + 0 + 0 + 0 = 37
      EXT SUM (9) : +0+0+0+0+0+0 = 0
      GAIN       : -37
      INT SUM  (10) :  + 13 + 0 + 0 + 0 + 0 + 0 + 0 = 13
      EXT SUM (10) : +0+12+0+0+0+0 = 12
      GAIN       : -1
11 EXCHANGED BEFORE , SKIPING IT!
12 EXCHANGED BEFORE , SKIPING IT!
13 EXCHANGED BEFORE , SKIPING IT!
    GETTING EXGAINS...
1 EXCHANGED BEFORE , SKIPING IT!
2 EXCHANGED BEFORE , SKIPING IT!
```

```
3 EXCHANGED BEFORE , SKIPING IT!
     EXGAIN ( 4 , 7 ) : 14+-31 - 2 * 0 = -17      <------NEW MAX
     EXGAIN ( 4 , 7 ) : 14+-31 - 2 * 0 = -17
8 EXCHANGED BEFORE , SKIPING IT!
     EXGAIN ( 4 , 9 ) : 14+-37 - 2 * 0 = -23
     EXGAIN ( 4 , 10 ) : 14+-1 - 2 * 0 = 13      <------NEW MAX
11 EXCHANGED BEFORE , SKIPING IT!
12 EXCHANGED BEFORE , SKIPING IT!
13 EXCHANGED BEFORE , SKIPING IT!
5 EXCHANGED BEFORE , SKIPING IT!
     EXGAIN ( 6 , 7 ) : -2+-31 - 2 * 14 = -61
8 EXCHANGED BEFORE , SKIPING IT!
     EXGAIN ( 6 , 9 ) : -2+-37 - 2 * 0 = -39
     EXGAIN ( 6 , 10 ) : -2+-1 - 2 * 0 = -3
11 EXCHANGED BEFORE , SKIPING IT!
12 EXCHANGED BEFORE , SKIPING IT!
13 EXCHANGED BEFORE , SKIPING IT!
     MAX AT :  4 , 10  GONNA CHANGE THEM!
    FMXS : 5
    EXCH ROWS    : , 1 , 5 , 2 , 3 , 4
    EXCH COLUMNS  : , 8 , 12 , 11 , 13 , 10
    CUMULATIVE  GAIN  : , 21 , 14 , 9 , -21 , -8
EXCHANGE 4<-->10
     ITERATION : 1.5 DONE!
     ITERATION : 1.6
     ----------------------------------------------
     GET GAINS ...
1 EXCHANGED BEFORE , SKIPING IT!
2 EXCHANGED BEFORE , SKIPING IT!
3 EXCHANGED BEFORE , SKIPING IT!
4 EXCHANGED BEFORE , SKIPING IT!
5 EXCHANGED BEFORE , SKIPING IT!
      INT SUM  (6) :  + 15 + 0 + 0 + 0 + 20 + 0 = 35
      EXT SUM  (6) :  + 14 + 0 + 0 + 16 + 0 + 10 + 25 = 65
      GAIN    6 :  30
```

```
        INT SUM  (7) :  + 0 + 3 + 24 + 0 + 0 + 5 + 0 = 32
        EXT SUM (7) : +0+0+0+13+0+14 = 27
        GAIN        : -5
 8 EXCHANGED BEFORE , SKIPING IT!
        INT SUM  (9) :  + 24 + 13 + 0 + 0 + 0 + 0 + 0 = 37
        EXT SUM (9) : +0+0+0+0+0+0 = 0
        GAIN        : -37
 10 EXCHANGED BEFORE , SKIPING IT!
 11 EXCHANGED BEFORE , SKIPING IT!
 12 EXCHANGED BEFORE , SKIPING IT!
 13 EXCHANGED BEFORE , SKIPING IT!
      GETTING EXGAINS...
 1 EXCHANGED BEFORE , SKIPING IT!
 2 EXCHANGED BEFORE , SKIPING IT!
 3 EXCHANGED BEFORE , SKIPING IT!
 4 EXCHANGED BEFORE , SKIPING IT!
 5 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 6 , 7 ) : 30+-5 - 2 * 14 = -3      <------NEW MAX
      EXGAIN ( 6 , 7 ) : 30+-5 - 2 * 14 = -3
 8 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 6 , 9 ) : 30+-37 - 2 * 0 = -7
 10 EXCHANGED BEFORE , SKIPING IT!
 11 EXCHANGED BEFORE , SKIPING IT!
 12 EXCHANGED BEFORE , SKIPING IT!
 13 EXCHANGED BEFORE , SKIPING IT!


