EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(DOKTORA TEZİ)

ÇİZGELERDE TEPE BİRLEŞTİRİLMİŞLİK SAYISI ÜZERİNE

Tina BEŞERİ SEVİM

Tez Danışmanı: Prof. Dr. Urfat NURİYEV

Matematik Anabilim Dalı

Bilim Dalı Kodu: 619.03.03

Sunuş Tarihi: 15.09.2010

Bornova-İZMİR

2010

Tina BEŞERİ SEVİM tarafından DOKTORA TEZİ olarak sunulan **"ÇİZGELERDE TEPE BİRLEŞTİRİLMİŞLİK SAYISI ÜZERİNE"** başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesinin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 15.09.2010 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	: Prof. Dr. Urfat NURİYEV	.....
Raportör Üye	: Yrd. Doç. Dr. Burak ORDİN	.....
Üye	: Prof. Dr. Alpay KIRLANGIÇ	.....
Üye	: Prof. Dr. Tatyana YAKHNO	.....
Üye	: Doç. Dr. Ünal UFUKTEPE	.....

ÖZET

ÇİZGELERDE TEPE BİRLEŞTİRİLMİŞLİK SAYISI ÜZERİNE

BEŞERİ SEVİM, Tina

Doktora Tezi, Matematik Bölümü
Tez Danışmanı: Prof. Dr. Urfat NURİYEV
Eylül 2010, 60 sayfa

Çizge Kuramı, temel matematikteki matematiksel ilişkiler konusu incelenirken kullanılabildiği gibi, modern hayatta karşılaşılan karmaşık ve geniş kapsamlı birçok probleme de çözüm getirebilmektedir. Herhangi bir ağ yapısı çizgelerle ifade edildiğinde bu çizgenin bazı tepelerinin bozulması durumunda dahi bağlantı noktaları arasındaki iletişimin sürdürülmesi istenir. Ağlarda iletişimi sağlayan bağlantı noktaları, çizgelerdeki tepelere karşılık gelir.

Bu tezde öncelikle tepe birleştirilmişlik kavramı ve bu kavramla ile ilgili tanım ve teoremler verilmiş, daha sonra literatürde yer alan tepe birleştirilmişlik algoritmaları incelenmiştir. Birleştirilmiş bir çizgenin tepe birleştirilmişlik sayısını veren yeni bir algoritma önerilmiştir. Bu algoritmanın literatürdeki diğer algoritmalardan farkı, silinen tepeler kümesini de verebilmesidir. Bu algoritmaya ek olarak, tepe birleştirilmişlik problemi ile ilgili matematiksel modeller önerilmiştir.

Bunun yanı sıra, bir çizgedeki yoğun tepe ve tepe yoğunluk sayısı kavramları tanımlanarak temel çizge sınıfları için tepe yoğunluk sayıları hesaplanmış ve sonuçları verilmiştir. Son olarak, genetik algoritma tekniği ile tepe birleştirilmişlik sayısını elde etmek için bir bilgisayar programı yazılmış ve farklı çizgeler için hesaplama denemeleri yapılmıştır.

Anahtar Sözcükler: Çizge Kuramı, tepe birleştirilmişlik, tepe birleştirilmişlik algoritmaları, matematiksel model, tepe yoğunluk sayısı.

ABSTRACT

VERTEX-CONNECTIVITY NUMBER ON GRAPHS

BEŞERİ SEVİM, Tina

PhD in Mathematics

Supervisor: Prof. Dr. Urfat NURİYEV

September 2010, 60 pages

Graph theory can be used when examining the mathematical relations in basic mathematics, can give a solution to many complex and comprehensive problems encountered in modern life. Any network structure, when it is expressed by graphs, even if corruption of some vertices, the communication between the ports is to be maintained. The ports in the network communications corresponds vertices in graph theory.

In this thesis, firstly the concept of vertex connectivity definitions and theorems are given, then vertex connectivity algorithms are investigated in the literature. A new algorithm is proposed for calculating the vertex connectivity number of a connected graph. The difference of this algorithm from similar algorithms in literature is to be given the set of deleted vertices. In addition to this algorithm, new mathematical models have been proposed related with the vertex connectivity problem.

Besides of this, an intense vertex and vertex intensity number concepts are defined. The intensity number is calculated for some graph families and the results were given.

Finally, by using genetic algorithms technique, we have obtained vertex connectivity number. A computer program is written and the computational experiments are carried out for different graphs.

Key Words: Graph Theory,vertex connectivity, vertex connectivity algorithms, mathematical models, vertex intensity number.

TEŞEKKÜR

Tez çalışmalarım süresince önerilerini ve bilimsel bilgilerini her zaman benimle paylaşan değerli danışmanım Prof. Dr. Urfat NURİYEV'e, programlama konusunda desteklerini her zaman hissettiğim Matematik Bölümü Araştırma Görevlisi Dr. Murat Erşen Berberler'e, yoğun tez çalışmalarımda bana her konuda yardım ederek desteğini benden hiçbir zaman esirgemeyen sevgili eşim Koray Sevim'e ve beni her zaman teşvik eden aileme sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

ÖZET . . . . . **v**

ABSTRACT . . . . . **vii**

TEŞEKKÜR . . . . . **ix**

ŞEKİLLER DİZİNİ . . . . . **xv**

ÇİZELGELER DİZİNİ . . . . . **xvii**

SİMGELER DİZİNİ . . . . . **xix**

1 GİRİŞ **1**

2 GRAFLARDA BİRLEŞTİRİLMİŞLİK KAVRAMI İLE İLGİLİ TEMEL TANIM VE TEOREMLER **5**

2.1 Temel Graf Bilgileri 5

2.2 Graflarda Tepe Birleştirilmişlik Kavramı ile İlgili Tanımlar 9

2.3 Graflarda Tepe Birleştirilmişlik Kavramı ile İlgili Teoremler 11

2.4 Temel Graflarda Tepe Birleştirilmişlik Ölçümleri 15

3 TEPE BİRLEŞTİRİLMİŞLİK ALGORİTMALARI **16**

3.1 Algoritmik Karmaşıklık 16

3.2 Graf Üzerinde Dolaşma Algoritmaları 17

3.2.1 Derinlik Öncelikli Arama (DFS-Depth First Search) 17

3.2.2 Genişlik Öncelikli Arama (BFS-Breadth First Search) . . . 18

3.3 En Kısa Yol Algoritmaları 18

3.3.1 Dijkstra Algoritması 19

3.3.2 Floyd Algoritması 19

3.4 Tepe Birleştirilmişlik Algoritmalarının Gelişimi 20

3.4.1 Tepe Birleştirilmişlik için Yeni Bir Algoritma (TBA) 25

4 GRAFLARDA TEPE BİRLEŞTİRİLMİŞLİK PROBLEMİNİN MATEMATİKSEL MODELLERİ 30
4.1 Problemin Tanımı 30
4.2 Problemin En Kısa Yol Problemine Dayanan Matematiksel Modeli . 30
4.2.1 Modelin Örneklenmesi 32
4.3 Problemin Genel Matematiksel Modeli 35
4.3.1 Modelin Örneklenmesi 37
4.3.2 Matematiksel Modelin Bir Graf için GAMS Programı Yardımıyla Çözümü 39

5 YOĞUNLUK KAVRAMI 40
5.1 Tepe Yoğunluk Kavramının Tanımı 40
5.2 Temel Grafların Tepe Yoğunluk Sayısı 42
5.2.1 Sonuçlar 43

6 GENETİK ALGORİTMA ve TEPE BİRLEŞTİRİLMİŞLİK 44
6.1 Genetik Algoritmaların Tanımı 46
6.2 Basit Bir Genetik Algoritma Yapısı 47
6.2.1 Kodlama 48
6.2.2 Başlangıç Popülasyonunun Oluşturulması 48
6.2.3 Uygunluk Fonksiyonu ve Seçme 49
6.2.4 Genetik Operatörler 49
6.3 Genetik Algoritma ile Tepe Birleştirilmişlik Hesabı 50

7 SONUÇ 55

KAYNAKLAR DİZİNİ 56

ÖZGEÇMİŞ 60

EKLER . . . . .

Ek 1 Genetik Algoritma ile Tepe Birleştirilmişliği Delphi Diliyle Hesaplayan Programın Kaynak Kodu (gatbs) . . . . .

İÇİNDEKİLER (devam)

Ek 2 Tepe Birleştirilmişlik Sayısını C Dili ile Hesaplayan Program Kaynak Kodu . . . . . .

Ek 3 Tepe Birleştirilmişlik Probleminin GAMS ile Çözümü . . . . . .

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

2.1 Temel Graflar 8
2.2 G grafında iki tepe arasındaki tepe birleştirilmişlik sayısının hesabı. 14

3.1 G=(V,E) grafının minimum tepe kesim kümesi S. 21
3.2 G=(V,E) yönlü grafının minimum tepe kesim kümesi S. 22
3.3 6 tepeli G grafı. 26
3.4 1 tepesi silinmiş G grafı. 27
3.5 1 ve 5 tepeleri silinmiş G grafı. 28
3.6 2 tepesi silinmiş G grafı. 28
3.7 1 ve 2 tepeleri silinmiş G grafı. 29

4.1 4 tepeli G=(V,E) grafı 39

5.1 5 tepeli G_1 grafı 41
5.2 5 tepeli G_2 grafı 41

6.1 Genetik algoritmaların yapısı ve işleyişi ile ilgili akış diyagramı . . 50
6.2 **gatbs**'nın Programına Giriş Ekranı 51
6.3 Graf Boyutunun Ayarlandığı Ekran 52
6.4 Grafın Bitişiklik Matrisinin Girildiği Ekran 52
6.5 Grafın Ekran Görüntüsü, Birey ve Jenerasyon Sayısının Seçimi . . . 53
6.6 Grafın Silenecek Tepe ve Ayrıtlarının Gösterimi 53
6.7 Sonuçların Alındığı Ekran 54

ÇİZELGELER DİZİNİ

Çizelge **Sayfa**

3.1 Tepe Birleştirilmişlik Algoritmalarının Tarihsel Gelişimi 24
3.2 G grafına ait Bitişiklik Matrisi 27

5.1 5 tepeli G_1 grafina ait Tepe Yoğunluk Tablosu 41
5.2 5 tepeli G_2 grafina ait Tepe Yoğunluk Tablosu 42
5.3 P_n yol grafının tepe yoğunluk sayısı 42
5.4 $K_{1,n-1}$ yıldız grafının tepe yoğunluk sayısı 42
5.5 C_n çevre grafının tepe yoğunluk sayısı 43
5.6 K_n tam grafının tepe yoğunluk sayısı 43

6.1 İkili düzende kodlama örneği 48
6.2 Permütasyon kodlama örneği 48
6.3 Değer kodlaması örneği 48

SİMGELER DİZİNİ

Simgeler	Açıklama
$deg(v)$	v **tepesinin derecesi**
$\delta(G)$	**en küçük tepe derecesi**
$\Delta(G)$	**en büyük tepe derecesi**
$d(u,v)$	u **ve v tepeleri arasındaki uzaklık**
K_n	n **tepeli tam graf**
P_n	n **tepeli yol graf**
C_n	n **tepeli çevre graf**
$K_{1,n-1}$	n **tepeli yıldız graf**
$W_{1,n}$	$(n+1)$ **tepeli tekerlek graf**
$K_{m,n}$	**iki parçalı tam graf**
$\overline{G}$	**tümleyen graf**
a_{ij}	**bitişiklik matrisi**
b_{ij}	**komşuluk matrisi**
$\kappa_G(s,t)$	**lokal tepe bağlantılılık sayısı**
$\kappa(G)$	**tepe bağlantılılık sayısı**
$\lambda(G)$	**ayrıt bağlantılılık sayısı**
$Y_t(G)$	**tepe yoğunluk sayısı**

1. GİRİŞ

18. yüzyılda ünlü matematikçi Euler'in Königsberg köprüleri problemine çözüm ararken ortaya çıkan çizge (graf) teorisi, 20. yüzyılın başında König ve Kuratowski'nin, Cayley'in ve daha yakınlarda Berge, Erdös ve Harary'nin çalışmalarıyla bir matematik dalı haline geldi. Bilgisayar alanında ve özellikle algoritmalar üzerinde yapılan araştırmalar, graf teoriye yeni bir soluk getirdi. Graf teori, çeşitli uygulamalar için oluşturulan problemleri, noktalar ve noktalar arası bağlantılar yardımıyla çizilen konfigürasyonlara indirgeyerek çözme olanağı verir. Graf Teori, genel olarak ayrık matematiğin bir alt dalı olarak sınıflandırılmıştır, fakat bunun yanında cebir ve matris teorisi ile de yakından ilgilidir. Ayrıca, graflar üzerine matematiksel anlamda geliştirilen birçok teorem, aksiyom ve algoritma yazılım dünyasında da yoğun kullanım alanı bulmaktadır. Bağlantı noktaları ve aralarında yol anlamında bağlantı olan her tür uygulama graf veri modeliyle ifade edilebilir. Graf teori, her şeyden önce çözümü aranan bir problemi ya da işi en etkin şekilde temsil edebilmeye ve düzenlemeye yarar. Bir problem graf biçimine çevrildikten sonra, tüm amaçları yerine getirecek en hızlı veya en az masraflı yolu bulmak için sistematik yöntemler aranır.

19. yüzyıldan itibaren elektrik devreleri ve molekül diyagramları başta olmak üzere grafların kullanım alanları gün geçtikçe artmıştır. Fizik, kimya gibi temel bilimlerde, mühendislik uygulamalarında ve tıp alanında birçok problemin çözümü ve modellenmesi graflara dayandırılarak yapılmaktadır. Graf Teori yardımıyla modellenebilen eşleme, ulaşım, bilgisayar ağları ve akış problemleri genel olarak programlama problemleri olarak isimlendirilir. Graf Teori ayrıca birbirinden son derece bağımsız olan ekonomi, yönetim bilimi, satış ve pazarlama, bilgi iletimi, psikoloji, kimya ve biyoloji alanlarında da kullanılabilmektedir. Özellikle büyük şehirlerde her türlü acil durum planlaması yapılırken, veriler birbirleri ile ilişkilendirilebilmeli ve tüm bu verilerin birlikte analizi yapılabilmelidir. Kent bilgi sistemi uygulamalarında, acil durumlarda ambulans, itfaiye ve polis araçlarının istenen noktaya en kısa sürede ulaşması, zamana bağlı çalışan otobüs, okul taşıtları, metro, dağıtım ve benzeri hizmetleri sorgulama ve izleme ihtiyacı vardır. Bütün bu analiz işlemleri ağ analizi ile mümkündür. Ağ analizinin gerçekleştirilebilmesi için tepe-ayrıt topolojisinin oluşturulması gerekir.

Bilimsel ve teknik modellemelerde ve onların bilgisayar ortamında benzetimlerinin gerçekleştirilmesinde, çoğu zaman graflar ve onlara ait teoremler, aksiyomlar çözüm olmaktadır. Eğer bir problemin çözümü graf veri modeli şekline benzetilebiliyorsa o problem için algoritmik bir durum elde edilmiş olunur ve problemin çözümünde tanımlanmış tüm graf teoremleri ve aksiyomları kullanılabilir ve graflar üzerine geliştirilmiş olan veri yapılarıyla onlara ait olan algoritmalar, fonksiyonlar program içerisinde doğrudan kullanılabilir.

Günümüzde Graf Teori, temel matematikteki matematiksel ilişkiler konusu incelenirken kullanılabildiği gibi, modern hayatta karşılaşılan karmaşık ve geniş kapsamlı çok sayıdaki probleme de çözüm getirebilmektedir. Graf Teori problemleri tanımlama ve yapısal olarak ilişkileri belirlemekte oldukça kolaylık sağlar. O halde graf nedir? Basit olarak bir graf, tepeler olarak adlandırılan noktalar ve herbiri bu tepeleri veya sadece tepenin kendisini birleştiren ve ayrıt olarak adlandırılan bir fonksiyondur. Bu tanımdan hareketle kimya alanında bir molekülün yapısını oluşturan atomları tepelerle, atomları bir arada tutan kimyasal bağları ayrıtlarla veya bilgisayar bilimlerinde bir bilgisayar ağındaki bilgisayarları tepelerle, bilgisayar ağındaki iletişim kablolarını ayrıtlarla modelleyebiliriz. Bir bilgisayar ağında en az kaç tane bilgisayar bozulursa ağ işlevini yitirir sorusunun cevabı graflardaki modelleme yardımıyla bulunabilir. Literatürde bu sorunun cevabı tepe bağlantılılık kavramı ile bulunmaktadır. Tepe bağlantılılık kavramı Graf Teorideki zedelenebilirlik ölçümlerinden bir tanesidir. Zedelenebilirlik ölçümlerini hesaplamak için öncelikle yapısı bilinen özel graflar üzerinde çalışılır, daha sonra tüm graflar için genelleme yapılır. Özel grafların belirli özelliklerinden yararlanılarak herhangi bir graf ile bu graflar arasındaki işlemsel ilişki incelenmektedir.

Ağ tasarlama, dayanıklılık ve planlama problemlerine algoritmik açıdan bakıldığında bağlantılılık kavramı önemli bir rol oynar. Bir ağın bağlantılılığının kalitesi tepelerinin bağlantılılığı ile ilgilidir. Yönsüz bir grafta, graftan silindiğinde grafı bağlantısız yapan veya graftaki bileşen sayısını arttıran tepeye kesim tepe, grafı parçalamak için graftan silinmesi gereken tepe sayısına ise kesim değeri denir. Kesim küme, silindiklerinde grafı bağlantısız yapan tepelerin bir kümesidir. En küçük kesim küme, kesim kümeler içinde eleman sayısı en az olan kümedir. Bu durumda tepe birleştirilmişlik sayısı ise bir graftaki minimum kesim değeridir. Bağlantılılıkla ilgili bir problemde genellikle bağlantılılık değerine ulaşılabilen ya da bu değerin korun-

abildiği bir graf dizayn edilir ya da var olan graf üzerinde değişiklik yapılır. Bağlantılılık değerini içeren pekçok problemin polinom zamanda çözülebildiği bilinmektedir. Bağlantılılıkla ilgili problemler için geliştirilen polinom zamanlı algoritmalarda genellikle Ford ve Fulkerson'un maksimum-akış minimum-kesim teoremi kullanılır.

Menger 1927 yılında, grafların birleştirilmişlik sayısı ile graftaki herhangi iki tepe arasındaki kesişmeyen yolların sayısı arasında bir ilişki olduğunu ispatlamıştır. Bu teorem ve farklı yorumlarından çıkan sonuçlar kullanılarak birçok algoritma geliştirilmiştir. Bu algoritmaların ortak özelliği grafın birleştirilmişlik sayısını bulmasıdır. Ancak bu algoritmalar kullanılarak bu sayının hangi tepe veya tepelerin silinmesi ile bulunabileceği bilinmemektedir. Dolayısıyla bu algoritmalar karmaşıklık bakımından iyi ancak ayrıntılar açısından yetersizdir. Menger'den günümüze kadar birçok bilim adamı ve araştırmacı literatüre önemli katkılar yapmıştır. Bu tezde de, graflarda tepe birleştirilmişlik kavramı üzerine inceleme yapılarak yeni algoritmalar ve matematiksel modeller önerilmiştir. Tezin izleyen bölümleri şu şekilde özetlenebilir:

İkinci bölümde temel graf bilgileri verilerek birleştirilmişlik ve tepe birleştirilmişlik kavramları tanıtılmıştır. Bu kavramlarla ilgili tanım ve teoremler verilerek temel graf sınıfları için tepe birleştirilmişlik ölçümleri sunulmuştur.

Üçüncü bölümde, algoritmik karmaşıklık kavramı, en kısa yol algoritmaları, arama algoritmaları ve akış algoritmalarından kısaca bahsedilerek tepe birleştirilmişlik sayısını veren algoritmalar incelenmiştir. Tepe birleştirilmişlik sayısını ve grafı bağlantısız yapmak için graftan silinmesi gereken tepeleri sunan yeni bir algoritma önerilmiş ve bu algoritmayı kullanarak yazılan bilgisayar programıyla örnek bir problem çözülmüştür.

Dördüncü bölümde, tepe birleştirilmişlik problemi, problemin önerilen matematiksel modelleri ve bu modellerin örnekleri verilmiştir.

Beşinci bölümde tepe birleştirilmişlik kavramına bağlı olarak graftaki en yoğun tepenin bulunması hedeflenmiştir. Bir grafta hangi tepe ya da tepeler atılırsa graftaki iletişim daha erken kesilir sorusuna yanıt aramak için grafın tepe yoğunluk sayısı tanımlanmıştır ve bu sayı Tepe Yoğunluk Sayısı - $Y_t(G)$ olarak gösterilmiştir. Ayrıca temel graf sınıfları $(P_n, C_n, K_{1,n-1}, K_n)$ için tepe yoğunluk sayılarını elde etmeye ve yukarıdaki soruya yanıt vermeye çalışılmıştır.

Altıncı bölümde genetik algoritmalar detaylı bir şekilde incelenerek tepe birleşti-

rilmişlik sayısının bulunması problemine genetik algoritma ile çözüm önerisi getirilmiş ve tasarlanmış programlar yoluyla hesaplama denemeleri yapılmıştır.

Yedinci bölümde tezde elde edilen sonuçlardan bahsedilmiştir.

Tezin son iki bölümünde ise kaynaklar dizini ve ve önerilen algoritmaların bilgisayar programlarının kaynak kodlarının verildiği ekler bulunmaktadır.

2. GRAFLARDA BİRLEŞTİRİLMİŞLİK KAVRAMI İLE İLGİLİ TEMEL TANIM VE TEOREMLER

Bu bölümde, graf teorideki temel kavramların tanımları ve açıklamaları verilmiştir. Aşağıda yer alan tanımlar Graph Theory and Its Applications (Gross and Yellen, 2004), Graph Theory: Modeling, Applications, and Algorithms (Agnarsson and Greenlaw, 2007) ve Graph Theory (Bondy and Murty, 2008) adlı kitaplardan alınmıştır.

2.1 Temel Graf Bilgileri

Tanım 2.1.1. Matematiksel anlamda **graf**, tepeler ve bu tepeler arasındaki ilişkiyi gösteren ayrıtlardan oluşan bir kümedir. Yönsüz bir $G = (V, E)$ grafı şunlardan oluşur:

- grafın tepe sayısı $|V| = n$ olmak üzere boş olmayan sonlu bir V tepeler kümesi
- grafın ayrıt sayısı $|E| = m$ olmak üzere sonlu bir E ayrıtlar kümesi
- $f : E \to V$ fonksiyonunda her bir e ayrıtı için $f(e)$ fonksiyonu, V'nin bir veya iki elemanlı bir alt kümesidir.

Tanım 2.1.2. Bir G grafı üzerindeki u ve v tepeleri, ayrıtlar kümesinde bulunan bir ayrıtla ilişkilendiriliyorsa bu tepeler **komşu tepe** adını alır ve $e = uv$ ya da $e = (u, v)$ şeklinde gösterilir. Diğer bir anlatımla e ayrıtı u ve v tepelerini birbirine bağlar veya u ve v tepeleri e kenarının uç tepeleridir denir.

Tanım 2.1.3. Bir G grafında herhangi bir $v \in V$ tepesine bitişik ayrıtların sayısına **v tepesinin derecesi** denir ve $deg(v)$ ile gösterilir. G grafının en küçük tepe derecesi $\delta(G)$, en büyük tepe derecesi ise $\triangle(G)$ ile gösterilir.

Tanım 2.1.4. Bir G grafındaki başlangıç ve bitiş tepeleri aynı olan ayrıta **bukle** denir.

Tanım 2.1.5. Bir G grafında, herhangi bir v tepesi herhangi bir ayrıt ile bitişik değilse, bu tepeye **izole tepe** denir.

Tanım 2.1.6. Tek bir tepe içeren, ayrıt içermeyen bir grafa **aşikar graf** denir.

Tanım 2.1.7. Bir G grafının herhangi iki tepesi arasında birden fazla ayrıt varsa bu ayrıtlara **çok katlı ayrıt**, bu tür graflara ise **çok katlı graf** denir.

Tanım 2.1.8. Çok katlı ayrıt ve bukle içermeyen graflara **basit graf** denir.

Tanım 2.1.9. Bir G grafının tepelerini birleştiren bütün e_i ayrıtlarının bir w_i ağırlığıyla birebir ilişkilendirilmesiyle oluşan grafa **ağırlıklandırılmış graf** denir. Ağırlıklandırılmamış bir grafın ayrıtlarının ağırlığı 1 olarak alındığında graf, ağırlıklandırılmış bir graf olarak düşünülür.

Tanım 2.1.10. Bir $G = (V, E)$ grafında $V' \subset V$ ve $E' \subset E$ olacak biçimde tanımlanan $H = (V', E')$ grafına G grafının bir **alt grafı** denir ve $H \subset G$ biçiminde gösterilir.

Tanım 2.1.11. Bir $G = (V, E)$ grafından $V' = V$ ve $E' \subset E$ olmak üzere oluşturulan $H = (V', E)$ alt grafı G grafının bir **örten alt grafı** olarak adlandırılır.

Tanım 2.1.12. $S \subset V$ boş olmayan bir küme olmak üzere, tepelerinin kümesi S, ayrıtlarının kümesi ise her iki tepesi S kümesinde olan ayrıtların oluşturduğu $G_1 \subset G$ alt grafına S kümesinin etkilediği **etkilenmiş alt graf** denir.

Tanım 2.1.13. Bir G grafının bazı tepelerinin veya ayrıtlarının çıkarılmasıyla elde edilen graf en az iki parçadan oluşuyorsa bu parçaların her birine grafın bir **bileşeni** adı verilir. Başka bir tanımla, bir G grafının bağlantılı olma özelliğine göre maksimal olan alt graflarına G'nin bileşenleri denir.

Tanım 2.1.14. Birleştirilmiş bir G grafındaki herhangi iki tepe u ve v olsun. u ve v tepeleri arasındaki **uzaklık**, bu tepeleri birbirine bağlayan en kısa yolun uzunluğudur ve $d(u, v)$ ile gösterilir.

Tanım 2.1.15. Farklı tepe çiftlerinin tümü bir ayrıt ile bağlı olan n tepeli basit grafa **tam graf** adı verilir ve K_n ile gösterilir. Bir K_n grafında tüm tepelerin derecesi $(n-1)$, ayrıtların sayısı ise $\dfrac{n(n-1)}{2}$'dir.

Tanım 2.1.16. Tüm tepeleri aynı r derecesine sahip grafa **r-düzenli graf** denir. Tam graflar aynı zamanda düzenli bir graf olup bunun tersi her zaman geçerli değildir. Yani, düzenli bir graf her zaman tam graf değildir.

Tanım 2.1.17. Bir G grafının alt graflarının içinden tam graf olanların, en büyüğüne G grafındaki bir **klik** denir.

Tanım 2.1.18. Bir G grafında $(1 \leq i \leq n)$ olmak üzere, her $e_i = v_{i-1}v_i$ ayrıtı için, tepelerin ve ayrıtların bir sıralı dizisi $(v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n)$ bir **yürüyüş**tür. Bir yürüyüşteki ayrıtların sayısı o yürüyüşün uzunluğunu verir.

Tanım 2.1.19. Bütün ayrıtları birbirinden farklı olan yürüyüşe **zincir** denir. Bir zincirde bir tepeden birden fazla kez geçilebilir.

Tanım 2.1.20. Bir graftaki v_0 tepesinden v_n tepesine uzunluğu n olan bir **yol**, $(n+1)$ tepeden ve n ayrıttan oluşan, $(v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n)$ şeklinde ifade edilen bir dizidir. Başka bir anlatımla, uç tepeleri 1, iç tepeleri 2 dereceli olan grafa **yol graf** denir. n tepeli yol graf P_n ile gösterilir.

Tanım 2.1.21. Tüm tepelerinin derecesi 2 ve tepe sayısı $n \geq 3$ olan kapalı bir yürüyüş **çevre graf** olarak adlandırılır. n tepeli bir çevre grafın ayrıt sayısı n tane olup, bu graf C_n ile gösterilir.

Tanım 2.1.22. n tepeli bir grafın bir tepesi $n-1$, diğer tüm tepeleri 1 dereceli ise bu grafa **yıldız graf** denir ve $K_{1,n-1}$ ile gösterilir.

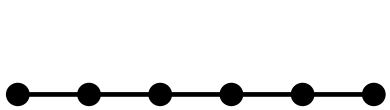
Tanım 2.1.23. n tepeli bir çevre grafın her bir tepesine, bu n tepeden farklı bir tek tepeden birer ayrıt eklenmesiyle elde edilen grafa **tekerlek graf** denir ve bu graf $W_{1,n}$ ile gösterilir. Tekerlek grafı $K_1 + C_n$ şeklinde de ifade etmek mümkündür.

Tanım 2.1.24. Bir $G = (V, E)$ grafı verildiğinde, $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$ olacak şekilde V kümesinin iki alt kümesi var ve E kümesindeki herbir ayrıt, V_1 kümesindeki bir tepe ile V_2 kümesindeki bir tepeyi birleştiriyorsa bu tür graflara **iki parçalı graf** denir.

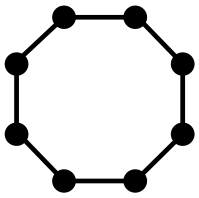
Tanım 2.1.25. Basit bir G grafının tepeler kümesi biri m adet tepe içeren V_1, diğeri n adet tepe içeren V_2 şeklinde iki ayrık kümeye parçalanıyor ve $s \in V_1,\ t \in V_2$ olmak üzere her bir (s, t) çifti arasında bir ayrıt bulunuyorsa G grafına m ve n ayrıtlı **iki parçalı tam graf** denir ve $K_{m,n}$ ile gösterilir.

Tanım 2.1.26. G grafı, n tepesi olan basit bir graf olsun. G grafının **tümleyen**i olan $\overline{G}$ grafı, K_n tam grafından G grafının ayrıtlarının silinmesiyle elde edilir.

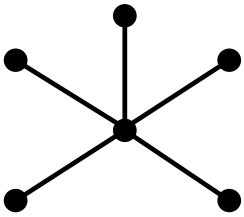
Tanım 2.1.27. G_1 ve G_2 graflarının tepeler ve ayrıtlar kümesi sırasıyla $V(G_1)$, $V(G_2)$ ve $E(G_1)$, $E(G_2)$ olsun. $E(G_1)$ ve $E(G_2)$ kümelerindeki tüm ayrıtlarla $V(G_1)$ ve $V(G_2)$ kümelerindeki tüm tepelerin oluşturduğu G grafına G_1 ve G_2 **graflarının birleşimi** denir ve $G = G_1 \cup G_2$ şeklinde gösterilir.



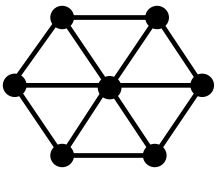
P_6 Yol Grafı



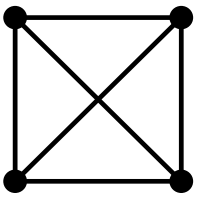
C_8 Çevre Grafı



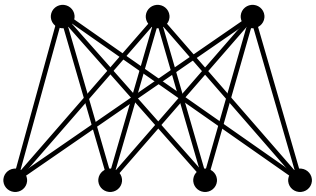
$K_{1,5}$ Yıldız Grafı



$W_{1,6}$ Tekerlek Grafı



K_4 Tam Grafı



$K_{3,4}$ İki Parçalı Tam Grafı

Şekil 2.1: Temel Graflar

Tanım 2.1.28. $G_1 \cap G_2 = \emptyset$ olmak üzere G_1 ve G_2 graflarının oluşturduğu $G_1 \cup G_2$ grafında, G_1 grafının her bir tepesini G_2 grafının tüm tepelerine birleştirerek elde edilen G grafına G_1 ve G_2 **graflarının toplamı** denir ve $G = G_1 + G_2$ ile gösterilir.

Tanım 2.1.29. Tepelerden tepelere olan bağlantıyı gösteren $(n \times n)$ boyutlu matris, **bitişiklik matrisi** adını alır. Bitişiklik matrisi,

$$a_{ij} = \begin{cases} 1, & \text{eğer } (v_i, v_j) \in E \text{ veya } (v_j, v_i) \in E \text{ ise;} \\ 0, & \text{diğer durumlarda.} \end{cases}$$

şeklinde gösterilir. Matrisin simetrikliği sol-üst köşe ile sağ-alt köşe arasından çizilecek bir köşegen çizgisine göredir. Bitişiklik matrisinde, matrisin satır veya sütun

toplamları grafın tepe derecelerini verir. Bitişiklik matrisinin tercih edilme sebeplerinden biri, matristen tek bir sorgulama ile tepeleri birleştiren ayrıtlar hakkında bir bilgi edinilebilmesidir. Dezevantajı ise bellekte n^2'lik yer kaplamasıdır.

Tanım 2.1.30. Tepelerle ayrıtlar arasındaki bağlantı ilişkisini gösteren $(n \times m)$ boyutlu matrise **komşuluk matrisi** adı verilir. Komşuluk matrisi,

$$b_{ij} = \begin{cases} 1, & \text{eğer } v_i \text{ tepesi } e_{ij} \text{ ayrıtına bitişik ise;} \\ 0, & \text{diğer durumlarda.} \end{cases}$$

şeklinde gösterilir. Komşuluk matrisinde satırlar tepe, sütunlar da ayrıtlara aittir. Bu matris, ayrıtlar arası bilgileri içermediğinden bilgisayarlı modellemede çok da tercih edilmemektedir ve bellekte $(n \times m)$'lik yer kaplar. Bu nedenle algoritmik açıdan en kötü graf ifadelendirilmesi olarak değerlendirilmektedir.

2.2 Graflarda Tepe Birleştirilmişlik Kavramı ile İlgili Tanımlar

Bu çalışmada, n-tepeli, buklesiz, katlı ayrıtsız, yönsüz, ağırlıklandırılmamış grafların tepe birleştirilmişlik ölçümü üzerinde durulmuştur (Esfahanian and Hakimi, 1984), (Galil, 1980).

Tanım 2.2.1. Eğer bir graftaki her $\{u, v\} \in V$ tepe çifti için, u'dan v'ye bir yol varsa, $G(V, E)$ grafına **birleştirilmiş** (**bağlantılı**) **graf** denir. Birleştirilmiş (bağlantılı) olmayan graflar, **birleştirilmemiş** (**bağlantılı olmayan**) **graf** adını alır. Verilen bir $G = (V, E)$ grafında, G'nin bağlantılı bir bileşeni, o grafın bağlantılı ve maksimal olan $G' = (V', E')$ etkilenmiş alt grafıdır. (Yani $V'' \supset V'$ olacak şekilde bağlantılı başka bir $G'' = (V'', E'')$ alt grafı yoktur.)

Tanım 2.2.2. Birleştirilmiş bir G grafını bağlantısız bir graf ya da sadece izole tepelerden oluşan bir graf haline getirmek için graftan atılması gerekli en az ayrıt sayısına **G grafının ayrıt birleştirilmişlik sayısı** denir ve $\lambda(G)$ ile gösterilir.

Tanım 2.2.3. Birleştirilmiş bir G grafını bağlantısız bir graf ya da sadece izole tepelerden oluşan bir graf haline getirmek için graftan atılması gerekli en az tepe sayısına **G grafının tepe birleştirilmişlik sayısı** denir ve $\kappa(G)$ ile gösterilir.

Tanım 2.2.4. n tepeli bir grafta iki ayrık s ve t tepesi için s'den t'ye olan tüm yolları yok etmek için atılması gereken en az tepe sayısı $\kappa_G(s, t)$ ile gösterilen **lokal tepe**

birleştirilmişlik olarak adlandırılır. Eğer s'den t'ye bir ayrıt varsa, $\kappa_G(s,t) = n-1$ olarak seçilir; çünkü diğer durumlarda κ_G, $(n-2)$'yi geçemez. Yönsüz graflar için $\kappa_G(s,t) = \kappa_G(t,s)$'dir.

Tanım 2.2.5. Bir $G = (V,E)$ grafındaki herhangi iki tepe s ve t ele alındığında, $st \notin E$ ise, $V - \{s,t\}$'den seçilen ve silindiklerinde G'deki her s ve t çifti arasındaki tüm yolları ortadan kaldıran tepelerin minimum sayısı $\kappa_G(s,t)$'ye karşılık gelmektedir. Bir ağın zedelenebilirliğini ölçmek için kullanılan ölçümlerden biri olan birleştirilmişlik, yani $\kappa(G)$ ise bu tanımdan hareketle şu şekilde ifade edilir:

$$\kappa(G) = min\{\kappa_G(s,t); s,t \in V\}.$$

Tanım 2.2.6. Bir $G = (V,E)$ grafında $X \subseteq V$ bir tepeler kümesi olmak üzere, $G - X$, X kümesindeki tüm tepelerin ve bu tepelere bitişik olan tüm ayrıtların silinmesiyle oluşan bir graftır. Eğer $|V| > k$ ve $|X| < k$ olacak şekilde her $X \subset V$ için $G - X$ bağlantılı oluyorsa $G = (V,E)$ yönsüz grafına **k-tepe bağlantılı** denir. G'nin tepe birleştirilmişliği, grafı k-tepe bağlantılı yapan en büyük k tamsayısıdır.

Boş olmayan her graf 0-tepe bağlantılıdır ve 1-tepe bağlantılı graflar en azından 2 tepesi olan bağlantılı graflardır. Bir tek tepeye sahip graf 0-tepe bağlantılıdır, fakat 1-tepe bağlantılı değildir. $k \geq 1$ olan herhangi bir G grafı bağlantılıdır. G'nin bu durumda 1-bağlantılı olduğu kabul edilir. Benzer şekilde eğer $k \geq 2$ ise G grafını bağlantısız hale getirebilmek için en az 2 tepe silinmesi gerekir, bu durumda da G'nin 2-bağlantılı olduğu kabul edilir. $k \geq 2$ veya $k \geq 3$ şeklinde küçük bir sınır belirleyerek, daha sonra k'nın gerçek değerini hesaplamak daha kolaydır. Eğer G bağlantısız bir graf ise, bu grafın bileşenleri olan maksimal bağlantılı alt grafları incelenir. İzole tepeden oluşan bir bileşen için $\kappa = 0$ olup, diğer tüm bileşenler Δ-bağlantılıdır. Eğer bağlantılı bir G grafı v kesim tepesine sahipse bu grafa ayrılabilir graf denir, çünkü v tepesi silindiğinde G grafı 2 veya daha fazla bileşene ayrılır. Ayrılabilir bir grafta $k = 1$'dir, fakat bu graf 2-bağlantılı olan alt graflar içerebilir. Benzer biçimde bağlantısız graflar da bağlantılı alt graflar içerebilirler. Bağlantısız bir grafın bileşenlerini bularak, G'nin maksimal ayrılamayan alt grafları bulunmuş olur. G grafının maksimal ayrılamayan alt graflarına, G'nin blokları denir. Ayrılabilir her graf 2 veya daha fazla blok içerir. 2-bağlantılı herhangi bir graf ayrılamayan bir graftır. K_2 grafı tek bir ayrıta sahip, kesim tepesi olmayan dolayısıyla ayrılamayan bir graftır.

Tanım 2.2.7. $\kappa(G) = \lambda(G) = \delta(G)$ ise G grafına **maksimal birleştirilmiş graf** adı verilir.

Tanım 2.2.8. $\kappa(G) = \delta(G)$ ise G grafına **maksimal tepe birleştirilmiş graf** adı verilir.

Tanım 2.2.9. $G = (V, E)$ yönsüz bir graf olsun. $s \in V$'den $t \in V$'ye uzanan iki yol P_1 ve P_2 olmak üzere, bu yollar s ve t dışında hiçbir tepeyi paylaşmıyorsa bu yollara **tepe ayrık yollar** adı verilir.

Tanım 2.2.10. Bağlantılı bir G grafında sadece tepelerden oluşan U alt grafı G'den silindiğinde elde edilen $G - U$ grafı en az 2 bileşene sahip oluyorsa U alt grafı **kesim küme** ya da **tepe kesim kümesi** adını alır. Başka bir ifadeyle, $G = (V, E)$ yönsüz grafında $C \subset V$ alt kümesi için eğer $G - C$'nin bağlantılı bileşenlerinin sayısı, G'dekilerden daha fazla ise C'ye tepe kesim kümesi denir. Eğer s ve t tepeleri G'nin aynı bağlantılı bileşeninde bulunup $G - C$'nin farklı bağlantılı bileşenlerinde yer alıyorlarsa, C'ye **s-t tepe kesimi** denir.

Tanım 2.2.11. Kesim kümeler içinde eleman sayısı en az olan kümeye **en küçük kesim küme** denir.

Tanım 2.2.12. Bağlantılı bir G grafında eğer V kesim küme ise $v \in V$ tepesine **kesim-tepe** denir. Bir diğer anlatımla, kesim-tepe graftan çıkarıldığında o grafın bağlantılı bileşenlerinin sayısını arttıran bir tepedir.

Tanım 2.2.13. Yönlü bir G grafında, her tepeden, diğer tüm tepelere yönlü bir yol varsa o graf **güçlü birleştirilmiş**tir. Yönlü bir G grafının etkilenmiş bir alt grafı, güçlü birleştirilmiş ve maksimal ise o alt graf **güçlü birleştirilmiş bileşen** adını alır. Yönlü bir grafın güçlü birleştirilmiş bileşenleri geliştirilmiş bir DFS kullanılarak $O(n + m)$ zamanda hesaplanabilir. Eğer yönlü bir grafın yönsüz hali de bağlantılı ise bu grafa **zayıf birleştirilmiş** denir.

2.3 Graflarda Tepe Birleştirilmişlik Kavramı ile İlgili Teoremler

Teorem 2.3.1. 3 *veya daha fazla tepeye sahip herhangi bir graf, tüm tepe çiftleri en az* 2 *içten ayrık yolla bağlantılı ise* 2*-bağlantılıdır.*

Öyleyse 2-bağlantılı bir grafta, tüm u, v tepe çiftleri P ve Q gibi en az 2 tane içten ayrık yolla bağlıdır. P ve Q birlikte bir çevre oluşturduğundan, her tepe çifti bir çevre üzerinde yer alır. Bu teoremin başka bir sonucu da tüm ayrıt çiftlerinin yine bir çevre üzerinde yer almasıdır.

Sonuç 2.3.2. 3 *veya daha fazla tepeye sahip herhangi bir graf, tüm ayrıt çiftleri bir çevre üzerinde yer alıyorsa* 2*-bağlantılıdır.*

Teorem 2.3.3. *Aşikar olmayan tüm* G *grafları için,*

$$\kappa(G) \leq \lambda(G) \leq \delta(G)$$

elde edilir.

Kanıt. $\delta(G)$ minimum dereceli tepeye bitişik olan ayrıtlar, bir ayrıt kesim küme oluşturur. Buradan $\lambda(G) \leq \delta(G)$ sonucuna ulaşılır. n tepeli herhangi bir grafın tepe bağlantılılığı, tam grafın bağlantılılığı gözönüne alınarak $\kappa(K_n) = n - 1$ ile sınırlansın. $G = (V, E)$ en az iki tepeli bir graf ve minimal ayrıt kesim kümenin, S tepe kümesini, diğer tüm tepelerin oluşturduğu $\overline{S} = V \setminus S$ kümesinden ayırır. S ve $\overline{S}$ arasındaki tüm ayrıtlar G'de ise,

$$\lambda(G) = \mid S \mid \cdot \mid \overline{S} \mid \geq \mid V \mid -1$$

elde edilir. Aksi halde $\{x, y\} \notin E$ olan $x \in S$ ve $y \in \overline{S}$ tepeleri mevcuttur. x'in, $\overline{S}$'deki tüm komşularının kümesi ve buna ek olarak $S \setminus \{x\}$'deki tepelerden $\overline{S}$'da komşusu olanlar bir tepe kesim oluştururlar. Kesim kümenin eleman sayısı S'den $\overline{S}$'ye doğru olan ayrıtların maksimum sayısı kadardır ve bu kesim (en azından) x ve y'yi birbirinden ayırır. □

Graflardaki bağlantılılık çalışmalarında iki yorum ortaya konulmuştur. Bunlardan ilki, grafı bağlantısız yapmak için atılan tepe sayısı, diğeri ise grafın verilen herhangi iki tepesini birleştiren alternatif yolların sayısıdır. Bu iki yorumdan yola çıkarak, G bağlantılı grafındaki komşu olmayan u ve v tepeleri ile ilgili şu iki optimizasyon problemi verilir:

Max. Problemi: G grafındaki içten ayrık $u - v$ yollarının max. sayısını bulma.