      MAX AT :  6 , 7  GONNA CHANGE THEM!
     FMXS : 6
     EXCH ROWS    :  , 1 , 5 , 2 , 3 , 4 , 6
     EXCH COLUMNS :  , 8 , 12 , 11 , 13 , 10 , 7
     CUMULATIVE  GAIN  :  , 21 , 14 , 9 , -21 , -8 , -11
EXCHANGE 6<-->7
      ITERATION : 1.6 DONE!
----------
MAXIMUM SUM AT :1
```

WE-VE EXCHANGED:
1<->8 MAKINGSUM:21
5<->12 MAKINGSUM:14
2<->11 MAKINGSUM:9
3<->13 MAKINGSUM:-21
4<->10 MAKINGSUM:-8
6<->7 MAKINGSUM:-11
WE-LL UNDO:
6<->7
4<->10
3<->13
2<->11
5<->12
EXCHANGE 6<-->7
EXCHANGE 4<-->10
EXCHANGE 3<-->13
EXCHANGE 2<-->11
EXCHANGE 5<-->12
ITERATION : 1 DONE
ITERATION 2
SUB ITERATION COUNT: 3

ITERATION : 2.1

GET GAINS ...

INT SUM (1) : + 0 + 0 + 0 = 0
EXT SUM (1) : + 0 + 0 + 15 = 15
GAIN 1 : 15
INT SUM (2) : + 0 + 0 + 18 = 18
EXT SUM (2) : + 0 + 0 + 0 = 0
GAIN 2 : -18
INT SUM (3) : + 0 + 18 + 0 = 18
EXT SUM (3) : + 25 + 0 + 25 = 50

```
      GAIN    3 :  32
      INT SUM  (4) :  + 0 + 5 + 16 = 21
      EXT SUM (4) : +0+0+25 = 25
      GAIN       : 4
      INT SUM  (5) :  + 5 + 0 + 10 = 15
      EXT SUM (5) : +0+0+0 = 0
      GAIN       : -15
      INT SUM  (6) :  + 16 + 10 + 0 = 26
      EXT SUM (6) : +15+0+25 = 40
      GAIN       : 14
    GETTING EXGAINS...
    EXGAIN ( 1 , 4 ) : 15+4 - 2 * 0 = 19      <------NEW MAX
    EXGAIN ( 1 , 4 ) : 15+4 - 2 * 0 = 19
    EXGAIN ( 1 , 5 ) : 15+-15 - 2 * 0 = 0
    EXGAIN ( 1 , 6 ) : 15+14 - 2 * 15 = -1
    EXGAIN ( 2 , 4 ) : -18+4 - 2 * 0 = -14
    EXGAIN ( 2 , 5 ) : -18+-15 - 2 * 0 = -33
    EXGAIN ( 2 , 6 ) : -18+14 - 2 * 0 = -4
    EXGAIN ( 3 , 4 ) : 32+4 - 2 * 25 = -14
    EXGAIN ( 3 , 5 ) : 32+-15 - 2 * 0 = 17
    EXGAIN ( 3 , 6 ) : 32+14 - 2 * 25 = -4
    MAX AT :  1 , 4  GONNA CHANGE THEM!
   FMXS : 1
   EXCH ROWS    : , 1
   EXCH COLUMNS  : , 4
   CUMULATIVE  GAIN  : , 19
EXCHANGE 1<-->4


    ITERATION : 2.1 DONE!
    ITERATION : 2.2
    ----------------------------------------------------
    GET GAINS ...
1 EXCHANGED BEFORE , SKIPING IT!
      INT SUM  (2) :  + 0 + 0 + 18 = 18
```