Min. Problemi: u ve v tepelerini ayırabilmek için G grafından atılması gerekli en az tepe sayısı.

Önerme 2.3.4. ***(Zayıf İkililik)*** *Bağlantılı bir G grafının komşu olmayan herhangi iki tepesi u ve v olsun. G grafındaki içten ayrık $u-v$ yollarının bir kümesi P_{uv} ve $u-v$ ayıran kümesinin tepelerinden oluşan bir küme S_{uv} olsun. Bu durumda $\mid P_{uv} \mid \leq \mid S_{uv} \mid$'dir.*

Kanıt. S_{uv} ayıran küme olduğundan, P_{uv}'deki her $u-v$ yolu, S_{uv}'nin en az bir tepesini içermelidir. P_{uv}'deki yollar da içten ayrık olduğundan, herhangi iki yol aynı tepeyi içermez. Öyleyse, G'deki içten ayrık $u-v$ yollarının sayısı en fazla $\mid S_{uv} \mid$ kadardır. □

Sonuç 2.3.5. *u ve v bağlantılı bir G grafının komşu olmayan herhangi iki tepesi olsun. G'deki içten ayrık $u-v$ yollarının maksimum sayısı, $u-v$ ayıran kümesinin minimum tepe sayısına eşit ya da daha küçüktür.*

Sonuç 2.3.6. ***(Optimalliğin garantisi)*** *u ve v bağlantılı bir G grafının komşu olmayan iki tepesi olsun. G'deki içten ayrık $u-v$ yollarının bir kümesi P_{uv}, $u-v$ ayıran kümesi S_{uv} ve $\mid P_{uv} \mid = \mid S_{uv} \mid$ olsun. Bu durumda P_{uv}, içten ayrık $u-v$ yollarının maksimum ölçüdeki bir kümesidir ve S_{uv} minimum ölçülü bir $u-v$ ayıran kümesidir.*

1927 yılında Menger tarafından ispatlanan bir teoremle, önceden sözü edilen iki optimizasyon problemi arasındaki güçlü ikililik kavramı kurulmuştur.

Teorem 2.3.7. *(Menger, 1927) Bağlantılı bir G grafında ayrık ve komşu olmayan iki tepe u ve v olsun. Buna göre, G'deki içten ayrık $u-v$ yollarının maksimum sayısı, u'yu v'den ayırmak için atılması gerekli tepelerin minimum sayısına eşittir.*

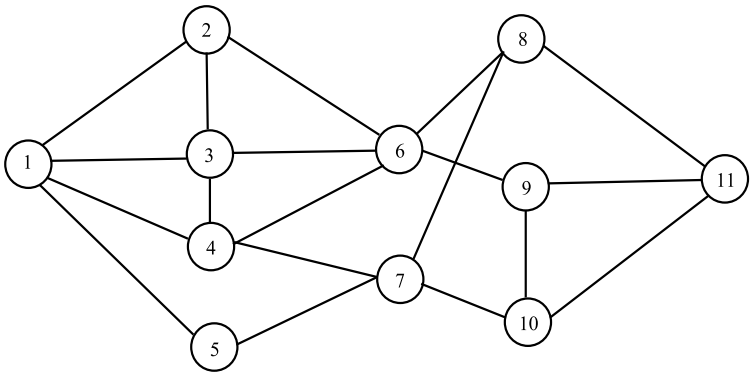
Menger Teoreminin başka bir ifadesi şu şekildedir:

Teorem 2.3.8. *(Menger, 1927) P ve Q yönsüz bir grafın tepelerinden oluşan alt kümeler ise, P ve Q'daki tepeleri birbirine bağlayan tepe-ayrık yolların maksimum sayısı, P'deki bir tepeden Q'daki bir tepeye doğru olan bütün yollar üzerindeki tepelerin minimum sayısına eşittir.*

Örnek 2.3.9. Aşağıda verilen bir $G=(V,E)$ grafındaki 1 ile 11 tepeleri arasındaki tepe birleştirilmişliğin araştırmasında tepe ayrık yolların sayısından yararlanılır.

(1,2,6,8,11) ve (1,4,7,10,11) yolları 6 ve 7 tepelerini kullanan iki farklı yoldur. Buna benzer şekilde, birinde 6 tepesinin, diğerinde ise 7 tepesinin kullanıldığı başka içten ayrık yollarda bulunabilir, fakat bu graf için bulunabilecek tepe ayrık yolların

maksimum sayısı 2'dir. Dolayısıyla Menger Teoremi (Menger, 1927) gereğince içten ayrık yolların maksimum sayısı, graftan atılması gereken minimum tepe sayısına eşittir, yani G grafı için $\kappa(G) = 2$'dir.



Şekil 2.2: G grafında iki tepe arasındaki tepe birleştirilmişlik sayısının hesabı.

Teorem 2.3.10. *(Ford and Fulkerson, 1956) Maksimum $s-t$ akışının değeri minimum $s-t$ kesiminin kapasitesine eşittir.*

Sonuç 2.3.11. *Yönsüz bir $G = (V, E)$ grafına ait iki tepe s ve t olsun. Eğer bu iki tepe komşu tepe değilse, tepe ayrık s-t yolarının maksimum sayısı, s-t tepe kesim kümenin minimum büyüklüğüne eşittir. (s ve t'yi bağlantısız yapmak için graftan çıkarılması gereken minimum tepe sayısına eşittir.)*

Teorem 2.3.12. *(Whitney, 1932) $G = (V, E)$ aşikar olmayan bir graf ve k pozitif bir tamsayı olsun. G grafi k-tepe bağlantılıdır ancak ve ancak her u, v tepe çifti için G'de en az k tane tepe ayrık $u - v$ yolu vardır.*

Sonuç 2.3.13. *G grafi k bağlantılı bir graf ve $(u, v_1, v_2, \ldots, v_k)$ G'nin herhangi $k+1$ ayrık tepesi olsun. $i = (1, \ldots, k)$ için u'dan v_i'ye öyle P_i yolları bulunabilir ki P_i kümesi içten ayrık olur.*

Teorem 2.3.14. *G grafi p tepeli ve q ayrıtlı birleştirilmiş bir graf ise $p \leq q + 1$ elde edilir.*

Teorem 2.3.15. *(Dirac, 1960) $k \geq 3$ için en az $(k + 1)$ tepeli k-bağlantılı bir graf G olsun ve bu graftaki k tepenin oluşturduğu herhangi bir küme U olsun. Buna göre, U'daki bütün tepeleri içeren bir çevre graf G'de bulunmaktadır.*

Bir sonraki teorem tepe bağlantılığın sınırlarını verir.

Teorem 2.3.16. *n tepeli ve m ayrıtlı bir graftaki maksimum tepe birleştirilmişlik sayısı,*

$$\begin{cases} \lfloor \frac{2m}{n} \rfloor, & \text{eğer } m \geq n-1 \text{ ise,} \\ 0, & \text{diğer durumlarda.} \end{cases}$$

n tepeli ve m ayrıtlı bir graftaki minimum tepe birleştirilmişlik sayısı,

$$\begin{cases} m - \binom{n-1}{2}, & \binom{n-1}{2} < m < \binom{n}{2} \text{ ise,} \\ 0, & \text{diğer durumlarda.} \end{cases}$$

Teorem 2.3.17. *İki ayrık k-tepe bileşenin en çok $(k-1)$ ortak tepesi vardır.*

Teorem 2.3.18. *(Mader, 1972) Ortalama derecesi en az $4k$ olan her graf, k-bağlantılı bir alt grafa sahiptir.*

2.4 Temel Graflarda Tepe Birleştirilmişlik Ölçümleri

- K_n tam grafında $(n-1)$ tepe silindikten sonra geriye izole tepe yani K_1 grafı kalır. Bu nedenle $\kappa(K_n) = n-1$'dir.

- P_n yol grafı için $n = 2$ durumunda herhangi iki tepesinden birinin silinmesi ile geriye izole tepe kalır. $n > 2$ durumunda ise grafın uç tepeleri hariç, graftan herhangi bir tepenin silinmesi ile graf bağlantısız olmaktadır. Bu nedenle $\kappa(P_n) = 1$'dir.

- C_n çevre grafında herhangi bir tepeden diğer tepeye farklı iki yoldan gidilebileceği için tek bir tepenin silinmesi grafın bağlantısız olması için yeterli değildir. Silinen tepeye komşu olmayan başka bir tepenin de silinmesi gerekmektedir. Bu nedenle $\kappa(C_n) = 2$'dir.

- $K_{1,n-1}$ yıldız grafında merkezdeki tepenin diğer tüm tepelere komşu olması ve merkez tepe dışındaki tepe çiftleri birbirine bitişik olmadığından merkez tepenin silinmesi ile graf $n-1$ tane izole tepe haline dönüşür. Bu nedenle $\kappa(K_{1,n-1}) = 1$'dir.

3. TEPE BİRLEŞTİRİLMİŞLİK ALGORİTMALARI

3.1 Algoritmik Karmaşıklık

Bir algoritmanın çalışma hızı karmaşıklık kavramına karşılık gelir ve bu kavram zaman birimiyle ifade edilmeyip doğrudan işlem adedi veya döngü sayısı ile gösterilir. Algoritma karmaşıklığı iki açıdan ele alınır. Biri alan karmaşıklığı, diğeri zaman karmaşıklığıdır. Alan karmaşıklığı probleminin çözümü, algoritmayı gerçeklerken kullanılan veri yapıları ile bağlantılıdır. Algoritmanın zaman karmaşıklığı ise, belirli miktardaki giriş verisine karşılık, yapılan karşılaştırma, tamsayı toplama, tamsayı çıkartma, tamsayı çarpma ve bölme işlemleri ile diğer basit işlemlerin sayısı olarak hesaplanır. Alan karmaşıklığı algoritmanın ihtiyaç duyacağı bellek miktarı hakkında bilgi verir. Zaman karmaşıklığı ise algoritmanın sonuca ulaşması için gerekli zaman hakkında bilgi verir. Bir algoritmanın hesaplama karmaşıklığının değerlendirilmesi onun ne kadar hızlı çalışacağı ve bilgisayarın hafızasında ne kadar yer kullanacağına ilişkin bilgiler vermektedir. Çalışma zamanın üst sınırı olan, en kötü durumdaki zamanın belirlenmesinde O notasyonu kullanılır.

Bir hesaplamanın zaman karmaşıklığı, belirli boyuttaki giriş değerleri için harcanan basit hesapsal adımların sayısını veren bir fonksiyondur. Bir algoritmanın n uzunluklu giriş verisi üzerinde, basit ikili işlemlerle ifade edilen çalışma zamanı (problemin çözümleme zamanı) üstten herhangi bir $P(n)$ polinomu ile sınırlı ise bunlara **polinom zamanlı algoritmalar** denir. Bu algoritmalarla çözülebilen bütün problemler **P sınıfı** olarak ifadelendirilir. P sınıfına dahil olan problemler iyi biçimlendirilmiş problemler cinsindendirler. Polinom zaman içerisinde deterministik olmayan makinelerde çözülebilen her bir algoritma deterministik olmayan polinom zamanlı algoritma olarak ele alınacaktır ve bu problemler NP (polinom olmayan) sınıfı oluşturmaktadır. P problemlere NP problemlerin alt kümesi biçiminde bakılabilir. P problemi NP sınıfına dahil herhangi bir problem kadar zorsa, bu P problemi NP-zor'dur. NP sınıfına dahil her problemin dönüşebildiği problem NP-zor'dur. Eğer problem NP'ye dahil ve aynı zamanda NP-zor ise, bu problem NP-tam'dır.

3.2 Graf Üzerinde Dolaşma Algoritmaları

Graflarla temsil edilen pek çok problemde çözümün bulunması için graflar üzerinde çeşitli şekillerde dolaşılması istenmektedir. Graf üzerinde dolaşma, grafın tepeleri ve ayrıtları üzerinde istenen bir işi yapacak veya bir problemi çözecek biçimde hareket etmektir. Mühendislikte, temel bilimlerde ve sosyal bilimlerde birçok problemin çözümü graf veri modeli ile ifade edilir ve çözümleri, çoğunlukla, graf üzerinde dolaşma algoritmalarıyla sağlanır. Dolayısıyla graf üzerinde dolaşma algoritmaları bilgisayar biliminde önemli bir yer tutar.

Greedy yaklaşımı, graf üzerinde belirli bir konuda optimum sonucu veya en iyi sonucu bulabilmek amacıyla dolaşma yapılırken bir sonraki tepeyi belirlemek için kullanılan bir seçme yöntemidir. Greedy yaklaşımında o andaki seçenekler içerisinden en iyi olarak görünen seçilir; kriter yerel değerlendirmelere göre yapılır ve seçilenin global olarak tüm sistem için en iyi seçim olacağı öngörülür. Greedy yaklaşımı graf üzerinde geliştirilmiş olan birçok problemin çözümünde, bir sonraki tepenin belirlenmesinde seçme unsuru olarak kullanılmaktadır, bilgisayar biliminde graf veri modeline yaklaştırılan problemlerde geniş bir kullanım alanı vardır.

Graf üzerinde dolaşma yapan yaklaşım yöntemlerinden diğer iki tanesi, kısaca, DFS (Derinlik Öncelikli Arama) ve BFS (Genişlik Öncelikli Arama) olarak adlandırılır ve graf üzerine geliştirilen algoritmaların birçoğu bu yaklaşım yöntemlerine dayanmaktadır.

3.2.1 Derinlik Öncelikli Arama (DFS-Depth First Search)

Derinlik öncelikli arama olarak adlandırılan DFS yöntemi, graf üzerinde dolaşma yöntemlerinden birisidir. Bu algoritmada, graf üzerinde dolaşmaya başlangıç tepesinin bir ayrıtından başlanıp o ayrıt üzerinden gidilebilecek en uzak (derin) tepeye kadar arama sürdürülür. Ara tepeler için de benzer davranışta bulunur; o tepenin bir ayrıtından gidilebilecek en uzak tepeye kadar gidilir; gidilemeyen tepe varsa o tepenin bir başka ayrıtından denenir. Bu algoritma, adı üzerinde derinlik öncelikli dolaşmadır; bir tepeden kendisine bağlı birçok tepe arasından, önce birine gidilir; oradan da bağlı olan tepelerden birine gidilir. Ancak bu şekilde ilerlerken gidilemeyen tepeler varsa rekürsif program davranışında olduğu gibi bir geriye dönülür ve o tepeden, bir sonraki tepeye gidilir. Buradaki sıralama, tepelerin herhangi bir

düzende sıralanmasına dayanır.

3.2.2 Genişlik Öncelikli Arama (BFS-Breadth First Search)

Genişlik-öncelikli arama olarak adlandırılan BFS yöntemi, graf üzerinde dolaşma yöntemlerinden birisidir. Bu yöntemin DFS'den en önemli farkı, dolaşmaya, başlangıç tepesinin bir ayrıtı üzerinden en uzağa gidilmesiyle değil, başlangıç tepesinden gidilebilecek tüm komşu tepelere gidilmesiyle başlanır. Ara tepelerde de başlangıç tepesi gibi davranılır; herhangi bir ara tepeye gelindiğinde, o tepeye komşu ve daha önce ziyaret edilmemiş olan tüm tepelere gidilir. BFS yöntemiyle graf üzerinde dolaşma, graf üzerinde dolaşarak işlem yapan diğer birçok algoritmaya esin kaynağı olmuştur. Örneğin, ayrıt maliyetleri yoksa veya eşitse, BFS yöntemi en kısa yol algoritması gibidir; bir tepeden her bir tepeye olan en kısa yolları bulur.

Bir grafın bağlantılı olup olmadığını ve bağlantılı bileşenlerinin tümünü bulmak için DFS (derinlik öncelikli arama) ya da BFS (genişlik öncelikli arama) kullanılır ve işlem $O(n + m)$ zamanda tamamlanır.

3.3 En Kısa Yol Algoritmaları

Ağlarla ilgili en önemli problemlerden birisi de iki tepe arasındaki en az maliyetle gidilebilen bir yolun varlığını belirleme problemi olup bu problem literatürde en kısa yol problemi olarak yer alır. En kısa yol problemi çok geniş kapsamlı ulaşım problemlerinin çözümüne uygulanmaktadır. Herhangi bir tepeden diğer tüm tepelere veya herbir tepeden diğer tüm tepelere olan en kısa yolların belirlenmesi için birçok algoritma geliştirilmiş olup bunlardan en ünlüleri, **Dijkstra** (bir tepeden diğer tüm tepelere en kısa yollar) ve **Floyd** (tüm tepeler için en kısa yollar) olarak verilebilir. Bu algoritmaların bazıları bir tepeden diğer tüm tepelere olan en kısa yolu bulurken, bazıları da, daha genel olarak, herbir tepeden diğer tüm tepelere en kısa yolları belirler. En kısa yolu bulan algoritmalar şöyle sınıflandırılabilir: tek bir hedefe en kısa yollar, bir başlangıç tepesinden en kısa yollar, iki tepe arasındaki en kısa yollar ve tüm tepeler arasındaki en kısa yollar. Bir başlangıç noktasından tüm tepelere en kısa yolu bulan algoritmalar, her defasında bir başlangıç tepesi verilerek N kez çalıştırıldığında genel algoritmalar gibi herbir tepeden diğer tüm tepelere en kısa yolları bulmak için kullanılabilir. Ancak böylesi durumlarda toplam zaman maliyeti/karmaşıklığı, grafın

seyrek/yoğun olmasına göre artabilir. Çoğunlukla yürütülme süresi uzar.

Bu bölümde, en kısa yol probleminin çözümünde kullanılan algoritmalardan Dijkstra ve Floyd algoritmaları hakkında teorik bilgi verilmiştir.

3.3.1 Dijkstra Algoritması

Dijkstra algoritması en kısa yol problemlerinin çözümü için en etkin algoritmalardan biridir. Belirli bir başlangıç noktasına göre en kısa yolu belirleyen bir algoritmadır; bir başlangıç tepesinden, diğer tüm tepelere olan en kısa yolu belirler. Ağırlıklı ve yönlü graflar için geliştirilmiş olup graf üzerindeki herbir ayrıtın ağırlığı sıfır veya sıfırdan büyük bir sayı olmalıdır. Ayrıtlara verilen ağırlıklar uzaklık, maliyet ve zaman gibi kriterleri göstermek için kullanılabilir. Bu algoritma en kısa yolu belirlerken Greedy yaklaşımını kullanır. Bu algoritmanın zaman karmaşıklığı $O(m \log n)$ olarak hesaplanmıştır.

3.3.2 Floyd Algoritması

Floyd Algoritması, graf üzerindeki herbir tepe için diğer tepelere olan en kısa yolları ve bu yolların uzaklıklarını bulmak için kullanılan bir algoritmadır. En kısa yolu bulmak için en genel algoritma Floyd'un algoritmasıdır. Grafın bitişiklik matrisi şeklinde tutulması durumunda bu algoritma $O(n^3)$ karmaşıklığında olmaktadır.

Floyd Algoritması

for $i = 1$ to n

for $j = 1$ to n

if $A[i, j] \neq 0$ then $D[i, j] = A[i, j]$

else $D[i, j] = \infty$

repeat

repeat

for $k = 1$ to n

for $i = 1$ to n

for $j = 1$ to n

if $(D[i, k] + D[k, j] < D[i, j])$ then $D[i, j] = D[i, k] + D[k, j]$

repeat

repeat

repeat

3.4 Tepe Birleştirilmişlik Algoritmalarının Gelişimi

Grafların tepe birleştirilmişliğinin yani kısaca $\kappa(G)$'nin hesaplanabilmesi için yıllardır birçok algoritma geliştirilmektedir. Bu algoritmaların pek çoğu, maksimum akış problemini çözerek $\kappa(G)$'yi hesaplar. Diğer bir anlatımla, bu algoritmalar bir maksimum akış alt programına çağrı yaparak birleştirilmişliği hesaplar. Bu algoritmaların hesabındaki en önemli kısım bu çağrılardır ve mümkün olduğunca az sayıda maksimum akış oluşturarak işlemin tamamlanması için denemeler yapılmaktadır.

Even ve Tarjan maksimum akışı temel alan birleştirilmişlik algoritmalarını ilk sunanlar arasındadır. Diğer sonuçlar Esfahanian ve Hakimi, Matula, Mansour ve Schieber ile Henzinger ve Rao'nun çalışmalarıyla elde edilmiştir. $\kappa(G)$'nın gerçek değerini hesaplamadan bu değerin beklenen değerden büyük olup olmamasının belirlenmesi problemi Tarjan, Mansour, Schieber ve Gabow tarafından çalışılmıştır. Bu bölümde komşu olmayan u ve v tepe çiftleri için maksimum-akış problemlerinin çözümünün indirgenmesiyle tepe birleştirilmişliğin hesabının algoritmalar yardımıyla nasıl yapılacağı üzerinde durulacaktır. (Even and Tarjan, 1975), (Even, 1979), (Kammer and Taubig, 2004).

Algoritma 1

Girdi: $G = (V, E)$ grafında komşu olmayan bir çift u, v tepesi
Çıktı: $\kappa_G(u, v)$'nin hesabı

1. Her $xy \in E$ ayrıtını (x, y) ve (y, x) şeklinde ayırarak G_1 yönlü grafını çağır.

2. G'deki u ve v dışındaki her w tepesi için, w'yi w_1 ve w_2 şeklinde iki yeni tepeye ayır ve (w_1, w_2) şeklinde yeni bir ayrıt oluştur. G'deki w'ye gelen bütün ayrıtları w_1'e bağla ve benzer şekilde G'deki w'den çıkan bütün ayrıtları da w_2'ye bağla.

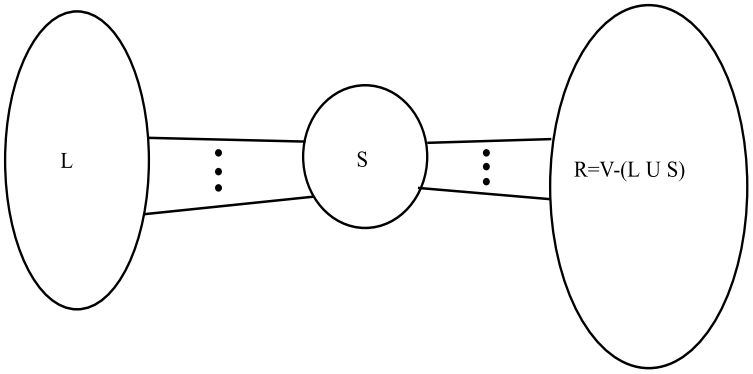
3. u'yu başlangıç tepe ve v'yi bitiş tepe olarak ata.