```
        EXT SUM  (2) :  + 0 + 0 + 0 = 0
        GAIN     2 :  -18
        INT SUM  (3) :  + 25 + 18 + 0 = 43
        EXT SUM  (3) :  + 0 + 0 + 25 = 25
        GAIN     3 :  -18
4 EXCHANGED BEFORE , SKIPING IT!
        INT SUM  (5) :  + 0 + 0 + 10 = 10
        EXT SUM (5) : +5+0+0 = 5
       GAIN        : -5
        INT SUM  (6) :  + 15 + 10 + 0 = 25
        EXT SUM (6) : +16+0+25 = 41
        GAIN        : 16
      GETTING EXGAINS...
1 EXCHANGED BEFORE , SKIPING IT!
4 EXCHANGED BEFORE , SKIPING IT!
       EXGAIN ( 2 , 5 ) : -18+-5 - 2 * 0 = -23      <-------NEW MAX
       EXGAIN ( 2 , 5 ) : -18+-5 - 2 * 0 = -23
       EXGAIN ( 2 , 6 ) : -18+16 - 2 * 0 = -2       <-------NEW MAX
       EXGAIN ( 3 , 5 ) : -18+-5 - 2 * 0 = -23
       EXGAIN ( 3 , 6 ) : -18+16 - 2 * 25 = -52
       MAX AT :  2 , 6  GONNA CHANGE THEM!
      FMXS : 2
      EXCH ROWS    : , 1 , 2
      EXCH COLUMNS  : , 4 , 6
      CUMULATIVE  GAIN  : , 19 , 17
EXCHANGE 2<-->6
       ITERATION : 2.2 DONE!
      ITERATION : 2.3
       ----------------------------------------------------
       GET GAINS ...


1 EXCHANGED BEFORE , SKIPING IT!
2 EXCHANGED BEFORE , SKIPING IT!
        INT SUM  (3) :  + 25 + 25 + 0 = 50
```

```
        EXT SUM  (3) :  + 0 + 0 + 18 = 18
        GAIN     3 :  -32
4 EXCHANGED BEFORE , SKIPING IT!
        INT SUM  (5) :  + 0 + 0 + 0 = 0
        EXT SUM (5) : +5+10+0 = 15
        GAIN         : 15
6 EXCHANGED BEFORE , SKIPING IT!
      GETTING EXGAINS...
1 EXCHANGED BEFORE , SKIPING IT!
2 EXCHANGED BEFORE , SKIPING IT!
4 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 3 , 5 ) : -32+15 - 2 * 0 = -17       <-------NEW MAX
      EXGAIN ( 3 , 5 ) : -32+15 - 2 * 0 = -17
6 EXCHANGED BEFORE , SKIPING IT!
      MAX AT :  3 , 5  GONNA CHANGE THEM!
     FMXS : 3
     EXCH ROWS     : , 1 , 2 , 3
     EXCH COLUMNS  : , 4 , 6 , 5
     CUMULATIVE  GAIN  : , 19 , 17 , 0
EXCHANGE 3<-->5
      ITERATION : 2.3 DONE!
----------
MAXIMUM SUM AT :1
WE-VE EXCHANGED:
1<->4  MAKINGSUM:19
2<->6  MAKINGSUM:17
3<->5  MAKINGSUM:0
WE-LL UNDO:
3<->5
2<->6
EXCHANGE 3<-->5
EXCHANGE 2<-->6