4. Her bir tepenin kapasitesine 1 ata ve sonuç ağı H'yi çağır.

5. H'deki f fonksiyonu için bir maksimum akış bul.

6. $\kappa(u, v)$ kümesi, f'deki toplam akışa eşitse dur.

Yukarıdaki algoritmanın zaman karmaşıklığı $O(mn^{2/3})$'tür. Maksimum akış programına girebilmek için yukarıdaki algoritma alt program olarak kullanılır ve birbirine komşu olmayan bütün tepeler için $\kappa(u, v)$ hesaplanır. Bu işlemde algoritma $\frac{n(n-1)}{2} - m$ kez çağrılır. Bu yüzden daha az sayıda maksimum akış çağrılarak $\kappa(u, v)$'nin hesaplanmasını sağlayan algoritmalar aranmıştır. Bir sonraki grafta G'den herhangi bir S tepe kesimi alalım (G'nin tam graf olmadığını ve izole tepesiz olduğunu varsayalım).



Şekil 3.1: G=(V,E) grafının minimum tepe kesim kümesi S.

Yukarıdaki şekilde L kümesi, $G - S$'nin bileşenlerinden birinin tepe kümesidir ve R de, $G - S$'nin diğer bütün tepe kümelerinin birleşimidir. G grafındaki bir $u \in L$ tepesi ve bir $v \in R$ tepesi için $\kappa(u, v) = \kappa(G)$ eşitliği açıktır. Rastgele bir u tepesinin seçimi yapıldığında ve $\kappa(G) = min\{\kappa(u, v) | v \in V - \{u\}$, G'deki u ve v komşu tepeler değil$\}$ hesaplansın. Bununla birlikte yukarıdaki bağlantıyı gerçekleştirmek için G grafı u tepesini içermeyen en küçük tepe kesimine sahip olmak zorundadır.

Herhangi bir G grafı için $\kappa(G) \leq \delta(G)$ vardır. Eğer $X \subset V$ kümesi $|X| > \delta$ şartıyla alınırsa, G'nin içindeki her bir S tepe kesimi için, X kümesinin S'de olmayan en az bir tepesi mevcuttur. Bu durumda $\kappa(G)$ şöyle hesaplanabilir:

$$\kappa(G) = min\{min\{\kappa(u, v) | v \in V - \{u\}, \text{G'deki v, u'ya komşu değil}\} | u \in X\}$$

Even ve Tarjan, $X \subset V$ ile $|X| = \kappa + 1$'in hesaplanmasının $\kappa(u, v)$'nin en küçük değerini bulmada yeterli olacağını sunmuşlardır. (Even and Tarjan, 1975)

Algoritma 2

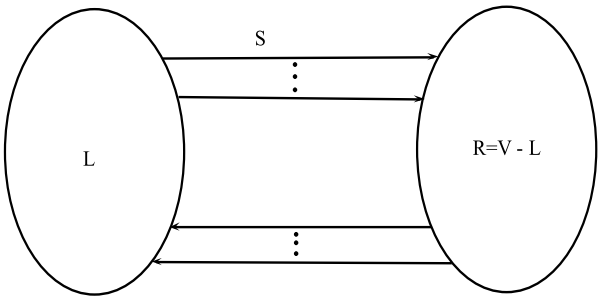
Girdi: $G = (V, E)$ grafı

Çıktı:κ_G'nin hesabı.

1. $i \to 1$, $N \to n - 1$ ata ve $V = \{u_1, u_2, \ldots, u_n\}$.
2. Herbir j için $j = i + 1, i + 2, \ldots, n$.
   - Eğer $i > N$ ise Adım 4'e git.
   - Eğer G'deki u_i ve u_j'ler komşu değilse $\kappa(u_i, u_j)$'yi Algoritma 1'den hesapla ve $N \to min\{N, \kappa(u_i, u_j)\}$ ata.
3. $i \to i + 1$ ata ve Adım 2'ye git.
4. $\kappa(G) \to N$ ata, dur.

Yukarıdaki algoritma $O((n-\delta-1)\kappa)$ maksimum akış çağrısına sahiptir. Bununla birlikte, bir sonraki gözlemde κ'nın çözümlenmesi için maksimum akışa birçok çağrı yapılır.

Keyfi bir $u \in V(G)$ tepesi seçilsin ve aşağıda yer alan Şekil 3.1'de bu tepenin durumu incelensin. u tepesini içermeyen en küçük tepe ayıracı S kümesi ise $\kappa(G) = \{\kappa(u, v)|v \in V - \{u\}$, G grafındaki v tepesi u tepesine komşu değil$\}$ elde edilir.



Şekil 3.2: G=(V,E) yönlü grafının minimum tepe kesim kümesi S.

Diğer taraftan, eğer u tepesi her minimum tepe ayıracı S içinde bulunuyorsa, bu u tepesine komşu en az bir tepe çiftinin, S kümesinin dışında olması gerektiğini gösterir ve bu durumda da,

$$\kappa(G) = min\{\kappa(x, y)|x, y \in A(u), x \text{ ve } y \text{ tepeleri G grafında komşu değil}\}$$

elde edilir. Rastgele seçilen bir u tepesi için yukarıdaki durumların doğruluğunu göstermek için her iki durumu birden ele alan algoritma aşağıda verilmiştir.

Algoritma 3

Girdi: $G = (V, E)$ grafı

Çıktı:$\kappa(G)$'nin değeri.

1. Minimum dereceli herhangi bir u tepesi seçilir.
2. $k_1 = min\{\kappa(u, v)|v \in V - \{u\}$, G'deki v, u'ya komşu değil$\}$ hesaplanır.
3. $k_2 = min\{\kappa(x, y)|x, y \in A(u)$, G'deki x, y'ye komşu değil$\}$ hesaplanır.
4. $\kappa(G) \rightarrow min\{k_1, k_2\}$'yı ata, dur.

Yukarıdaki algoritma maksimum akışı $O(n - \delta - 1 + \frac{\delta(\delta - 1)}{2}$ kez çağırır. Yukarıdaki algoritmalarla ilgili daha ayrıntılı bilgiye (Esfahanian and Hakimi, 1984) makalesinden erişilebilir. $\kappa(G)$'nin hesaplanması için üzerinde çalışılmış bu algoritmalar maksimum akışa dayandırılmasına rağmen, araştırmacılar diğer yöntemleri de denemişledir. Bir grafın k-tepesiyle bağlantılı olup olmadığını anlamak için de bazıları maksimum akışa dayandırılan, bazıları da maksimum akışa dayandırılmayan algoritmalar geliştirilmiştir. Aşağıdaki tablo, birleştirilmişlikle ilgili algoritmaların bir özetini vermektedir.

Karar	Yazar	Karmaşıklık	Karar
$\kappa = 2$	Tarjan (1972)	$O(m+n)$	DFS kullanır.
$\kappa = 3$	Hopcroft & Tarjan (1973)	$O(m+n)$	Üç Bağlantılı Bileşenleri kullanır.
κ	Even & Tarjan (1975)	$O(\kappa(n-\delta-1)mn^{2/3})$	Maksimum Akış kullanır.
$\kappa = k$	Even (1975)	$O(kn^3)$	Maksimum Akış kullanır.
κ	Galil (1980)	$O(min\{\kappa, n^{2/3}\}mn)$	Maksimum Akış kullanır.
$\kappa = k$	Galil (1980)	$O(min\{k, n^{1/2}\}kmn)$	Maksimum Akış kullanır.
κ	Becker, et el. (1982)		Olasılığa Dayalı Algoritma kullanır.
κ	Esfahanian & Hakimi (1984)	$O((n-\delta-1+\delta(\delta-1)/2)mn^{2/3})$	Maksimum Akış kullanır.
$\kappa = 4$	Kanevsky & Ramachandran (1991)	$O(n^2)$	
$\kappa = k$	Cheriyan & Thurimella (1991)	$O(k^3n^2)$	
κ	Henzinger & Rao (1996)	$O(\kappa mn \log n)$	Rastgele Seçim Algoritması

Çizelge 3.1: Tepe Birleştirilmişlik Algoritmalarının Tarihsel Gelişimi

Sonuç olarak herhangi bir $G = (V, E)$ grafı için bağlantılılığın kontrolü şu aşamalardan oluşur:

1. G grafını, bu grafın temelini oluşturan basit graf ile değiştirelim. Köşegen haricindeki sıfırdan farklı her bir elemanı 1 ve köşegen üzerindeki elemanları da 0 ile değiştirelim.

2. Yeni bir graf oluşturmak için v_1 ile komşu olan $\{v_2, \ldots, v_n\}$ tepelerinden biri olan ilk tepe ile birleştirelim. Yeni tepeyi v_1 ile isimlendirerek grafı G ile gösterelim.

3. 1. adımı tekrar edelim.

4. 2. adım ve 3. adımı v_1 ile v_1'e komşu başka bir tepe kalmayana kadar tekrar edelim.

5. 2. adımdan 4. adıma kadar olan kısmı v_2 ve geri kalan tüm tepeler üzerine uygulayalım. Sonuç grafının izole edilmiş tepelerinin sayısı ilk grafın bağlantılı bileşenlerinin sayısına eşittir.

3.4.1 Tepe Birleştirilmişlik için Yeni Bir Algoritma (TBA)

Çalışmanın bu bölümünde birleştirilmiş bir grafın tepe birleştirilmişlik sayısını veren yeni bir algoritma önerilmiştir. Bu algoritmanın diğer algoritmalardan farkı graftan atılan tepelerin kümesini de vermesidir.

TBA Algoritması

Girdi: Aşikar olmayan bağlantılı bir $G = (V, E)$ grafı.

Çıktı: $\kappa(G)$'nin değeri.

1. Grafın bitişiklik matrisini oku ve tepe derecelerini hesapla, eğer matristeki 1'lerin sayısı $n(n-1)$ ise (yani graf tam graf ise) $\kappa(G) = (n-1)$ olarak ata ve silinen tepeler kümesine ilk tepe hariç diğer tepeleri ata. Adım 11'e git.

2. Eğer derecesi 1 olan tepe varsa $\kappa(G) = 1$ olarak ata ve silinen tepeler kümesine derecesi 1 olan tepenin komşusunu ata. Adım 11'e git.

3. Grafın en düşük dereceli tepesini kaynak tepe olarak ata.

4. Kaynak tepeye komşu olmayan tepeyi hedef tepe olarak ata.

5. Yol=0 al.

6. $\kappa(G)$=0 ve silinen tepeler kümesini $\emptyset$ al.

7. Dijkstra algoritması ile kaynak ve hedef tepe arasındaki en kısa yolu bul ve yolu 1 arttır. Kaynak ile hedef arasında yol yok ise Adım 10'a git.

8. $\kappa(G)$'yi 1 arttır. yol=1 ise hedeften önceki tepeyi sil, değilse hedefle kaynak arasındaki maksimum tepe tekrarsız yollar ile birleştirilmişliği hesapla, en kısa yoldaki hedeften kaynağa olan yoldaki tepeleri ara hedef olarak tek tek ata-yarak ele al. Kaynakla ara hedefler arasındaki lokal birleştirilmişliği sırasıyla hesapla. Eğer hedefle kaynak arasındaki lokal birleştirilmişlik önceki lokal birleştirilmişlikten farklıysa o tepeyi sil, eğer birleştirilmişlik hiçbir tepede değişmezse ve kaynak ile hedeften başka silinecek tepe kalmadıysa kaynak-tan bir sonraki tepeyi sil. Silinen tepeyi silinen tepeler kümesine at. Grafta ve matriste silinen tepeyi güncelle. Adım 7'ye git.

9. Hedef tepe ilk hedef tepeyse $\kappa(G)$ ve silinen tepeler kümesini sakla, değilse ve yeni $\kappa(G)$ eskisinden küçükse $\kappa(G)$'yi ve silinen tepeler kümesini güncelle.

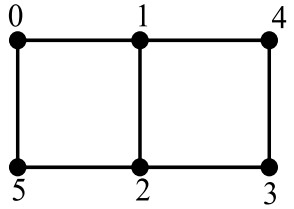
10. Hedef tepeyi bir sonraki kaynak tepeye komşu olmayan tepe al. Adım 5'e git.

11. $\kappa(G)$'yi ve silinen tepeler kümesini yazdır.

12. Son.

Örnek 3.4.1.

TBA ile aşağıdaki G grafına ait tepe birleştirilmişlik sayısı ve silinecek tepeler kümesi şu şekilde bulunur:



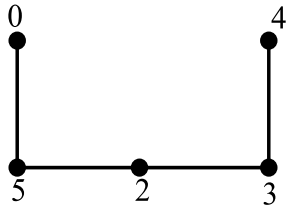
Şekil 3.3: 6 tepeli G grafı.

Çizelge 3.2: G grafına ait Bitişiklik Matrisi

	0	1	2	3	4	5
0	-	1	0	0	0	1
1	1	-	1	0	1	0
2	0	1	-	1	0	1
3	0	0	1	-	1	0
4	0	1	0	1	-	0
5	1	0	1	0	0	-

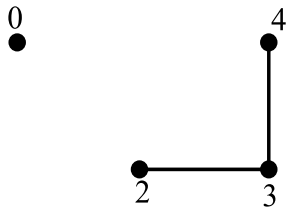
Öncelikle G grafının bitişiklik matrisi okunarak en düşük dereceli tepeler 0, 3, 4 ve 5 olarak bulunur. Bitişiklik matrisindeki sıraya göre bu tepelerden 0 tepesi kaynak tepe olur. Kaynak tepeye yani 0'a komşu olmayan ilk tepe 2 tepesi olup bu tepe hedef tepesi olarak belirlenir.

$0 - 2$ arasında yol $= 0$, $\kappa(G) = 0$ ve silinecek tepeler kümesi $\emptyset$ olarak seçilir. 0 ile 2 tepeleri arasındaki en kısa yollardan biri olan 012 yolu alınır, yol $= 1$ ve $\kappa(G) = 1$ olarak yeniden düzenleme yapılır ve hedeften önceki tepe yani 1 tepesi silinir, silinecek tepeler kümesi $= \{1\}$ olarak bulunur. 1 tepesi silindikten sonra elde edilen graf şu şekilde olur:



Şekil 3.4: 1 tepesi silinmiş G grafı.

1 tepesi silindikten sonra elde edilen grafta 0 ile 2 tepeleri arasındaki en kısa yollardan biri olan 052 yolu alınır, yol $= 2$ ve $\kappa(G) = 2$ olarak yeniden düzenleme yapılır ve $0 - 2$ arasında 1 tane tepe ayrık yol bulunur, bu durumda $0 - 5$ arasındaki tepe ayrık yol da 1 olarak bulunduğundan kaynaktan bir sonraki tepe olan 5 tepesi silinir. Silinecek tepeler kümesi $= \{1, 5\}$ olarak güncellenir. 5 tepesi silindikten sonra elde edilen graf şu şekilde olur:

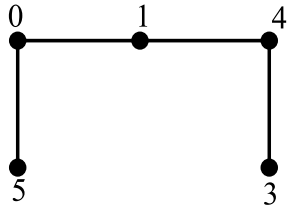


Şekil 3.5: 1 ve 5 tepeleri silinmiş G grafı.

5 tepesi de silindikten sonra 0 ile 2 tepeleri arasında yol bulunamadığından algoritma sonlanır ve $\kappa(G) = 2$ ve silinecek tepeler kümesi $= \{1, 5\}$ olarak bulunur.

Hedef tepe, kaynak tepeye komşu olmayan yeni bir tepe olarak yeniden belirlenir, bu tepe G grafı için 3 tepesidir. $0 - 3$ arasında yol $= 0$, $\kappa(G) = 0$ ve silinecek tepeler kümesi $\emptyset$ olarak seçilir. 0 ile 3 tepeleri arasındaki en kısa yollardan biri olan 0123 yolu alınır, yol $= 1$ ve $\kappa(G) = 1$ olarak yeniden düzenleme yapılır ve hedeften önceki tepe yani 2 tepesi silinir, silinecek tepeler kümesi $= \{2\}$ olarak bulunur. 2 tepesi silindikten sonra elde edilen graf şu şekilde olur:

2 tepesi silindikten sonra elde edilen grafta 0 ile 3 tepeleri arasındaki en kısa yollardan biri olan 0143 yolu alınır, yol $= 2$ ve $\kappa(G) = 2$ olarak yeniden düzenleme yapılır ve $0-3$ arasında 1 tane tepe ayrık yol bulunur, bu durumda $0-1-4$ arasındaki



Şekil 3.6: 2 tepesi silinmiş G grafı.

tepe ayrık yol da 1 olarak bulunur, en son durumda $0-1$ arasındaki tepe ayrık yol da 1 olarak bulunduğundan kaynaktan bir sonraki tepe olan 1 tepesi silinir. Silinecek tepeler kümesi $=\{1,2\}$ olarak güncellenir. 1 tepesi silindikten sonra elde edilen graf şu şekilde olur:

Şekil 3.7: 1 ve 2 tepeleri silinmiş G grafı.

1 tepesi de silindikten sonra 0 ile 3 tepeleri arasında yol bulunamadığından algoritma sonlanır ve $\kappa(G)=2$ ve silinecek tepeler kümesi $=\{1,2\}$ olarak bulunur.

Hedef tepe, kaynak tepeye komşu olmayan yeni bir tepe olarak son kez belirlenir, bu tepe G grafı için 4 tepesidir. $0-4$ arasında yol $=0$, $\kappa(G)=0$ ve silinecek tepeler kümesi $\emptyset$ olarak seçilir. 0 ile 4 tepeleri arasındaki en kısa yollardan biri olan 014 yolu alınır, yol $=1$ ve $\kappa(G)=1$ olarak yeniden düzenleme yapılır ve hedeften önceki tepe yani 1 tepesi silinir, silinecek tepeler kümesi $=\{1\}$ olarak bulunur. 1 tepesi silindikten sonra elde edilen graf Şekil 3.4'deki gibidir.

1 tepesi silindikten sonra elde edilen grafta 0 ile 4 tepeleri arasındaki en kısa yollardan biri olan 05234 yolu alınır, yol $=2$ ve $\kappa(G)=2$ olarak yeniden düzenleme yapılır ve $0-4$ arasında 1 tane tepe ayrık yol bulunur, bu durumda $0-3$ arasındaki tepe ayrık yol da 1 olarak bulunur, $0-2$ arasındaki tepe ayrık yol 1 ve en son

$0 - 5$ arasındaki tepe ayrık yol 1 olarak bulunduğundan kaynaktan bir sonraki tepe olan 5 tepesi silinir. Silinecek tepeler kümesi $= \{1, 5\}$ olarak güncellenir. 5 tepesi silindikten sonra elde edilen graf Şekil 3.5'deki gibi bulunur.

5 tepesi de silindikten sonra 0 ile 4 tepeleri arasında yol bulunamadığından algoritma sonlanır ve $\kappa(G) = 2$ ve silinecek tepeler kümesi $= \{1, 5\}$ olarak bulunur.

Kaynak tepeden, bu tepeye komşu olmayan diğer tüm tepelere yapılan incelemeler sonucunda $\kappa(G) = 2$ ve ilk bulunan silinecek tepeler kümesi $= \{1, 5\}$ bu grafın tepe birleştirilmişliği hakkında bilgi verir.

4. GRAFLARDA TEPE BİRLEŞTİRİLMİŞLİK PROBLEMİNİN MATEMATİKSEL MODELLERİ

4.1 Problemin Tanımı

En az $(k+1)$ tepeli bir grafın her u, v tepe çifti k tepe ayrık yolla birleştirilebiliyorsa bu graf k tepe birleştirilmiş graftır. k'nın en büyük değerini bulma problemi grafın tepe birleştirilmişlik problemine karşılık gelir. Bu sayı grafı birleştirilmemiş yapmak için graftan atılması gereken en az tepe sayısıdır. Bu bölümde problemin matematiksel modelleri verilmiştir.

4.2 Problemin En Kısa Yol Problemine Dayanan Matematiksel Modeli

$G = (V, E)$ grafında ($|V| = n$, $|E| = m$) **i**. tepe ile **j.** tepeyi birleştiren ayrıtı **(i,j)** ile gösterelim. Varsayalım ki G grafı , $A = \{a_{ij}\}$, $(i = \overline{1,n},\ j = \overline{1,n})$ bitişiklik matrisi şeklinde verilmiştir.

Burada,

$$a_{ij} = \begin{cases} 1, & \text{eğer i ile j tepeleri arasında ayrıt varsa;} \\ 0, & \text{diğer durumlarda.} \end{cases}$$

Aşağıda x_{ij} $(i = \overline{1,n},\ j = \overline{1,n})$ ve x_i $(i = \overline{1,n})$ değişkenlerini tanımlayalım :

$$x_{ij} = \begin{cases} a_{ij}, & i\text{'den } j\text{'ye geçilirse;} \\ 0, & \text{diğer durumlarda.} \end{cases}$$

$$x_i = \begin{cases} M, & i \text{ tepesi siliniyorsa;} \\ 1, & \text{diğer durumlarda.} \end{cases}$$

Burada $M > \dfrac{n(n-1)}{2}$ koşulunu sağlayan büyük bir tamsayıdır.

p başlangıç tepesi ve q son tepe olmak üzere $(p = \overline{1, n-1},\ q = \overline{p+1, n})$ tüm p, q tepe çiftleri arasındaki en kısa yolları aşağıdaki şekilde modelleyebiliriz:

p başlangıç tepesinden yalnız bir tepeye geçileceğinden,

$$\sum_{i=1}^{n} x_{pi} = 1, (i \neq p)$$

ve son q tepesine yalnız bir tepeden gelineceğinden,

$$\sum_{i=1}^{n} x_{iq} = 1, (i \neq q)$$

sağlanmalıdır.

Her tepeye ya bir giriş bir çıkış olacağından, ya da giriş ve çıkış olmayacağından,

$$\sum_{i=1}^{n} x_{ik} = \sum_{j=1}^{n} x_{kj} \quad (k \neq p, k \neq q, k = \overline{1, n})$$

kısıtları gerçekleşmelidir.