      ITERATION : 2 DONE
ITERATION 2
```

SUB ITERATION COUNT: 3

ITERATION : 2.1

GET GAINS ...

INT SUM (1) : + 0 + 0 + 25 + 0 + 5 + 16 + 0 + 0 + 0 = 46

EXT SUM (1) : + 0 + 0 + 0 + 0 = 0

GAIN 1 : -46

INT SUM (2) : + 0 + 0 + 18 + 0 + 0 + 0 + 0 + 10 + 0 = 28

EXT SUM (2) : + 0 + 0 + 0 + 0 = 0

GAIN 2 : -28

INT SUM (3) : + 25 + 18 + 0 + 0 + 0 + 25 + 0 + 0 + 0 = 68

EXT SUM (3) : + 0 + 0 + 0 + 0 = 0

GAIN 3 : -68

INT SUM (4) : + 0 + 0 + 0 + 0 + 0 + 15 + 0 + 6 + 0 = 21

EXT SUM (4) : + 0 + 0 + 0 + 0 = 0

GAIN 4 : -21

INT SUM (5) : + 5 + 0 + 0 + 0 + 0 + 10 + 5 + 0 + 0 = 20

EXT SUM (5) : + 0 + 0 + 0 + 8 = 8

GAIN 5 : -12

INT SUM (6) : + 16 + 0 + 25 + 15 + 10 + 0 + 14 + 0 + 0 = 80

EXT SUM (6) : + 0 + 0 + 20 + 0 = 20

GAIN 6 : -60

INT SUM (7) : + 0 + 0 + 0 + 0 + 5 + 14 + 0 + 3 + 24 = 46

EXT SUM (7) : + 13 + 0 + 0 + 0 = 13

GAIN 7 : -33

INT SUM (8) : + 0 + 10 + 0 + 6 + 0 + 0 + 3 + 0 + 13 = 32

EXT SUM (8) : + 0 + 0 + 0 + 0 = 0

GAIN 8 : -32

INT SUM (9) : + 0 + 0 + 0 + 0 + 0 + 0 + 24 + 13 + 0 = 37

EXT SUM (9) : + 0 + 0 + 0 + 0 = 0

GAIN 9 : -37

INT SUM (10) : + 0 + 12 + 0 + 0 = 12

EXT SUM (10) : +0+0+0+0+0+0+13+0+0 = 13

GAIN : 1

```
  INT SUM  (11) :  + 12 + 0 + 15 + 0 = 27
  EXT SUM (11) : +0+0+0+0+0+0+0+0+0 = 0
  GAIN        : -27
  INT SUM  (12) :  + 0 + 15 + 0 + 10 = 25
  EXT SUM (12) : +0+0+0+0+0+20+0+0+0 = 20
  GAIN        : -5
  INT SUM  (13) :  + 0 + 0 + 10 + 0 = 10
  EXT SUM (13) : +0+0+0+0+8+0+0+0+0 = 8
  GAIN        : -2
GETTING EXGAINS...
EXGAIN ( 1 , 10 ) : -46+1 - 2 * 0 = -45      <------NEW MAX
EXGAIN ( 1 , 10 ) : -46+1 - 2 * 0 = -45
EXGAIN ( 1 , 11 ) : -46+-27 - 2 * 0 = -73
EXGAIN ( 1 , 12 ) : -46+-5 - 2 * 0 = -51
EXGAIN ( 1 , 13 ) : -46+-2 - 2 * 0 = -48
EXGAIN ( 2 , 10 ) : -28+1 - 2 * 0 = -27      <------NEW MAX
EXGAIN ( 2 , 11 ) : -28+-27 - 2 * 0 = -55
EXGAIN ( 2 , 12 ) : -28+-5 - 2 * 0 = -33
EXGAIN ( 2 , 13 ) : -28+-2 - 2 * 0 = -30
EXGAIN ( 3 , 10 ) : -68+1 - 2 * 0 = -67
EXGAIN ( 3 , 11 ) : -68+-27 - 2 * 0 = -95
EXGAIN ( 3 , 12 ) : -68+-5 - 2 * 0 = -73
EXGAIN ( 3 , 13 ) : -68+-2 - 2 * 0 = -70
EXGAIN ( 4 , 10 ) : -21+1 - 2 * 0 = -20      <------NEW MAX
EXGAIN ( 4 , 11 ) : -21+-27 - 2 * 0 = -48
EXGAIN ( 4 , 12 ) : -21+-5 - 2 * 0 = -26
EXGAIN ( 4 , 13 ) : -21+-2 - 2 * 0 = -23
EXGAIN ( 5 , 10 ) : -12+1 - 2 * 0 = -11      <------NEW MAX
EXGAIN ( 5 , 11 ) : -12+-27 - 2 * 0 = -39
EXGAIN ( 5 , 12 ) : -12+-5 - 2 * 0 = -17
EXGAIN ( 5 , 13 ) : -12+-2 - 2 * 8 = -30
EXGAIN ( 6 , 10 ) : -60+1 - 2 * 0 = -59
EXGAIN ( 6 , 11 ) : -60+-27 - 2 * 0 = -87
EXGAIN ( 6 , 12 ) : -60+-5 - 2 * 20 = -105
```