Yukarıdaki koşullar altında en kısa $< p, q >$ yolunun uzunluğunu

$$min_{(i,j)} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{x_{ij} x_i x_j}{x_p x_q} = M_{pq}$$

şeklinde gösterilebilir. p başlangıç tepesi ve q son tepenin silinmesi $< p, q >$ yolunu 2 farklı yola parçalamadığı için M_{pq}'nün ifadesinde bu x_p ve x_q'ya bölünerek gözönüne alınır.

$$\sum_{p=1}^{n-1} \sum_{q=p+1}^{n} M_{pq} \geq M$$

kısıtı en az bir silinmiş tepeden geçen yolları sağlayacaktır.

Yukarıdaki açıklamalarla $p = \overline{1, n-1},\ q = \overline{p+1, n}$ olmak üzere Graflarda Tepe Birleştirilmişlik Problemi aşağıdaki şekilde modellenebilir:

T1) $x_i = 1 \vee M, \quad i = \overline{1, n}$

T2) $x_{ij} = 0 \vee a_{ij},\ i = \overline{1,n},\ j = \overline{1,n}$ olmakla tüm tepe çiftleri arasındaki en kısa yollar M_{pq} $(p = \overline{1,n-1}, q = \overline{p+1,n})$ aşağıdaki kısıtlar altında hesaplanır.

T3) $\sum_{i=1}^{n} x_{pi} = 1\ (i \neq p)$

T4) $\sum_{i=1}^{n} x_{iq} = 1\ (i \neq q)$

T5) $\sum_{i=1}^{n} x_{ik} = \sum_{j=1}^{n} x_{kj}\ (k \neq p, k \neq q, k = \overline{1,n})$

T6) $min \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{x_{ij}x_i x_j}{x_p x_q} = M_{pq}\ (i \neq j)$

Bu yollardan en az bir tanesinin silinmiş tepeden geçmesi için,

T7) $\sum_{p=1}^{n-1}\sum_{q=p+1}^{n} M_{pq} \geq M$ koşulu sağlanmalıdır. Yukarıdaki koşullar altında,

T8) $\sum_{i=1}^{n} x_i \rightarrow min$

grafın bütünlüğünü bozacak en az sayıda silinecek tepelerin sayısını verecektir.

4.2.1 Modelin Örneklenmesi

Bu bölümde, tepe birleştirilmişlik probleminin matematiksel modeli 4 tepeli herhangi bir graf üzerinde incelenecektir.

Kısıtlar:

T1) $x_i = 1 \vee M,\ \ i = \overline{1,n}$

T2) $x_{ij} = 0 \vee a_{ij},\ i = \overline{1,n},\ j = \overline{1,n}$

T3) $\sum_{i=1}^{n} x_{pi} = 1,\ (i \neq p)$

T4) $\sum_{i=1}^{n} x_{iq} = 1,\ (i \neq q)$

T5) $\sum_{i=1}^{n} x_{ik} = \sum_{j=1}^{n} x_{kj},\ (k \neq p, k \neq q, k = \overline{1,n})$

T6) $min \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{x_{ij}x_i x_j}{x_p x_q} = M_{pq},\ (i \neq j)$

p=1 ve q=2 için

3. kısıt: $x_{12} + x_{13} + x_{14} = 1$

4. kısıt: $x_{12} + x_{32} + x_{42} = 1$

5. kısıt: $k = 3$ için $x_{13} + x_{23} + x_{43} = x_{31} + x_{32} + x_{34}$

$k = 4$ için $x_{14} + x_{24} + x_{34} = x_{41} + x_{42} + x_{43}$

6. kısıt:

$$min \left(\begin{array}{l} \frac{x_{12}x_1x_2}{x_1x_2} + \frac{x_{13}x_1x_3}{x_1x_2} + \frac{x_{14}x_1x_4}{x_1x_2} + \frac{x_{21}x_2x_1}{x_1x_2} + \frac{x_{23}x_2x_3}{x_1x_2} + \frac{x_{24}x_2x_4}{x_1x_2} + \\ \\ \frac{x_{31}x_3x_1}{x_1x_2} + \frac{x_{32}x_3x_2}{x_1x_2} + \frac{x_{34}x_3x_4}{x_1x_2} + \frac{x_{41}x_4x_1}{x_1x_2} + \frac{x_{42}x_4x_2}{x_1x_2} + \frac{x_{43}x_4x_3}{x_1x_2} \end{array} \right) = M_{12}$$

p=1 ve q=3 için

3. kısıt: $x_{12} + x_{13} + x_{14} = 1$

4. kısıt: $x_{13} + x_{23} + x_{43} = 1$

5. kısıt: $k = 2$ için $x_{12} + x_{32} + x_{42} = x_{21} + x_{23} + x_{24}$

$k = 4$ için $x_{14} + x_{24} + x_{34} = x_{41} + x_{42} + x_{43}$

6. kısıt:

$$min \left(\begin{array}{l} \frac{x_{12}x_1x_2}{x_1x_3} + \frac{x_{13}x_1x_3}{x_1x_3} + \frac{x_{14}x_1x_4}{x_1x_3} + \frac{x_{21}x_2x_1}{x_1x_3} + \frac{x_{23}x_2x_3}{x_1x_3} + \frac{x_{24}x_2x_4}{x_1x_3} + \\ \\ \frac{x_{31}x_3x_1}{x_1x_3} + \frac{x_{32}x_3x_2}{x_1x_3} + \frac{x_{34}x_3x_4}{x_1x_3} + \frac{x_{41}x_4x_1}{x_1x_3} + \frac{x_{42}x_4x_2}{x_1x_3} + \frac{x_{43}x_4x_3}{x_1x_3} \end{array} \right) = M_{13}$$

p=1 ve q=4 için

3. kısıt: $x_{12} + x_{13} + x_{14} = 1$

4. kısıt: $x_{14} + x_{24} + x_{34} = 1$

5. kısıt: $k = 2$ için $x_{12} + x_{32} + x_{42} = x_{21} + x_{23} + x_{24}$

$k = 3$ için $x_{13} + x_{23} + x_{43} = x_{31} + x_{32} + x_{34}$

6. kısıt:

$$min \left(\begin{array}{l} \frac{x_{12}x_1x_2}{x_1x_4} + \frac{x_{13}x_1x_3}{x_1x_4} + \frac{x_{14}x_1x_4}{x_1x_4} + \frac{x_{21}x_2x_1}{x_1x_4} + \frac{x_{23}x_2x_3}{x_1x_4} + \frac{x_{24}x_2x_4}{x_1x_4} + \\ \\ \frac{x_{31}x_3x_1}{x_1x_4} + \frac{x_{32}x_3x_2}{x_1x_4} + \frac{x_{34}x_3x_4}{x_1x_4} + \frac{x_{41}x_4x_1}{x_1x_4} + \frac{x_{42}x_4x_2}{x_1x_4} + \frac{x_{43}x_4x_3}{x_1x_4} \end{array} \right) = M_{14}$$

p=2 ve q=3 için

3. kısıt: $x_{21} + x_{23} + x_{24} = 1$

4. kısıt: $x_{13} + x_{23} + x_{43} = 1$

5. kısıt: $k = 1$ için $x_{21} + x_{31} + x_{41} = x_{12} + x_{13} + x_{14}$

$k = 4$ için $x_{14} + x_{24} + x_{34} = x_{41} + x_{42} + x_{43}$

6. kısıt:

$$min \left(\begin{array}{l} \dfrac{x_{12}x_1x_2}{x_2x_3} + \dfrac{x_{13}x_1x_3}{x_2x_3} + \dfrac{x_{14}x_1x_4}{x_2x_3} + \dfrac{x_{21}x_2x_1}{x_2x_3} + \dfrac{x_{23}x_2x_3}{x_2x_3} + \dfrac{x_{24}x_2x_4}{x_2x_3} + \\ \\ \dfrac{x_{31}x_3x_1}{x_2x_3} + \dfrac{x_{32}x_3x_2}{x_2x_3} + \dfrac{x_{34}x_3x_4}{x_2x_3} + \dfrac{x_{41}x_4x_1}{x_2x_3} + \dfrac{x_{42}x_4x_2}{x_2x_3} + \dfrac{x_{43}x_4x_3}{x_2x_3} \end{array} \right) = M_{23}$$

p=2 ve q=4 için

3. kısıt: $x_{21} + x_{23} + x_{24} = 1$

4. kısıt: $x_{14} + x_{24} + x_{34} = 1$

5. kısıt: $k = 1$ için $x_{21} + x_{31} + x_{41} = x_{12} + x_{13} + x_{14}$

$k = 3$ için $x_{13} + x_{23} + x_{43} = x_{31} + x_{32} + x_{34}$

6. kısıt:

$$min \left(\begin{array}{l} \dfrac{x_{12}x_1x_2}{x_2x_4} + \dfrac{x_{13}x_1x_3}{x_2x_4} + \dfrac{x_{14}x_1x_4}{x_2x_4} + \dfrac{x_{21}x_2x_1}{x_2x_4} + \dfrac{x_{23}x_2x_3}{x_2x_4} + \dfrac{x_{24}x_2x_4}{x_2x_4} + \\ \\ \dfrac{x_{31}x_3x_1}{x_2x_4} + \dfrac{x_{32}x_3x_2}{x_2x_4} + \dfrac{x_{34}x_3x_4}{x_2x_4} + \dfrac{x_{41}x_4x_1}{x_2x_4} + \dfrac{x_{42}x_4x_2}{x_2x_4} + \dfrac{x_{43}x_4x_3}{x_2x_4} \end{array} \right) = M_{24}$$

p=3 ve q=4 için

3. kısıt: $x_{31} + x_{32} + x_{34} = 1$

4. kısıt: $x_{14} + x_{24} + x_{34} = 1$

5. kısıt: $k = 1$ için $x_{21} + x_{31} + x_{41} = x_{12} + x_{13} + x_{14}$

$k = 2$ için $x_{12} + x_{32} + x_{42} = x_{21} + x_{23} + x_{24}$

6. kısıt:

$$min \left(\begin{array}{l} \dfrac{x_{12}x_1x_2}{x_3x_4} + \dfrac{x_{13}x_1x_3}{x_3x_4} + \dfrac{x_{14}x_1x_4}{x_3x_4} + \dfrac{x_{21}x_2x_1}{x_3x_4} + \dfrac{x_{23}x_2x_3}{x_3x_4} + \dfrac{x_{24}x_2x_4}{x_3x_4} + \\ \\ \dfrac{x_{31}x_3x_1}{x_3x_4} + \dfrac{x_{32}x_3x_2}{x_3x_4} + \dfrac{x_{34}x_3x_4}{x_3x_4} + \dfrac{x_{41}x_4x_1}{x_3x_4} + \dfrac{x_{42}x_4x_2}{x_3x_4} + \dfrac{x_{43}x_4x_3}{x_3x_4} \end{array} \right) = M_{34}$$

T7) $\displaystyle\sum_{p=1}^{n-1} \sum_{q=p+1}^{n} M_{pq} \geq M$

7. kısıt: $M_{12} + M_{13} + M_{14} + M_{23} + M_{24} + M_{34} \geq M$

T8) $\sum_{i=1}^{n} x_i \to min$

4.3 Problemin Genel Matematiksel Modeli

$G = (V, E)$ grafında ($|V| = n$, $|E| = m$) **i.** tepe ile **j.** tepeyi birleştiren ayrıtı **(i,j)** ile gösterelim. Varsayalım ki G grafı , $A = \{a_{ij}\}$, ($i = \overline{1,n},\ j = \overline{1,n}$) bitişiklik matrisi şeklinde verilmiştir.

Burada,

$$a_{ij} = \begin{cases} 1, & \text{eğer i ile j tepeleri arasında ayrıt varsa;} \\ 0, & \text{diğer durumlarda.} \end{cases}$$

Aşağıda x_i ($i = \overline{1,n}$) ve x_{ij}^{pq} ($i = \overline{1,n},\ j = \overline{1,n},\ p = \overline{1,n-1},\ q = \overline{p+1,n}$) değişkenlerini tanımlayalım :

$$x_{ij}^{pq} = \begin{cases} a_{ij}, & \text{<}p, q\text{> yolu üzerinde } i\text{'den } j\text{'ye geçilirse;} \\ 0, & \text{diğer durumlarda.} \end{cases}$$

$$x_i = \begin{cases} M, & i \text{ tepesi siliniyorsa;} \\ 1, & \text{diğer durumlarda.} \end{cases}$$

Burada $M > \dfrac{n(n-1)}{2}$ koşulunu sağlayan büyük bir tamsayıdır.

p başlangıç tepesi ve q son tepe olmak üzere tüm p, q tepe çiftleri ($p = \overline{1,n-1},\ q = \overline{p+1,n}$) arasındaki yolları aşağıdaki şekilde modelleyebiliriz:

p başlangıç tepesinden yalnız bir tepeye geçileceğinden,

$$\sum_{i=1}^{n} x_{pi}^{pq} = 1, (i \neq p,\ p = \overline{1,n-1},\ q = \overline{p+1,n})$$

ve son q tepesine yalnız bir tepeden gelineceğinden,

$$\sum_{i=1}^{n} x_{iq}^{pq} = 1, (i \neq q,\ p = \overline{1,n-1},\ q = \overline{p+1,n})$$

sağlanmalıdır.

Her tepeye ya bir giriş bir çıkış olacağından, ya da giriş ve çıkış olmayacağından,

$$\sum_{i=1}^{n} x_{ik}^{pq} = \sum_{j=1}^{n} x_{kj}^{pq} \;\; (k \neq p,\; k \neq q,\; k = \overline{1,n},\; p = \overline{1,n-1},\; q = \overline{p+1,n})$$

kısıtları gerçekleşmelidir.

Yukarıdaki koşullar altında en kısa $< p, q >$ yolunun uzunluğunu

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij}^{pq} x_i x_j = M_{pq}$$

şeklinde gösterilsin.

$$\sum_{p=1}^{n-1} \sum_{q=p+1}^{n} M_{pq} \geq M$$

kısıtı en az bir silinmiş tepeden geçen $< p, q >$ yollarını sağlayacaktır.

Yukarıdaki açıklamalarla Graflarda Tepe Birleştirilmişlik Probleminin Matematiksel Modeli aşağıdaki şekilde yazılabilir:

T1) $\sum_{i=1}^{n} x_{pi}^{pq} = 1,\; (i \neq p,\; p = \overline{1,n-1},\; q = \overline{p+1,n})$

T2) $\sum_{i=1}^{n} x_{iq}^{pq} = 1,\; (i \neq q,\; p = \overline{1,n-1},\; q = \overline{p+1,n})$

T3) $\sum_{i=1}^{n} x_{ik}^{pq} = \sum_{j=1}^{n} x_{kj}^{pq},\; (k \neq p,\; k \neq q,\; k = \overline{1,n},\; p = \overline{1,n-1},\; q = \overline{p+1,n})$

T4) $\sum_{p=1}^{n-1} \frac{1}{x_p} \sum_{q=p+1}^{n} \frac{1}{x_q} \sum_{i=1}^{n} \sum_{j=1}^{n},\; x_{ij}^{pq} x_i x_j \geq M$

T5) $x_{ij}^{pq} = 0 \vee a_{ij},\; (i = \overline{1,n},\; j = \overline{1,n},\; p = \overline{1,n-1},\; q = \overline{p+1,n})$

T6) $x_i = 1 \vee M, \quad i = \overline{1,n}$

T7) $\sum_{i=1}^{n} x_i \rightarrow min$

4.3.1 Modelin Örneklenmesi

Bu bölümde, tepe birleştirilmişlik probleminin genel matematiksel modeli 4 tepeli herhangi bir graf üzerinde incelenecektir.

Kısıtlar:

T1) $\sum_{i=1}^{n} x_{pi}^{pq} = 1, \quad (i \neq p,\ p = \overline{1, n-1},\ q = \overline{p+1, n})$

$x_{12}^{12} + x_{13}^{12} + x_{14}^{12} = 1$ (p=1 ve q=2 için)
$x_{12}^{13} + x_{13}^{13} + x_{14}^{13} = 1$ (p=1 ve q=3 için)
$x_{12}^{14} + x_{13}^{14} + x_{14}^{14} = 1$ (p=1 ve q=4 için)
$x_{21}^{23} + x_{23}^{23} + x_{24}^{23} = 1$ (p=2 ve q=3 için)
$x_{21}^{24} + x_{23}^{24} + x_{24}^{24} = 1$ (p=2 ve q=4 için)
$x_{31}^{34} + x_{32}^{34} + x_{34}^{34} = 1$ (p=3 ve q=4 için)

T2) $\sum_{i=1}^{n} x_{iq}^{pq} = 1, \quad (i \neq q,\ p = \overline{1, n-1},\ q = \overline{p+1, n})$

$x_{12}^{12} + x_{32}^{12} + x_{42}^{12} = 1$ (p=1 ve q=2 için)
$x_{13}^{13} + x_{23}^{13} + x_{43}^{13} = 1$ (p=1 ve q=3 için)
$x_{14}^{14} + x_{24}^{14} + x_{34}^{14} = 1$ (p=1 ve q=4 için)
$x_{13}^{23} + x_{23}^{23} + x_{43}^{23} = 1$ (p=2 ve q=3 için)
$x_{14}^{24} + x_{24}^{24} + x_{34}^{24} = 1$ (p=2 ve q=4 için)
$x_{14}^{34} + x_{24}^{34} + x_{34}^{34} = 1$ (p=3 ve q=4 için)

T3) $\sum_{i=1}^{n} x_{ik}^{pq} = \sum_{j=1}^{n} x_{kj}^{pq}, \quad (k \neq p,\ k \neq q,\ k = \overline{1, n},\ p = \overline{1, n-1},\ q = \overline{p+1, n})$

$p = 1, q = 2$ ve $k = 3$ için $\quad x_{13}^{12} + x_{23}^{12} + x_{43}^{12} = x_{31}^{12} + x_{32}^{12} + x_{34}^{12}$
$p = 1, q = 2$ ve $k = 4$ için $\quad x_{14}^{12} + x_{24}^{12} + x_{34}^{12} = x_{41}^{12} + x_{42}^{12} + x_{43}^{12}$
$p = 1, q = 3$ ve $k = 2$ için $\quad x_{12}^{13} + x_{32}^{13} + x_{42}^{13} = x_{21}^{13} + x_{23}^{13} + x_{24}^{13}$
$p = 1, q = 3$ ve $k = 4$ için $\quad x_{14}^{13} + x_{24}^{13} + x_{34}^{13} = x_{41}^{13} + x_{42}^{13} + x_{43}^{13}$
$p = 1, q = 4$ ve $k = 2$ için $\quad x_{12}^{14} + x_{32}^{14} + x_{42}^{14} = x_{21}^{14} + x_{23}^{14} + x_{24}^{14}$
$p = 1, q = 4$ ve $k = 3$ için $\quad x_{13}^{14} + x_{23}^{14} + x_{43}^{14} = x_{31}^{14} + x_{32}^{14} + x_{34}^{14}$
$p = 2, q = 3$ ve $k = 1$ için $\quad x_{21}^{23} + x_{31}^{23} + x_{41}^{23} = x_{12}^{23} + x_{13}^{23} + x_{14}^{23}$
$p = 2, q = 3$ ve $k = 4$ için $\quad x_{14}^{23} + x_{24}^{23} + x_{34}^{23} = x_{41}^{23} + x_{42}^{23} + x_{43}^{23}$
$p = 2, q = 4$ ve $k = 1$ için $\quad x_{21}^{24} + x_{31}^{24} + x_{41}^{24} = x_{12}^{24} + x_{13}^{24} + x_{14}^{24}$

$p=2, q=4$ ve $k=3$ için $\quad x_{13}^{24}+x_{23}^{24}+x_{43}^{24}=x_{31}^{24}+x_{32}^{24}+x_{34}^{24}$

$p=3, q=4$ ve $k=1$ için $\quad x_{21}^{34}+x_{31}^{34}+x_{41}^{34}=x_{12}^{34}+x_{13}^{34}+x_{14}^{34}$

$p=3, q=4$ ve $k=2$ için $\quad x_{12}^{34}+x_{32}^{34}+x_{42}^{34}=x_{21}^{34}+x_{23}^{34}+x_{24}^{34}$

T4) $\sum_{p=1}^{n-1}\frac{1}{x_p}\sum_{q=p+1}^{n}\frac{1}{x_q}\sum_{i=1}^{n}\sum_{j=1}^{n}x_{ij}^{pq}x_ix_j \geq M$

$$
\begin{aligned}
&\left(\frac{x_{12}x_1x_2}{x_1x_2}+\frac{x_{13}x_1x_3}{x_1x_2}+\frac{x_{14}x_1x_4}{x_1x_2}+\frac{x_{21}x_2x_1}{x_1x_2}+\frac{x_{23}x_2x_3}{x_1x_2}+\frac{x_{24}x_2x_4}{x_1x_2}+\right.\\
&\left.\frac{x_{31}x_3x_1}{x_1x_2}+\frac{x_{32}x_3x_2}{x_1x_2}+\frac{x_{34}x_3x_4}{x_1x_2}+\frac{x_{41}x_4x_1}{x_1x_2}+\frac{x_{42}x_4x_2}{x_1x_2}+\frac{x_{43}x_4x_3}{x_1x_2}\right)+\\
&\left(\frac{x_{12}x_1x_2}{x_1x_3}+\frac{x_{13}x_1x_3}{x_1x_3}+\frac{x_{14}x_1x_4}{x_1x_3}+\frac{x_{21}x_2x_1}{x_1x_3}+\frac{x_{23}x_2x_3}{x_1x_3}+\frac{x_{24}x_2x_4}{x_1x_3}+\right.\\
&\left.\frac{x_{31}x_3x_1}{x_1x_3}+\frac{x_{32}x_3x_2}{x_1x_3}+\frac{x_{34}x_3x_4}{x_1x_3}+\frac{x_{41}x_4x_1}{x_1x_3}+\frac{x_{42}x_4x_2}{x_1x_3}+\frac{x_{43}x_4x_3}{x_1x_3}\right)+\\
&\left(\frac{x_{12}x_1x_2}{x_1x_4}+\frac{x_{13}x_1x_3}{x_1x_4}+\frac{x_{14}x_1x_4}{x_1x_4}+\frac{x_{21}x_2x_1}{x_1x_4}+\frac{x_{23}x_2x_3}{x_1x_4}+\frac{x_{24}x_2x_4}{x_1x_4}+\right.\\
&\left.\frac{x_{31}x_3x_1}{x_1x_4}+\frac{x_{32}x_3x_2}{x_1x_4}+\frac{x_{34}x_3x_4}{x_1x_4}+\frac{x_{41}x_4x_1}{x_1x_4}+\frac{x_{42}x_4x_2}{x_1x_4}+\frac{x_{43}x_4x_3}{x_1x_4}\right)+\\
&\left(\frac{x_{12}x_1x_2}{x_2x_3}+\frac{x_{13}x_1x_3}{x_2x_3}+\frac{x_{14}x_1x_4}{x_2x_3}+\frac{x_{21}x_2x_1}{x_2x_3}+\frac{x_{23}x_2x_3}{x_2x_3}+\frac{x_{24}x_2x_4}{x_2x_3}+\right.\\
&\left.\frac{x_{31}x_3x_1}{x_2x_3}+\frac{x_{32}x_3x_2}{x_2x_3}+\frac{x_{34}x_3x_4}{x_2x_3}+\frac{x_{41}x_4x_1}{x_2x_3}+\frac{x_{42}x_4x_2}{x_2x_3}+\frac{x_{43}x_4x_3}{x_2x_3}\right)+\\
&\left(\frac{x_{12}x_1x_2}{x_2x_4}+\frac{x_{13}x_1x_3}{x_2x_4}+\frac{x_{14}x_1x_4}{x_2x_4}+\frac{x_{21}x_2x_1}{x_2x_4}+\frac{x_{23}x_2x_3}{x_2x_4}+\frac{x_{24}x_2x_4}{x_2x_4}+\right.\\
&\left.\frac{x_{31}x_3x_1}{x_2x_4}+\frac{x_{32}x_3x_2}{x_2x_4}+\frac{x_{34}x_3x_4}{x_2x_4}+\frac{x_{41}x_4x_1}{x_2x_4}+\frac{x_{42}x_4x_2}{x_2x_4}+\frac{x_{43}x_4x_3}{x_2x_4}\right)+\\
&\left(\frac{x_{12}x_1x_2}{x_3x_4}+\frac{x_{13}x_1x_3}{x_3x_4}+\frac{x_{14}x_1x_4}{x_3x_4}+\frac{x_{21}x_2x_1}{x_3x_4}+\frac{x_{23}x_2x_3}{x_3x_4}+\frac{x_{24}x_2x_4}{x_3x_4}+\right.\\
&\left.\frac{x_{31}x_3x_1}{x_3x_4}+\frac{x_{32}x_3x_2}{x_3x_4}+\frac{x_{34}x_3x_4}{x_3x_4}+\frac{x_{41}x_4x_1}{x_3x_4}+\frac{x_{42}x_4x_2}{x_3x_4}+\frac{x_{43}x_4x_3}{x_3x_4}\right)\geq M
\end{aligned}
$$

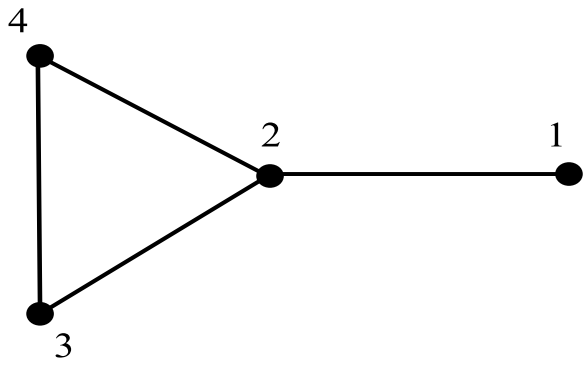
T5) $x_{ij}^{pq}=0 \vee a_{ij},\ (i=\overline{1,n},\ j=\overline{1,n},\ p=\overline{1,n-1},\ q=\overline{p+1,n})$

T6) $x_i=1 \vee M, \quad i=\overline{1,n}$

T7) $\sum_{i=1}^{n}x_i \to min$

4.3.2 Matematiksel Modelin Bir Graf için GAMS Programı Yardımıyla Çözümü

Bu bölümde, tepe birleştirilmişlik probleminin genel matematiksel modeli GAMS Programı yardımıyla aşağıda verilen 4 tepeli graf üzerinde çözdürülecektir. Çözüme ait detaylar Ek3'de verilmiştir.



Şekil 4.1: 4 tepeli G=(V,E) grafı

5. YOĞUNLUK KAVRAMI

İletişim ağlarındaki maksimum akış minimum kesime dayanan Menger Teoremi (Menger, 1927) literatüre birleştirilmişliğin en önemli teoremlerinden biri olarak geçmekle birlikte, bu teoremde grafın bağlantısız olmasını sağlayan tepelerin hangileri olduğu ile ilgili bir bilgi verilmemektedir. Bu bölümde tepe yoğunluğu ve kritik tepe tanımları yapılarak, bu kavramlar yardımıyla grafın bağlantılılığını bozan tepe ya da tepeler hakkında önceden bilgi sahibi olabilmek hedeflenmiştir. Bu nedenle, tepe bağlantılılık sayıları bilinen temel graf sınıfları için tepe yoğunluğu hesaplamaları yapılmıştır.

5.1 Tepe Yoğunluk Kavramının Tanımı

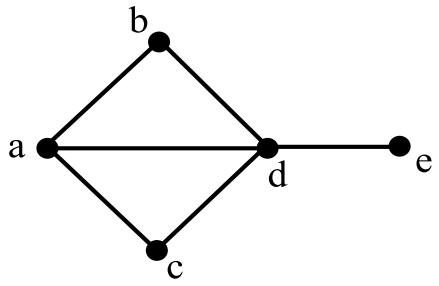
Tanım 5.1.1. Bir graftaki tüm tepe çiftleri arasındaki en kısa yolların üzerinde bulunan tepelerin kullanılma sıklığına **tepe yoğunluğu** denir.

Kesim tepeye denk gelen yeni bir kavram ise şu şekilde tanımlanır.

Tanım 5.1.2. Bir grafta tepe yoğunluğu en büyük olan tepe ya da tepelere **kritik tepe** denir.

Tanım 5.1.3. Bir graftaki kritik tepenin yoğunluğuna grafın **tepe yoğunluk sayısı** denir ve bu sayı $Y_t(G)$ şeklinde gösterilir.

Verilen bir grafta hangi tepenin (ya da tepelerin) yoğun olduğunu hesaplamak için bir Tepe Yoğunluk Tablosu hazırlanmalıdır. Bu tablonun satırlarından seçilen tepe, başlangıç tepesini, sütunlarından seçilen tepe ise geçilen tepeyi göstermektedir. Başlangıç tepesi ile grafın diğer tüm tepe çiftleri arasındaki en kısa yolların üzerinde bulunan tepeler geçilen tepe anlamına gelmektedir. Graflardaki en kısa yolu bulmak için kullanılan Floyd Algoritması yardımıyla tepe yoğunluğu da bulunabilir.



Şekil 5.1: 5 tepeli G_1 grafı

Örnek 5.1.4.

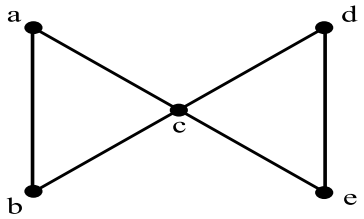
5 tepeli G_1 grafı için Tepe Yoğunluk Tablosu şu şekilde olacaktır:

Çizelge 5.1: 5 tepeli G_1 grafına ait Tepe Yoğunluk Tablosu

	a	b	c	d	e
a	-	0	0	1	0
b	1	-	0	2	0
c	1	0	-	2	0
d	0	0	0	-	0
e	0	0	0	3	-
$\sum$	2	0	0	8	0

Tepe Yoğunluk Tablosu kullanılarak kritik tepeyi bulmak için tablodaki herbir sütunda bulunan sayılar toplanarak, en büyük sayıya sahip sütun bulunur. Bu örnekte kritik tepe d tepesidir, yani grafın en yoğun tepesi d tepesidir. Bu durumda G_1 grafının tepe birleştirilmişlik sayısı $\kappa(G_1) = 1$ olarak bulunur.

Örnek 5.1.5.



Şekil 5.2: 5 tepeli G_2 grafı

Çizelge 5.2: 5 tepeli G_2 grafına ait Tepe Yoğunluk Tablosu

	a	b	c	d	e
a	-	0	2	0	0
b	0	-	2	0	0
c	0	0	-	0	0
d	0	0	2	-	0
e	0	0	2	0	-
$\sum$	0	0	8	0	0

Tepe Yoğunluk Tablosu kullanılarak G_2 grafına ait kritik tepe c tepesi olarak bulunur. Yani graftan c tepesinin silinmesi ile graf bağlantısız hale gelecektir, bu durumda G_2 grafının tepe birleştirilmişlik sayısı $\kappa(G_2) = 1$'dir.

5.2 Temel Grafların Tepe Yoğunluk Sayısı

Bu bölümde temel graf sınıflarının tepe yoğunlukları incelenmiştir. Tepe yoğunluk tablosu yardımıyla temel graf sınıflarından yol graf, yıldız graf, çevre graf ve tam grafın tepe yoğunluk sayılarını veren tablolar oluşturulmuştur.

- Aşağıdaki tabloda, P_n yol grafının tepe yoğunluk sayısı verilmiştir.

Çizelge 5.3: P_n yol grafının tepe yoğunluk sayısı

n	2	3	4	5	6	7	8	...
$Y_t(P_n)$	0	2	4	8	12	18	24	...

- Aşağıdaki tabloda, $K_{1,n-1}$ yıldız grafının tepe yoğunluk sayısı verilmiştir.

Çizelge 5.4: $K_{1,n-1}$ yıldız grafının tepe yoğunluk sayısı

n	4	5	6	7	...
$Y_t(K_{1,n-1})$	6	12	20	30	...

- Aşağıdaki tabloda, C_n çevre grafının tepe yoğunluk sayısı verilmiştir.

Çizelge 5.5: C_n çevre grafının tepe yoğunluk sayısı

n	3	4	5	6	7	8	...
$Y_t(P_n)$	0	2	2	6	6	12	...

- Aşağıdaki tabloda, K_n tam grafının tepe yoğunluk sayısı verilmiştir.

Çizelge 5.6: K_n tam grafının tepe yoğunluk sayısı

n	3	4	5	6	7	8	...
$Y_t(P_n)$	0	0	0	0	0	0	...

5.2.1 Sonuçlar

Sonuç 5.2.1. *n tepeli bir P_n yol grafı için, $n \geq 2$ olmak üzere,*

$$Y_t(P_n) = \begin{cases} \dfrac{n(n-2)}{2}, & n \textit{ çift ise;} \\ \dfrac{(n-1)^2}{2}, & n \textit{ tek ise} \end{cases}$$

elde edilir.

Sonuç 5.2.2. *n tepeli bir $K_{1,n-1}$ yıldız grafı için, $n \geq 4$ olmak üzere,*

$$Y_t(K_{1,n-1}) = (n-1)(n-2)$$

elde edilir.

Sonuç 5.2.3. *n tepeli bir C_n çevre grafı için, $n \geq 3$ olmak üzere,*

$$Y_t(C_n) = \begin{cases} \dfrac{n(n-2)}{4}, & n \textit{ çift ise;} \\ \dfrac{(n-1)(n-3)}{4}, & n \textit{ tek ise} \end{cases}$$

elde edilir.

Sonuç 5.2.4. *n tepeli bir K_n tam grafı için,*

$$Y_t(K_n) = 0$$

elde edilir.

6. GENETİK ALGORİTMA ve TEPE BİRLEŞTİRİLMİŞLİK

Son yirmi yıl içinde, değişik mühendislik konularının bilgisayar aracılığı ile modellenmesi, benzetilmesi (simülasyon), en iyilenmesi (optimizasyon) ve gelecek davranışlarının tahmini için, doğal olayların işleyiş ve davranış biçimlerinden esinlenerek ilgi çekici yöntemler geliştirilmiştir. Bunlar arasında canlıların genetik davranış biçimleri genetik algoritmaların (GA) ortaya çıkmasında çok önemli rol oynamıştır. Genetik algoritmalar, doğal seçim ilkelerine dayanan bir arama ve optimizasyon yöntemidir. Temel ilkeleri J. H. Holland (Holland, 1975) tarafından ortaya atılmıştır. Olasılık kurallarına göre çalışan genetik algoritmalar, yalnızca amaç fonksiyonuna gereksinim duyar ve çözüm uzayının tamamını değil, belirli bir kısmını tararlar. Böylece, etkin arama yaparak çok daha kısa sürede çözüme ulaşırlar. Diğer bir önemli üstünlükleri ise çözümlerden oluşan popülasyonu eş zamanlı incelemeleri ve böylelikle yerel en iyi çözümlere takılmamalarıdır.

Genel olarak, En İyileme (Eİ) (en büyükleme ve en küçükleme) sorunlarının çözümü için uygun olan GA kullanımı, diğer yöntemlerle karşılaştırıldığında çok daha iyi çözümlere en kısa zamanda sayısal örgünlük ve rastgele düzen içinde gidebilmektedir. GA, bir Eİ sorununa en uygun çözümü en kısa zamanda bulabilmek için evrim teorisinden esinlenerek ortaya konulmuş bir yöntemdir. Sorun genetik sayı dizileri ile kodlanmış karar değişkenleri ile çözüm seçenekleri arasında evrim işlemlerini kullanarak en uygun çözüme yaklaşmaya çalışılmasıdır. GA'ların temelinde rastgele örnekleme bulunur ve bu nedenle GA'lar belirsiz yöntemlerdir. GA'larda belirsizlik yöntemleri kullanılır ve algoritmanın işleyişinde belirginlik yoktur. Bunun doğal sonucu olarak aynı sorun için aynı GA modeli değişik defalar kullanılınca birbirinden biraz farklı olan sonuçlar verebilir. Biyolojik evrim teorisinden esinlenerek ortaya konulmuş olan GA yöntemleri, çözüm alanını stotastik yani rastgele biçimde bombardımana tutarak en iyi çözümü arayan bir yöntemdir. Çözüme ulaşabilmek için önce karar değişkeni uzayında rastgele olarak noktalar topluluğu alınır, daha sonra gösterilecek kuralların ışığı altında bu noktalar arasında eşleştirmeler yapılarak toplumun bazı üyeleri yok olurken onların yerine yenileri gelir. Yeni gelenlerin topluma katılması ile o toplumun öncekinden daha sağlıklı yani hedefe daha yakın olması sağlanır. Bu toplumun üyeleri arasında gerekli genetik işlemlerin uygulan-

ması ile hedefe daha yakın yeni bir toplum elde edilir. Böylece hedefe bir yön veya yol boyunca değil, birçok yönlerden ve kısa yollardan yaklaşılır. Bu toplumdaki üyelerin herbiri kodlanmış birer genetik sayı dizisi ile temsil edilir. Bu sayı dizilerine kromozom adı verilir. Karar değişkenlerinin kromozomlar vasıtasıyla kodlanmasından sonra toplumda bulunan her kromozomun dinçliğinin ayrı ayrı hesaplanması gerekir. Üyelerin dinçliklerinin hesaplanmasına yarayacak bir hedef fonksiyonu bulunmalıdır. Hedef fonksiyonu kriterine göre topluma üye olan kromozomlar ya hayatlarını devam ettirirler veya o toplumu terk ederler. Hedef fonksiyonunun en önemli özelliği eşleşerek toplumda daha dinç üyelerin meydana gelmesine sebep olacak üyelerin seçilmesine yardımcı olmasıdır. İncelenecek olayla ilgili ne kadar bulunabilirse o kadar fazla sözel ve sayısal bilgiler toplanarak, bunların herşeyden önce akıl, mantık ve basit klasik yöntemlerle aralarında nasıl ilişki ve ne tür mantık kurallarının bulunduğunu belirlemeye çalışmalıdır. GA'ların çalışma prensibi kısaca şöyle özetlenebilir:

Adım 1. Olası çözümlerin kodlandığı bir çözüm kümesi oluşturulur, bu çözüm kümesi popülasyon olarak, çözümlerin kodları da kromozom olarak adlandırılır. Problemin türüne göre değişik kodlama şekilleri mevcuttur. İkilik düzene kodlama, permütasyon kodlaması, reel sayı kodlaması v.b.

Adım 2. Popülasyondaki her bir kromozomun uygunluk fonksiyonuna göre ne kadar iyi olduğu bulunur. Uygunluk fonksiyonuna göre iyi çözüm sonuçları veren kromozomlar, yeni popülasyona alınmak üzere seçilirler. Problemin türüne göre geliştirilmiş bir çok seçim mekanizması mevcuttur.

Adım 3. Seçilen kromozomlar eşlenerek, çaprazlama ve mutasyon operatörleri uygulanır. Bu sayede yeni bir popülasyon oluşturulur.

Adım 4. Tüm kromozomların uygunlukları tekrar hesaplanır.

Adım 5. Eğer durdurma kriteri sağlanmamışsa Adım 2'ye gidilir. Durdurma kriteri istenen bir nesil sayısı ya da popülasyondaki durağanlığın gerçekleşmesi olabilir.

Adım 6. O ana kadar bulunmuş en iyi kromozom optimum sonuç olarak değerlendirilir.

GA ile bir sorunu çözmeye çalışırken birbirini takip eden aşağıdaki beş aşamanın tamamlanması gereklidir.

- Sorunun ne olduğuna ait etraflı ve ayrıntılı ön bilgilere, sözel ve özellikle de sayısal verilere mümkün olduğu kadar ulaşmaya çalışmalı.
- GA ile çözümlenmesi beklenen sorun ile ilgili olabilecek değişkenlerin kromozom yapılarına karar vermek ve veri hazırlığında bulunmak.
- Karar değişkenlerinden kromozomlara geçmek için gerekli olabilecek tüm alt yapıyı hazırlamak.
- Sorunla ilgili hedef fonksiyonunun karar değişkenlerine bağlı olarak analitik ifadesini tespit etmek.
- Hedef fonksiyonundan her bir karar takımına (kromozom) karşı gelecek derecelerin hesaplamasına yarayacak dönüşümü belirlemek.

6.1 Genetik Algoritmaların Tanımı

GA yöntemi, evrim teorisi esaslarına göre çalışarak verilen bir sorun için en iyi çözüm veya çözümleri arayarak bulmaya yarar. Bu arayışı, karar değişkeni uzayındaki birçok başlangıç noktasından başlayarak, paralel işlemler dizisi ile en iyi yöne doğru topluca gelişerek yapar. GA'ların esası doğal seçme ve genetik kurallarına dayanmaktadır. Bu kurallar ortama en fazla uyum sağlayan canlıların hayata devam etmesi ve uyum sağlayamayanların da elenmesi olarak algılanmalıdır. GA'lar bu iki kuralı bir arada kullanarak en iyiyi aramayı hedef edinen bir En İyileme (Eİ) yöntemidir. GA'lar basit hesaplamalar gerektirir ama basitliğinden dolayı etkinliği azalmaz. Diğer pek çok Eİ yöntemlerindeki süreklilik ve türev alınabilme şartlarını gerektirmez. Uygulamalarında ağır matematik bilgileri kulanılmaz. GA'lar rastgele arama yöntemlerinden farklı olarak, ihtimal ilkelerini karar değişkeni uzayında genetik işlemler yapmada araç olarak kullanır. GA'lar ile diğer geleneksel yöntemler arasındaki farklar şunlardır :

1) GA, çözümlemesinde karar değişkenlerini genetik sayı sistemine göre kodlayarak (ikili sayı sistemine dönüştürerek) kullanır. Bu sayı sisteminde karar değişkenlerinin genleri topluca karar uzayında bir noktayı temsil eder.

2) GA'da bir nokta yerine aynı anda noktalar topluluğundan hareket edilir. Bu topluluğun GA evrimi ile gelişmesi sonucunda Eİ çözüme ulaşılır. Bu evrim sırasında sistem yerel en iyiye takılmaz.
3) GA evrimi sırasında, karar değişkenlerinin belirttiği noktalardaki hedef fonksiyonu değerleri kullanılır. Türev ve integral işlemlerine gerek olmadığından başlangıç ve sınır şartları ile bazı klasik kabullerin yapılmasına gerek yoktur.
4) GA evrim işlemleri belirlilik değil belirsizlik (rastgelelik, ihtimal) kurallarına dayanır.

6.2 Basit Bir Genetik Algoritma Yapısı

GA, araştırma uzayında bulunan çözümlerin bazılarının oluşturduğu bir başlangıç popülasyonunu kullanmaktadır. Başlangıç popülasyonu her jenerasyonda, tabii seçme ve tekrar üreme işlemleri ile ard arda geliştirilir. En son kuşağın en uygun yani en kaliteli bireyi, problem için optimal çözüm olmaktadır. Bu çözüm her zaman optimum olmayabilir ama kesinlikle optimuma yakın bir optimal çözümdür. Holland (1975) basit bit dizileri kullanarak karmaşık yapıların kodlanabileceğini göstermiştir. Yapılar, çözülecek problem için çözümleri temsil etmektedir. Bunlar muhtemel tüm çözümleri içine alan araştırma uzayından alınır ve bu dizilerin veya çözümlerin belirli bir miktarı genetik algoritmanın kullanılacağı popülasyonu oluşturur. Daha sonra temel genetik operatörlerin belirli bir seti, ard arda gelen jenerasyonlarda çözümleri geliştirmek için kullanılır. Şayet bu işlem uygun şekilde kontrol edilirse, çözüm popülasyonunun ortalama kalitesi çok hızlı olarak gelişme gösterir. Yani, çözülecek probleme çok iyi uyarlanmış yapıları içeren çözüm bulununcaya kadar devam etmektedir. Basit bir genetik algoritmanın temel adımları şu şekildedir:
Adım 1: Muhtemel çözümlerden bir başlangıç popülasyonunu oluştur.
Adım 2: Popülasyondaki her çözümün uygunluk değerini hesapla.
Adım 3: Durdurma kriteri sağlanıyorsa araştırmayı durdur. Aksi halde, aşağıdaki adımları gerçekleştir.
3.1 Tabii seleksiyon işlemini uygula (uygunluk değerleri daha yüksek olan çözümler yeni popülasyonda daha fazla temsilciye, daha düşük olanlar ise daha az temsilciye sahip olacaktır).
3.2 Önceki popülasyonda var olan en iyi çözümü muhafaza et.
3.3 Çaprazlama işlemini uygula (mevcut iki çözümden yeni iki yapı üretilir).