```
EXGAIN ( 6 , 13 ) : -60+-2 - 2 * 0 = -62
EXGAIN ( 7 , 10 ) : -33+1 - 2 * 13 = -58
EXGAIN ( 7 , 11 ) : -33+-27 - 2 * 0 = -60
EXGAIN ( 7 , 12 ) : -33+-5 - 2 * 0 = -38
EXGAIN ( 7 , 13 ) : -33+-2 - 2 * 0 = -35
EXGAIN ( 8 , 10 ) : -32+1 - 2 * 0 = -31
EXGAIN ( 8 , 11 ) : -32+-27 - 2 * 0 = -59
EXGAIN ( 8 , 12 ) : -32+-5 - 2 * 0 = -37
EXGAIN ( 8 , 13 ) : -32+-2 - 2 * 0 = -34
EXGAIN ( 9 , 10 ) : -37+1 - 2 * 0 = -36
EXGAIN ( 9 , 11 ) : -37+-27 - 2 * 0 = -64
EXGAIN ( 9 , 12 ) : -37+-5 - 2 * 0 = -42
EXGAIN ( 9 , 13 ) : -37+-2 - 2 * 0 = -39
MAX AT : 5 , 10 GONNA CHANGE THEM!
FMXS : 1
EXCH ROWS : , 5
EXCH COLUMNS : , 10
CUMULATIVE GAIN : , -11
EXCHANGE 5<-->10
ITERATION : 2.1 DONE!
ITERATION : 2.2
----------------------------------------------------
GET GAINS ...
INT SUM (1) : + 0 + 0 + 25 + 0 + 0 + 16 + 0 + 0 + 0 = 41
EXT SUM (1) : + 5 + 0 + 0 + 0 = 5
GAIN 1 : -36
INT SUM (2) : + 0 + 0 + 18 + 0 + 0 + 0 + 0 + 10 + 0 = 28
EXT SUM (2) : + 0 + 0 + 0 + 0 = 0
GAIN 2 : -28
INT SUM (3) : + 25 + 18 + 0 + 0 + 0 + 25 + 0 + 0 + 0 = 68
EXT SUM (3) : + 0 + 0 + 0 + 0 = 0
GAIN 3 : -68
INT SUM (4) : + 0 + 0 + 0 + 0 + 0 + 15 + 0 + 6 + 0 = 21
EXT SUM (4) : + 0 + 0 + 0 + 0 = 0
GAIN 4 : -21
```

```
 5 EXCHANGED BEFORE , SKIPING IT!
        INT SUM  (6) :  + 16 + 0 + 25 + 15 + 0 + 0 + 14 + 0 + 0 = 70
        EXT SUM  (6) :  + 10 + 0 + 20 + 0 = 30
        GAIN     6 :  -40