3.4 Mutasyon işlemini uygula (çözümlerde rastgele değişim meydana getirilir).
Adım 4: Adım 2 ye git.

6.2.1 Kodlama

Genetik algoritmaların en büyük özelliği parametre değerlerinin kodlanmasıdır. Problemin türüne göre değişik kodlama şekilleri mevcuttur (Gen and Cheng, 1996). Bunlar ikili düzende kodlama, permütasyon (sıralı) kodlama ve değer kodlamasıdır. İkili Düzende Kodlama en yaygın olarak kullanılan kodlama türüdür. Bu kodlama türünde kromozomlar 0 ve 1 şeklinde kodlanırlar.

Çizelge 6.1: İkili düzende kodlama örneği

Kromozom A	1100101111000111110001010001
Kromozom B	0010001111001110101010101111

Permütasyon (sıralı) kodlama sıra gözeten problemlerde kullanılır.

Çizelge 6.2: Permütasyon kodlama örneği

Kromozom A	1 5 9 2 3 6 4 7 8
Kromozom B	4 2 5 6 9 7 1 8 3

Değer Kodlamasında direkt olarak parametre değerleri alınır. Bu kodlama özel problemler için oldukça yararlıdır.

Çizelge 6.3: Değer kodlaması örneği

Kromozom A	1.2342 5.4234 2.6876 3.7285
Kromozom B	AKSOJKRTPWBNDIHGDF

6.2.2 Başlangıç Popülasyonunun Oluşturulması

Başlangıç popülasyonu genellikle rastgele sayı üreticisi kullanılarak oluşturulur. Şayet problemle ilgili başlangıçta bazı çözümler kabaca biliniyorsa başlangıç popülasyonu bu çözümler kullanılarak oluşturulabilir.

6.2.3 Uygunluk Fonksiyonu ve Seçme

Popülasyonun kalitesini belirlemede kullanılır. Kromozomların ne kadar iyi olduğunu bulan fonksiyona uygunluk fonksiyonu denir. Bu fonksiyon işletilerek kromozomların uygunluklarının bulunmasına ise değerlendirme adı verilir. Uygunluk fonksiyonu kromozomların şifresini çözer ve sonra hesaplama yaparak bu kromozomların uygunluğunu bulur. Uygunluğu yüksek olan kromozomların seçilmesi için geliştirilmiş değişik yöntemler vardır. Bunlar şu şekilde sınıflandırılabilir (Goldberg, 1989).

1. Orantılı yeniden üretim yöntemleri (Rulet tekerleği yöntemi)
2. Sıralı seçim yöntemleri
3. Turnuva seçim yöntemleri

6.2.4 Genetik Operatörler

Tekrar Üreme

Tekrar üreme operatörü, tabii seçme işlemi olarak adlandırılan kalitesi yüksek bireylerin hayatta kalmaları ve sayılarının artması, kalitesi düşük bireylerin ise sayılarının azalarak kaybolması prensibine göre çalışan bir genetik algoritma elemanıdır. Bu operatörün uygulanması için kullanılan en basit metod rulet tekerleği tekniğidir.

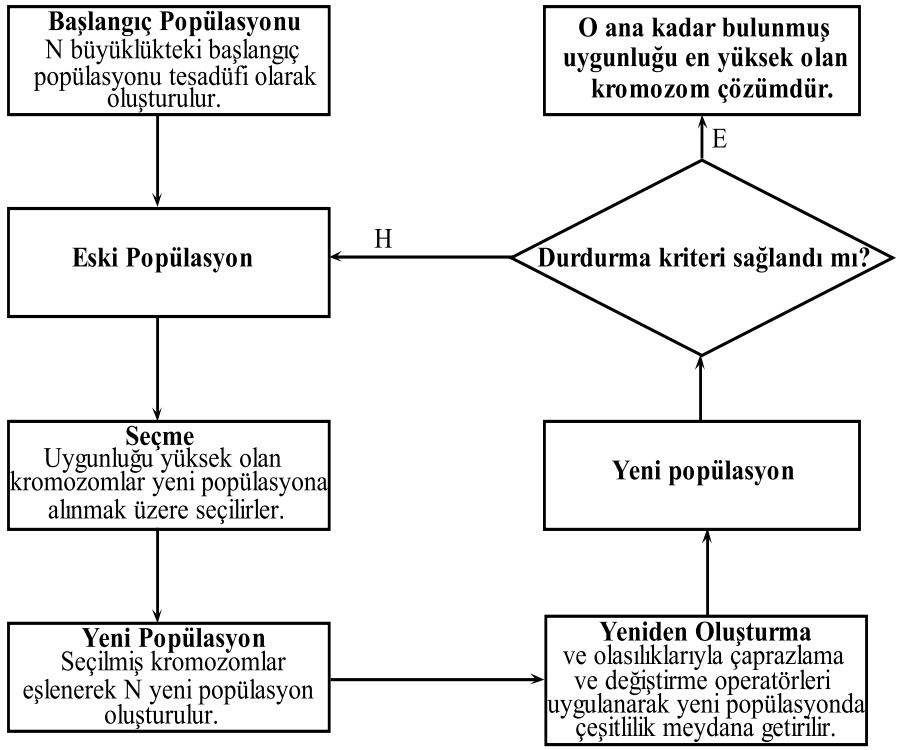
Çaprazlama

Çaprazlama ise genetik algoritmanın çok önemli araçlarından birisidir. Çaprazlama popülasyonda çeşitliliği arttırarak en iyiye yaklaşmayı sağlar. Eşleştirme havuzunda bulunan yapıların birer çifti rastgele seçilir ve çaprazlama operatörü bu iki yapıdan yeni iki yapı meydana getirmek için kullanılır. Eski yapılar (ebeveyn) ve çaprazlamadan sonra ortaya çıkan yeni iki yapı (çocuklar) mevcut jenerasyonda tutulur veya eski ile yeni yapılar yer değiştirirler. İkinci durumda kötü yapılar atılır ve popülasyon büyüklüğü sabit olarak korunur. İkili kodlama düzeninde en çok kullanılan çaprazlama yöntemleri tek nokta çaprazlama, iki nokta çaprazlama ve tek düze çaprazlamadır.

Tek nokta çaprazlamada, kromozom uzunluğu l olarak alır. $[0, l-1]$arasında rastgele bir sayı seçilir ve seçilen noktada kromozom parçaları yer değiştirir.

6.3 Genetik Algoritma ile Tepe Birleştirilmişlik Hesabı

Çalışmanın bu bölümünde birleştirilmiş bir grafın tepe birleştirilmişlik sayısını ve silinen tepeleri veren bir algoritma önerilmiştir. Bu algoritmayı daha iyi kavrayabilmek için, genetik algoritmaların temel yapısı ve işleyişi hakkında bir akış diyagramı verilmiştir.



Şekil 6.1: Genetik algoritmaların yapısı ve işleyişi ile ilgili akış diyagramı

Başlangıç popülasyonun n adet elemanı tüm tepelerin birer kez kullanıldığı bireyler ile üretilmiş olup geri kalanı rastgele oluşturulmuştur. Uygunluk fonksiyonuna tepe sayısının katkısı birleştirilmişlik tanımı gereği ters orantılıdır yani seçilen tepe sayısı paydada yer alır. Ayrıca bireyin problemi çözme kabiliyeti var ise (yani çizgeyi bağlantısız yapıyorsa) önceki adımda elde edilen kesre 1 sayısı ilave edilir aksi halde kesir aynı değerinde kalır. Sonuç olarak uygunluk fonksiyonun son hali;

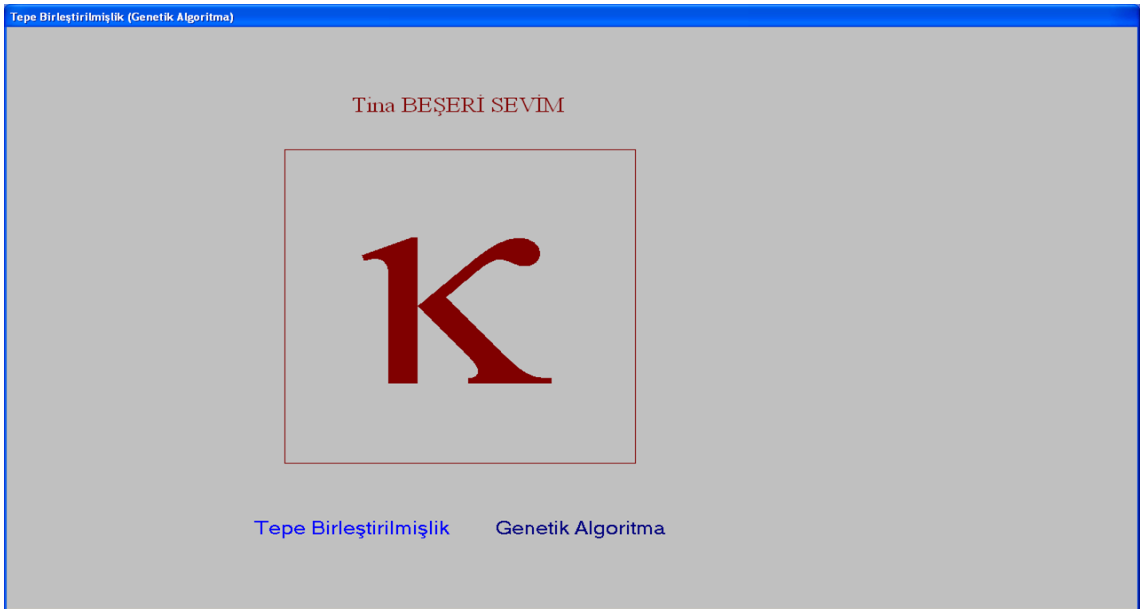
$$f = \frac{1}{|v_i|} + (0, 1)$$

'dir.

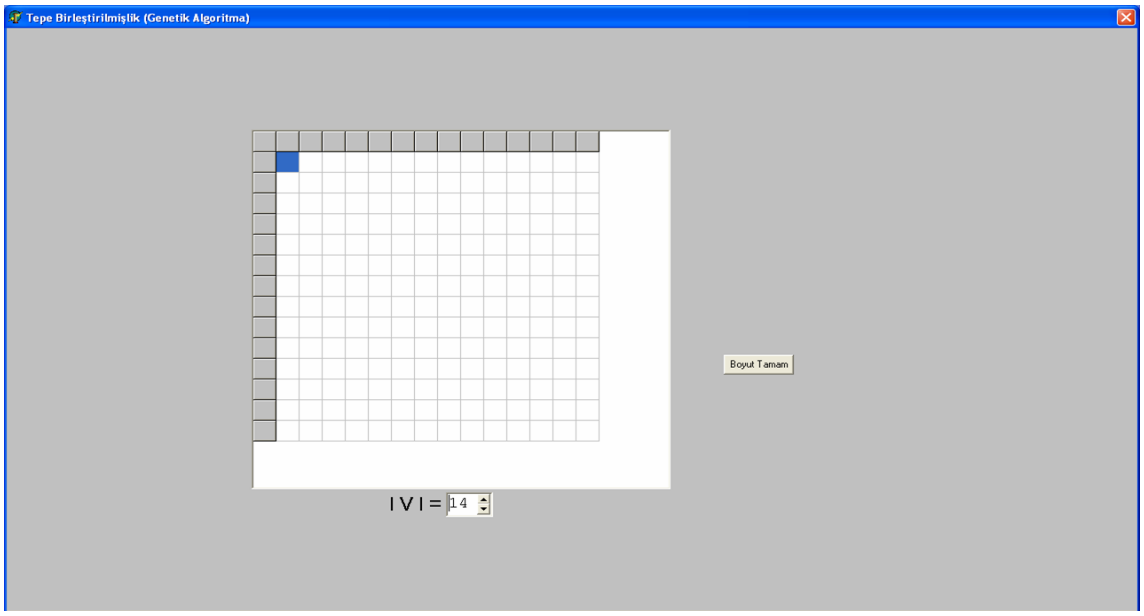
Uygunluk fonksiyonundan elde edilen sonuçlara göre bireyler uygunluk değerleri ile doğru orantılı olarak rulet tekerleği üzerine yerleştirilir. Rastgele seçilen her

çiftten tek nokta çaprazlama tekniği kullanılarak bir oğul oluşturulur. Bu işlem birey sayısının yarısı kadar yapılır. Bireylerin diğer yarısı da elitizm tekniği ile yani uygunluk fonksiyonu sonuçları büyükten küçüğe sıralanarak en iyi bireylerin alınmasıyla elde edilir. Nesil(jenerasyon) sayısı kadar bu işlemler yapılır ve algoritma sonlanır. Sonuçta tüm aşamalarda popülasyonun en iyi bireyi saklanır ve gerektiğinde güncellenerek çözüm bulunmuş olur.

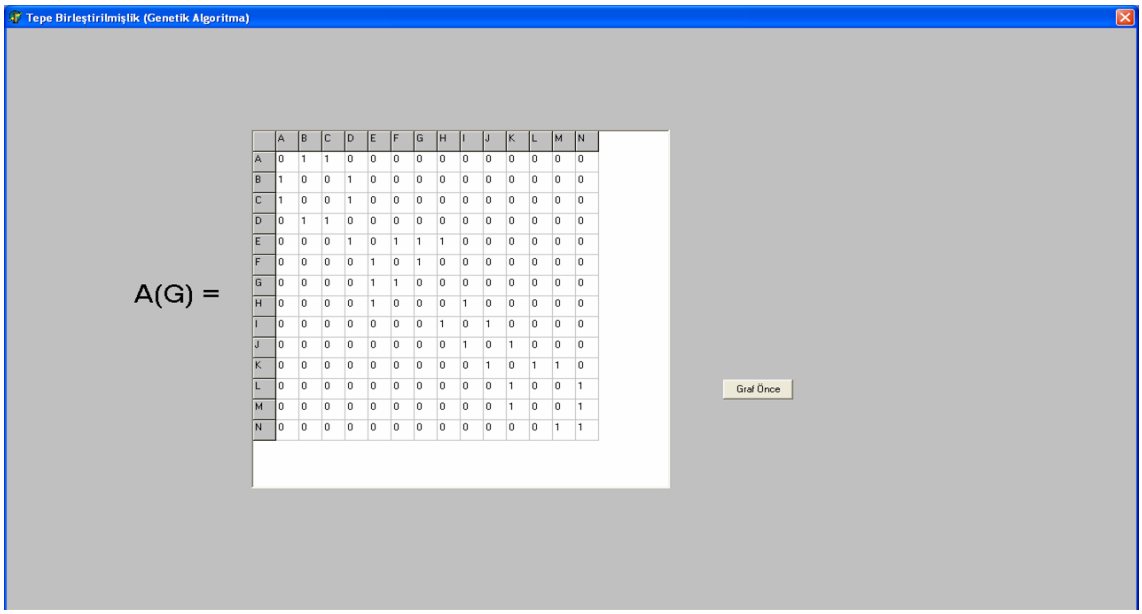
Bir sonraki sayfada, bu algoritmayı kullanarak tepe birleştirilmişlik sayısını hesaplayan gatbs yazılımının ekran görüntüleri verilecektir.



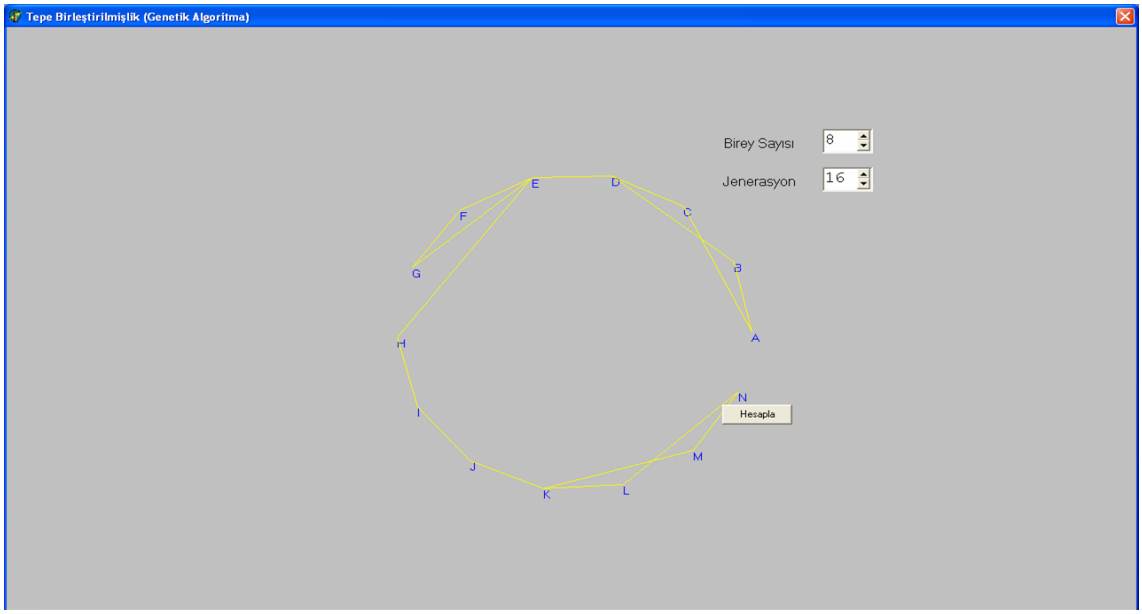
Şekil 6.2: **gatbs** Programına Giriş Ekranı



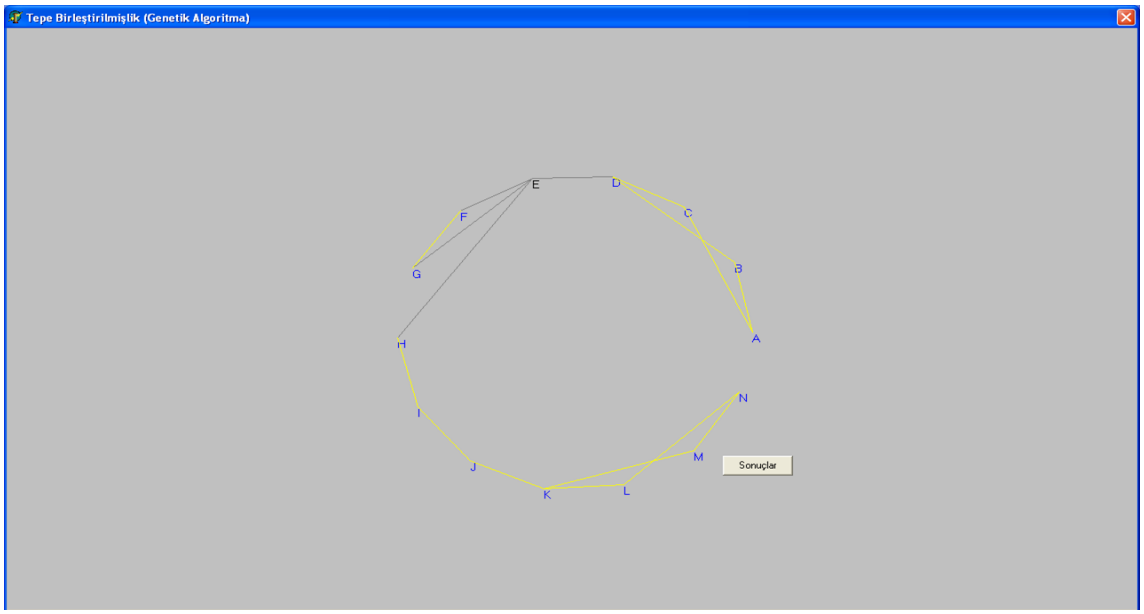
Şekil 6.3: Graf Boyutunun Ayarlandığı Ekran



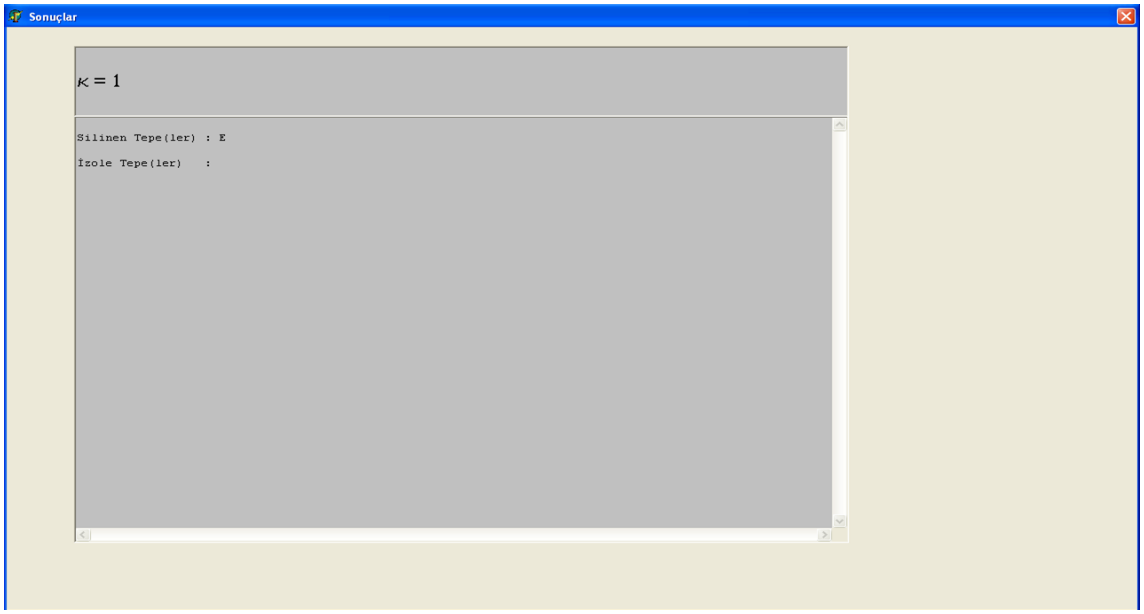
Şekil 6.4: Grafın Bitişiklik Matrisinin Girildiği Ekran



Şekil 6.5: Grafın Ekran Görüntüsü, Birey ve Jenerasyon Sayısının Seçimi



Şekil 6.6: Grafın Silinecek Tepe ve Ayrıtlarının Gösterimi



Şekil 6.7: Sonuçların Alındığı Ekran

7. SONUÇ

Bu tezde, grafların zedelenebilirlik ölçümlerinden biri olan tepe birleştirilmişlik kavramı ayrıntılı olarak incelenmiş, konuyla ilgili temel tanım ve teoremler verilmiştir. Literatürdeki tepe birleştirilmişlik sayısını bulan algoritmalar incelenip, verimlilikleri karşılaştırılmıştır. Geliştirilen yeni algoritma ile tepe birleştirilmişlik sayısının yanı sıra grafın bağlantısız olması için graftan silinen tepeler de bulunabilmektedir. Bu algoritmanın C++ dilindeki programı Ek 2'de sunulmuştur. Bunun yanı sıra, tepe birleştirilmişlik probleminin matematiksel modelleri verilip, genel matematiksel modelin Ek 3'de GAMS arayüzü ile bir örnek üzerinde çözümü yapılmıştır. Tepe birleştirilmişlik kavramına bağlı olarak graftaki en yoğun tepenin bulunmasını hedefleyen tepe yoğunluk sayısı tanımlanarak temel graf sınıfları için tepe yoğunluk değerleri hesaplanmıştır.