        INT SUM  (7) :  + 0 + 0 + 0 + 0 + 13 + 14 + 0 + 3 + 24 = 54
        EXT SUM  (7) :  + 5 + 0 + 0 + 0 = 5
        GAIN     7 :  -49
        INT SUM  (8) :  + 0 + 10 + 0 + 6 + 0 + 0 + 3 + 0 + 13 = 32
        EXT SUM  (8) :  + 0 + 0 + 0 + 0 = 0
        GAIN     8 :  -32
        INT SUM  (9) :  + 0 + 0 + 0 + 0 + 0 + 0 + 24 + 13 + 0 = 37
        EXT SUM  (9) :  + 0 + 0 + 0 + 0 = 0
        GAIN     9 :  -37
10 EXCHANGED BEFORE , SKIPING IT!
        INT SUM  (11) :  + 0 + 0 + 15 + 0 = 15
        EXT SUM (11) : +0+0+0+0+12+0+0+0+0 = 12
        GAIN         : -3
        INT SUM  (12) :  + 0 + 15 + 0 + 10 = 25
        EXT SUM (12) : +0+0+0+0+0+20+0+0+0 = 20
        GAIN         : -5
        INT SUM  (13) :  + 8 + 0 + 10 + 0 = 18
        EXT SUM (13) : +0+0+0+0+0+0+0+0+0 = 0
        GAIN         : -18
      GETTING EXGAINS...
10 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 1 , 11 ) : -36+-3 - 2 * 0 = -39      <------NEW MAX
      EXGAIN ( 1 , 11 ) : -36+-3 - 2 * 0 = -39
      EXGAIN ( 1 , 12 ) : -36+-5 - 2 * 0 = -41
      EXGAIN ( 1 , 13 ) : -36+-18 - 2 * 0 = -54
      EXGAIN ( 2 , 11 ) : -28+-3 - 2 * 0 = -31      <------NEW MAX
      EXGAIN ( 2 , 12 ) : -28+-5 - 2 * 0 = -33
      EXGAIN ( 2 , 13 ) : -28+-18 - 2 * 0 = -46
      EXGAIN ( 3 , 11 ) : -68+-3 - 2 * 0 = -71
      EXGAIN ( 3 , 12 ) : -68+-5 - 2 * 0 = -73
```

```
      EXGAIN ( 3 , 13 ) : -68+-18 - 2 * 0 = -86
      EXGAIN ( 4 , 11 ) : -21+-3 - 2 * 0 = -24      <-------NEW MAX
      EXGAIN ( 4 , 12 ) : -21+-5 - 2 * 0 = -26
      EXGAIN ( 4 , 13 ) : -21+-18 - 2 * 0 = -39
 5 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 6 , 11 ) : -40+-3 - 2 * 0 = -43
      EXGAIN ( 6 , 12 ) : -40+-5 - 2 * 20 = -85
      EXGAIN ( 6 , 13 ) : -40+-18 - 2 * 0 = -58
      EXGAIN ( 7 , 11 ) : -49+-3 - 2 * 0 = -52
      EXGAIN ( 7 , 12 ) : -49+-5 - 2 * 0 = -54
      EXGAIN ( 7 , 13 ) : -49+-18 - 2 * 0 = -67
      EXGAIN ( 8 , 11 ) : -32+-3 - 2 * 0 = -35
      EXGAIN ( 8 , 12 ) : -32+-5 - 2 * 0 = -37
      EXGAIN ( 8 , 13 ) : -32+-18 - 2 * 0 = -50
      EXGAIN ( 9 , 11 ) : -37+-3 - 2 * 0 = -40
      EXGAIN ( 9 , 12 ) : -37+-5 - 2 * 0 = -42
      EXGAIN ( 9 , 13 ) : -37+-18 - 2 * 0 = -55
      MAX AT :  4 , 11  GONNA CHANGE THEM!
     FMXS : 2
     EXCH ROWS    : , 5 , 4
     EXCH COLUMNS  : , 10 , 11
     CUMULATIVE  GAIN  : , -11 , -35
EXCHANGE 4<-->11
      ITERATION : 2.2 DONE!
      ITERATION : 2.3
      ----------------------------------------------
      GET GAINS ...
       INT SUM  (1) :  + 0 + 0 + 25 + 0 + 0 + 16 + 0 + 0 + 0 = 41
       EXT SUM  (1) :  + 5 + 0 + 0 + 0 = 5
       GAIN     1 :  -36
       INT SUM  (2) :  + 0 + 0 + 18 + 0 + 0 + 0 + 0 + 10 + 0 = 28
       EXT SUM  (2) :  + 0 + 0 + 0 + 0 = 0
       GAIN     2 :  -28
       INT SUM  (3) :  + 25 + 18 + 0 + 0 + 0 + 25 + 0 + 0 + 0 = 68
       EXT SUM  (3) :  + 0 + 0 + 0 + 0 = 0
```

```
        GAIN    3 :  -68
4 EXCHANGED BEFORE , SKIPING IT!
5 EXCHANGED BEFORE , SKIPING IT!
        INT SUM  (6) :  + 16 + 0 + 25 + 0 + 0 + 0 + 14 + 0 + 0 = 55
        EXT SUM  (6) :  + 10 + 15 + 20 + 0 = 45
        GAIN    6 :  -10
        INT SUM  (7) :  + 0 + 0 + 0 + 0 + 13 + 14 + 0 + 3 + 24 = 54
        EXT SUM  (7) :  + 5 + 0 + 0 + 0 = 5
        GAIN    7 :  -49
        INT SUM  (8) :  + 0 + 10 + 0 + 0 + 0 + 0 + 3 + 0 + 13 = 26
        EXT SUM  (8) :  + 0 + 6 + 0 + 0 = 6
        GAIN    8 :  -20
        INT SUM  (9) :  + 0 + 0 + 0 + 0 + 0 + 0 + 24 + 13 + 0 = 37
        EXT SUM  (9) :  + 0 + 0 + 0 + 0 = 0
        GAIN    9 :  -37
10 EXCHANGED BEFORE , SKIPING IT!
11 EXCHANGED BEFORE , SKIPING IT!
        INT SUM  (12) :  + 0 + 0 + 0 + 10 = 10
        EXT SUM (12) : +0+0+0+15+0+20+0+0+0 = 35
        GAIN        : 25
        INT SUM  (13) :  + 8 + 0 + 10 + 0 = 18
        EXT SUM (13) : +0+0+0+0+0+0+0+0+0 = 0
        GAIN        : -18
      GETTING EXGAINS...
10 EXCHANGED BEFORE , SKIPING IT!
11 EXCHANGED BEFORE , SKIPING IT!
      EXGAIN ( 1 , 12 ) : -36+25 - 2 * 0 = -11      <------NEW MAX
      EXGAIN ( 1 , 12 ) : -36+25 - 2 * 0 = -11
      EXGAIN ( 1 , 13 ) : -36+-18 - 2 * 0 = -54
      EXGAIN ( 2 , 12 ) : -28+25 - 2 * 0 = -3       <------NEW MAX
      EXGAIN ( 2 , 13 ) : -28+-18 - 2 * 0 = -46
      EXGAIN ( 3 , 12 ) : -68+25 - 2 * 0 = -43
      EXGAIN ( 3 , 13 ) : -68+-18 - 2 * 0 = -86
4 EXCHANGED BEFORE , SKIPING IT!
5 EXCHANGED BEFORE , SKIPING IT!
```

```
      EXGAIN ( 6 , 12 ) : -10+25 - 2 * 20 = -25
      EXGAIN ( 6 , 13 ) : -10+-18 - 2 * 0 = -28
      EXGAIN ( 7 , 12 ) : -49+25 - 2 * 0 = -24
      EXGAIN ( 7 , 13 ) : -49+-18 - 2 * 0 = -67
      EXGAIN ( 8 , 12 ) : -20+25 - 2 * 0 = 5      <------NEW MAX
      EXGAIN ( 8 , 13 ) : -20+-18 - 2 * 0 = -38
      EXGAIN ( 9 , 12 ) : -37+25 - 2 * 0 = -12
      EXGAIN ( 9 , 13 ) : -37+-18 - 2 * 0 = -55
      MAX AT :  8 , 12  GONNA CHANGE THEM!
     FMXS : 3
     EXCH ROWS    : , 5 , 4 , 8
     EXCH COLUMNS  : , 10 , 11 , 12
     CUMULATIVE  GAIN  : , -11 , -35 , -30
EXCHANGE 8<-->12
      ITERATION : 2.3 DONE!
---------
MAXIMUM SUM AT :1
WE-VE EXCHANGED:
5<->10  MAKINGSUM:-11
4<->11  MAKINGSUM:-35
8<->12  MAKINGSUM:-30
WE-LL UNDO:
8<->12
4<->11
EXCHANGE 8<-->12
EXCHANGE 4<-->11
      ITERATION : 2 DONE
_____ FOUND GROUPS :
  N1 = {  V4 , V2 , V3  }
  N2 = {  V8 , V10 , V6  }
  N3 = {  V7 , V1 , V9  }
  N4 = {  V5 , V11 , V12 , V13  }
CALCULATION COMPLETED.....
```