Son olarak, genetik algoritma tekniği ile ilgili kısa bir bilgilendirmeden sonra bu teknik yardımıyla tepe birleştirilmişlik sayısını hesaplayan ve Ek 1'de yer alan DELPHİ dilinde bir bilgisayar programı sunulmuştur. Bu program üzerinde denemeler yapılıp ekran görüntüleri verilmiştir.

KAYNAKLAR DİZİNİ

Agnarsson, G. and Greenlaw, R., 2007, *Graph Theory: Modeling, Applications, and Algorithms*, Prentice Hall, N.J.,464p.

Aldous, J.M. and Wilson, R.J., 2000, *Graphs and Applications: An Introductory Approach*, Springer, London, 444p.

Altunkaynak, B. ve Bakır, M. A., 2003, *Tam Sayılı Programlama Teori, Modeller ve Algoritmalar*, Nobel Yayıncılık, Ankara, 630s.

Balakrishnan, R. and Ranganathan, K., 2000, *A Textbook of Graph Theory*, Springer, New York, 227p.

Becker, M., Degenhardt, W., Doenhardt, J., Hertel, S., Kaninke G. and Keber W., 1982, A probabilistic algorithm for vertex connectivity of graphs. *Information Processing Letters*, 15, 135-136pp.

Beineke, L.W. and Harary, F., 1967, The connectivity function of a graph, *Mathematica*, 14, 197-202pp.

Biggs, N.L., Lloyd E.K. and Wilson, R.J., 1999, *Graph Theory 1736-1936*, Oxford University Press, New York, 240p.

Bollobás, B., 1994, *Graph Theory: An Introductory Course*, Springer, New York, 180p.

Bollobás, B., 1998, *Modern Graph Theory*, Springer, New York, 408p.

Bollobás, B., 2004, *Extremal Graph Theory*, Dover Publications, 512p.

Bondy, J.A. and Murty, U.S.R., 1976, *Graph Theory with Applications*, London, Macmillan Press Ltd., 263p.

Bondy, J. A.and Murty, U. S. R., 2008, *Graph Theory*, Springer, New York, 654p.

Buckley, F. and Harary, F., 1990, *Distance in Graphs*, Addison-Wesley Publishing Company, 335p.

KAYNAKLAR DİZİNİ (devam)

Buckley, F. and Lewinter, M., 2003, *A Friendly Introduction to Graph Theory*, Prentice Hall, N.J., 384p.

Chartrand, G., 1966, A graph-theoretic approach to a comunications problem, *SIAM Journal on Applied Mathematics*, 14, 778-781pp.

Chartrand, G. and Lesniak, L., 1996, *Graphs and Digraphs*, Chapman and Hall, California, 421p.

Çölkesen, R., 2006, *Veri Yapıları ve Algoritmalar*, Papatya Yayıncılık, İstanbul, 424s.

Diestel, R., 2006, *Graph Theory*, Springer, New York, 415p

Dirac, G.A., 1960, In abstrakten Graphen vorhandene vollständige 4-Graphen und ihre Unterteilungen, *Math. Nachr.*, 22, 61-85pp.

Esfahanian, A.H. and Hakimi, S.L., 1984, On computing the connectivities of graphs and digraphs, *Networks*, 355-366pp.

Even, S., 1975, An algorithm for determining whether the connectivity of a graph is at least *k*, *SIAM Journal of Computing*, 4, 393-396pp.

Even, S. and Tarjan, R. E., 1975, Network flow and testing graph connectivity *SIAM Journal of Computing*, 4, 507-518pp.

Even, S., 1979, Graph algorithms, *Computer Science Press*, 17-19pp.

Ford, L.R. and Fulkerson, D.R., 1956, Maximal flow through a network, *Canad. J. Math.*, 8, 399-404pp.

Galil, Z., 1980, Finding the vertex connectivity of graphs, *SIAM J. Computing*, 9, 197-199pp.

Gen, M. and Cheng, R., 1996, *Genetic Algorithms and Engineering Design*, Wiley-Interscience, 432p.

Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimization and Machine*, Addison-Wesley, 432p.

KAYNAKLAR DİZİNİ (devam)

Gross, J. L. and Yellen, J., 2004, *Handbook of Graph Theory*, CRC Press, London, 1167p.

Gross, J. and Yellen, J., 2006, *Graph Theory and Its Applications*, Chapman and Hall, CRC Press, Florida, 800p.

Harary, F., 1962, The maximum connectivity of a graph, *Proceedings of the National Academy of Science of the USA*, 48, 1142-1146pp.

Harary, F., 1972, *Graph Theory*, Addison-Wesley Publishing Company, 274p.

Hartuv, E. and Shamir, R., 2000, A clustering algorithm based on graph connectivity, *Information Processing Letters*, 76, 175-181pp.

Hellwig, A. and Volkmann, L., 2008, The connectivity of a graph and its complement, *Discrete Applied Mathematics*, 156, 3325-3328pp.

Henzinger, M.R., Rao, S. and Gabow, H.N., 1996, Computing vertex connectivity: new bounds from old techniques, *Proc. 37th IEEE F.O.C.S*, 462-471pp.

Holland, J. H., 1975, Adaptation in Natural and Artificial Systems, *University of Michigan Press*, 15, 135-136pp.

Kammer, F. and Taubig, H., 2004, Connectivity, *Network A Analysis*, 143-177pp.

Kara, İ., 2000, *Doğrusal Programlama*, Bilim Teknik Yayınevi, Ankara, 270s.

Kocay, W. and Kreher, D. L., 2005, *Graphs, Algorithms and Optimization*, Chapman and Hall/CRC, London, 504p.

Mader, W., 1972, Ecken vom Grad n in minimalen n-fach zusammenhängenden Graphen, *Archive der Mathematik*, 23, 219-224pp.

Mader, W., 1979, Connectivity and edge-connectivity in finite graphs, *Surveys in Combinatorics (Proc. Seventh British Combinatorial Conf.)*, Cambridge Univ. Press, 66-95pp.

Melnikov, O., Sarvanov, V., Tyshkevich, R., Yemelichev, V. and Zverovich, I., 1998, *Exercises in Graph Theory*, Kluwer Academic Publishers, Netherlands, 344p.

KAYNAKLAR DİZİNİ (devam)

Menger, K., 1982, Zur allgemeinen Kurventheorie, *Fund. Math.*, 10, 96-115pp.

Nabiyev, V.V., 2007, *Algoritmalar Teoriden Uygulamalara*, Seçkin Yayıncılık, Ankara, 799s.

Nagamochi, H. and Ibaraki, T., 2002, Graph connectivity and its augmentation: applications of MA orderings, *Discrete Applied Mathematics*, 123, 447-472pp.

Ore, O., 1996, *Graphs and Their Uses*, The Mathematical Association of America, Washington, 160p.

Read, R. C. and Wilson, R. J., 1998, *An Atlas of Graphs*, Clarendon Press, New York, 464p.

Tarjan, R.E., 1972, Depth first search and linear graph algorithms, *SIAM J. Computing.*, 1, 146-160pp.

Şen, Z., 2004, *Genetik Algoritmalar ve En İyileme Yöntemleri*, Su Vakfı Yayınları, İstanbul, 142s.

Wallis, W.D., 2007, *A Beginner's Guide to Graph Theory*, Birkhauser, Boston, 252p.

West, D. B., 2001, *Introduction to Graph Theory*, Prentice-Hall, London, 470p.

Whitney, H., 1932, Congruent graphs and the connectivity of graphs, *Amer. J. Math.*, 54, 150-168pp.

Wilson, R. J., 1996, *Introduction to Graph Theory*, Addison Wesley, London, 184p.

Xu, J., 2003, *Theory and Application of Graphs*, Kluwer Academic Publishers, Boston, 342p.

ÖZGEÇMİŞ

4 Mayıs 1979 tarihinde Tahran'da doğan Tina Beşeri Sevim, liseden mezun olduğu 1996 yılında Dokuz Eylül Üniversitesi, Buca Eğitim Fakültesi, Matematik Öğretmenliği Bölümü'nü kazandı. Dokuz Eylül Üniversitesi'nden 2000 yılında mezun olarak aynı yıl İzmir Yüksek Teknoloji Enstitüsü, Matematik Bölümü'nde araştırma görevlisi olarak çalışmaya başladı. 2001 yılı Eylül ayında İYTE Matematik Bölümü'nde Yüksek Lisans eğitimine başladı. 2004 yılında Matematik Bölümünden Yüksek Lisans derecesi ile mezun olarak 2004 yılı Eylül ayında Ege Üniversitesi, Matematik Bölümü Bilgisayar Bilimleri Ana Bilimdalı'nda Doktora eğitimine başladı. Halen İzmir Yüksek Teknoloji Enstitüsü, Matematik Bölümü'nde araştırma görevlisi olarak çalışmaktadır.

Yayınları

1. Beşeri Sevim T., Berberler M.E., "Asal Sayıların Bulunması İçin Bir Elek Önerisi", XXI. Ulusal Matematik Sempozyumu Bildiri Kitapçığı, Eylül 2008, Koç Üniversitesi, İstanbul.

2. Nuriyev U., Berberler M.E., Beşeri Sevim T., "Çizge Teorisinin Öğretiminde Bilgisayar Teknolojisinin Kullanımı", I. Uluslararası Bilgisayar ve Öğretim Teknolojileri Sempozyumu Bildiri Kitapçığı, Mayıs 2007, Çanakkale Onsekiz Mart Üniversitesi, Çanakkale.

3. Ufuktepe Ü., Bacak G., Beseri T., "Graph Coloring with webMathematica", Springer-Verlag Lecture Notes in Computer Science, 2004, Volume 3039, 376-381pp.

4. Ufuktepe Ü., Beseri T., Bacak G., "Graph Theory with webMathematica", Proceedings of the 6th International Mathematica Symphosium, Canada, 2004.

5. Bacak G., Beşeri T., "Çizge Kuramına Genel Bir Bakış." Matematik Dunyasi Dergisi, Eylül 2002.

Ek 1. Genetik Algoritma ile Tepe Birleştirilmişliği Delphi Diliyle Hesaplayan Programın Kaynak Kodu (gatbs)

```
 unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, jpeg, ExtCtrls, StdCtrls, Spin, Grids;

const
 MAX=32767;
 maxn=16;
 maxp=832;

type
  krom = array[1..maxn] of byte;
  TForm1 = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    StringGrid1: TStringGrid;
    SpinEdit1: TSpinEdit;
    SpinEdit2: TSpinEdit;
    SpinEdit3: TSpinEdit;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    procedure moveto(x,y : integer);
    procedure lineto(x,y : integer);
    procedure line(x1,y1,x2,y2 : integer);
    procedure circle(x,y,r : integer);
    procedure graphdraw;
    procedure FormActivate(Sender: TObject);
    procedure Image1Click(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure SpinEdit2Change(Sender: TObject);
```

```
    procedure SpinEdit3Change(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1         : TForm1;
  n             : word;
  V             : array[1..maxn] of word;
  IsoV          : array[1..maxn] of word;
  Graph         : array[1..maxn,1..maxn] of longint;
  Graph2        : array[1..maxn,1..maxn] of longint;
  Gxy           : array[1..maxn,1..2] of longint;
  Distance      : array[1..maxn,1..maxn] of longint;
  pop,pop2      : array[1..maxp] of krom;
  result        : krom;
  p             : word;  {8*maxn}
  maxgeneration : word;  {16}
  os            : char;

implementation

uses unit2;

{$R *.DFM}

{ Orijin X : x+round(form1.width/2)
Orijin Y : -y+round(form1.height/2) }

procedure TForm1.moveto(x,y : integer);
begin
 canvas.moveto(x+round(form1.width/2),-y+round(form1.height/2));
end;

procedure TForm1.lineto(x,y : integer);
begin
 canvas.lineto(x+round(form1.width/2),-y+round(form1.height/2));
end;

procedure TForm1.line(x1,y1,x2,y2 : integer);
begin
 canvas.moveto(x1+round(form1.width/2),-y1+round(form1.height/2));
 canvas.lineto(x2+round(form1.width/2),-y2+round(form1.height/2));
end;

procedure TForm1.circle(x,y,r : integer);
begin
 if os='o' then canvas.font.color:=clBlue
 else
 begin
  if V[r]=1 then canvas.font.color:=clBlue
   else if V[r]=0 then canvas.font.color:=clBlack;
  if IsoV[r]=1 then canvas.font.color:=clRed;
 end;
 canvas.font.size:=10;
 canvas.textout(x+round(form1.width/2),-y+round(form1.height/2)
```

```
                ,chr(64+r));
end;

procedure TForm1.graphdraw;
var
 i,j,r,delta,theta : word;
begin
 form1.refresh;
 delta:=round(360/n);
 theta:=0;
 r:=200;
 for i:=1 to n do
 begin
  Gxy[i,1]:=round(r*cos(theta*pi/180));
  Gxy[i,2]:=round(r*sin(theta*pi/180));
  circle(Gxy[i,1],Gxy[i,2],i);
  theta:=theta+delta;
 end;
 for i:=1 to n do
  for j:=1 to n do
   if i<>j then
    if (Graph[i,j]<>0) then
    begin
     if (Graph[i,j] <> Graph2[i,j]) then
     canvas.pen.color:=clGray
     else  canvas.pen.color:=clYellow;
     line(Gxy[i,1],Gxy[i,2],Gxy[j,1],Gxy[j,2]);
    end;
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
if form1.width<>1024 then
showmessage('Bu programı 1024 x 768
             piksel çözünürlükte çalıştırınız.');
end;

procedure TForm1.Image1Click(Sender: TObject);
begin
 label1.visible:=false;
 label2.visible:=false;
 label3.visible:=false;
 image1.visible:=false;
 form1.bordericons:=[bisystemmenu];
 stringgrid1.visible:=true;
 spinedit1.visible:=true;
 button1.visible:=true;
 label4.visible:=true;
end;

procedure TForm1.SpinEdit1Change(Sender: TObject);
begin
 if (spinedit1.value>1) and (spinedit1.value<17) then
 begin
  stringgrid1.colcount:=spinedit1.value+1;
  stringgrid1.rowcount:=spinedit1.value+1;
 end;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
{ Boyut Tamam } var
 i,j : byte;
begin
 button1.visible:=false;
 n:=spinedit1.value;
 spinedit1.visible:=false;
 label4.visible:=false;
 for i:=1 to n do
 begin
  stringgrid1.cells[i,0]:=chr(64+i);
  stringgrid1.cells[0,i]:=chr(64+i);
 end;
 for i:=1 to n do
  for j:=1 to n do
   stringgrid1.cells[i,j]:=chr(48);
 stringgrid1.options:=stringgrid1.options+[goediting];
 label5.visible:=true;
 button2.visible:=true;
end;

procedure TForm1.Button2Click(Sender: TObject);   { Graf Önce }
var
 i,j,k : word;
begin
 k:=1;
 for i:=1 to n do
  for j:=1 to n do
   if (strtoint(stringgrid1.cells[i,j])<0) or
      (strtoint(stringgrid1.cells[i,j])>1) or
      (strtoint(stringgrid1.cells[i,j])
           <> strtoint(stringgrid1.cells[j,i])) then k:=0;
 if (k=1) then
 begin
  label5.visible:=false;
  button2.visible:=false;
  stringgrid1.options:=stringgrid1.options-[goediting];
  for i:=1 to n do
   for j:=1 to n do
   begin
    Graph[i,j]:=strtoint(stringgrid1.cells[j,i]);
    Graph[i,i]:=0;
    Graph2[i,j]:=strtoint(stringgrid1.cells[j,i]);
    Graph2[i,i]:=0;
   end;
  for i:=1 to n do
   IsoV[i]:=0;
  stringgrid1.visible:=false;
  form1.refresh;
  os:='o';
  graphdraw;
  button2.visible:=false;
  button3.visible:=true;
  label6.visible:=true;
  label7.visible:=true;
  spinedit2.visible:=true;
  spinedit3.visible:=true;
```

```
  p:=spinedit2.value*n;
  maxgeneration:=spinedit3.value;
 end
    else showmessage('H A T A L I   V E R İ   G İ R İ Ş İ');
end;

procedure TForm1.Button3Click(Sender: TObject);   { Hesapla }
label
 es2yenidensec,generationnext,son;
var
 i,j,k,l,m,c,z,it : word;
 s                : string;
 sum              : longint;
 stop             : boolean;
 kv,delta,g1s,jt  : word;
 es               : string;
 dv               : array[1..maxn,1..2] of word;
 ff               : array[1..maxp,1..2] of real;
 r,Er,resultr     : real;
 ffp              : array[1..maxp] of real;

 function condisp01:byte;
 var
  i,j,k : word;
  cd    : byte;
 begin
  cd:=1;
  for i:=1 to n do
  begin
   for j:=1 to n do
    if (Graph2[i,j] <> 0) then Distance[i,j]:=Graph2[i,j]
                          else Distance[i,j]:=MAX;
   Distance[i,i]:=0;
  end;
  for k:=1 to n do
   for i:=1 to n do
    for j:=1 to n do
     if (Distance[i,k] + Distance[k,j] < Distance[i,j])
           then Distance[i,j]:=Distance[i,k] + Distance[k,j];
  for i:=1 to n do
  begin
   sum:=0;
   for j:=1 to n do
   begin
    if (Distance[i,j]=MAX) and (V[i]=1) and (V[j]=1) then cd:=0;
    sum:=sum+Distance[i,j];
   end;
   if (sum=(n-1)*MAX) and (V[i]=1) then cd:=0;
  end;
  if cd=0 then condisp01:=1
          else condisp01:=0;
 end;

begin
 k:=1;
 for i:=1 to n do
  for j:=1 to n do
   if (strtoint(stringgrid1.cells[i,j])<0) or
```

```
      (strtoint(stringgrid1.cells[i,j])>1) or
      (strtoint(stringgrid1.cells[i,j])
        <> strtoint(stringgrid1.cells[j,i])) then k:=0;
 if (k=1) then
 begin
  button3.visible:=false;
  stringgrid1.options:=stringgrid1.options-[goediting];

  for i:=1 to n do
  begin
   dv[i,1]:=i;
   dv[i,2]:=0;
   for j:=1 to n do
   begin
    Graph[i,j]:=strtoint(stringgrid1.cells[j,i]);
    Graph[i,i]:=0;
    dv[i,2]:=dv[i,2]+Graph[i,j];
    Graph2[i,j]:=strtoint(stringgrid1.cells[j,i]);
    Graph2[i,i]:=0;
   end;
  end;

 for i:=1 to n do v[i]:=1;
 if condisp01=1 then
 begin
  form2.Memo1.Clear;
  goto son;
 end;

 for i:=1 to n do
   for j:=i+1 to n do
    if dv[i,2]<dv[j,2] then
    begin
     k:=dv[i,1];
     dv[i,1]:=dv[j,1];
     dv[j,1]:=k;
     k:=dv[i,2];
     dv[i,2]:=dv[j,2];
     dv[j,2]:=k;
    end;

  delta:=dv[n,2];

  form2.Memo1.Clear;

{ *** Genetic Algorithm *** }

  for i:=1 to p do
   for j:=1 to n do
    pop[i,j]:=0;

  for i:=1 to n do
   pop[i,dv[i,1]]:=1;

{ Initial Population }
  randomize;
  jt:=0;
  for i:=n+1 to p do
```

```
  begin
   g1s:=random(delta)+1;
   k:=0;
   repeat
    repeat
     j:=random(n)+1;
    until (pop[i,j]=0) and (j<>jt);
    jt:=j;
    pop[i,j]:=1;
    k:=k+1;
   until k=g1s;
  end;

  it:=0;

  resultr:=0;

generationnext:
  it:=it+1;
{ Evalution Fitting Function }
  Er:=0;
  for k:=1 to p do
  begin
   for i:=1 to n do
    V[i]:=1;
   for i:=1 to n do
    for j:=1 to n do
     graph2[i,j]:=graph[i,j];
   g1s:=0;
   for j:=1 to n do
    if pop[k,j]=1 then
    begin
     V[j]:=0;
     g1s:=g1s+1;
     for i:=1 to n do
      graph2[j,i]:=0;
     for i:=1 to n do
      graph2[i,j]:=0;
    end;
   ff[k,1]:=k;
   ff[k,2]:=(1/g1s)+condisp01;
{ Save Best Person }
   if (ff[k,2]>resultr) then
   begin
    resultr:=ff[k,2];
    for j:=1 to n do
     result[j]:=pop[trunc(ff[k,1]),j];
   end;
   Er:=Er+ff[k,2];
  end;

{ Evalution Percentage ff }
  for k:=1 to p do
  begin
   r:=100*ff[k,2]/Er;
   if k=1 then ffp[k]:=r
          else ffp[k]:=ffp[k-1]+r;
  end;
```

```
{ Cross 1 point }
  for k:=1 to (p div 2) do
  begin
   l:=random(100)+1;
   i:=0;
   while (ffp[i+1] < l) and (i+1<p)  do
    i:=i+1;
   i:=i+1;
   if i=0 then i:=1;
es2yenidensec:
   m:=random(100)+1;
   j:=0;
   while (ffp[j+1] <= m) and (j+1<=p) do
    j:=j+1;
   j:=j+1;
   if j=0 then j:=1;
   if i=j then goto es2yenidensec;
   c:=random(n)+1;
   for z:=1 to c do
    pop2[k,z]:=pop[i,z];
   for z:=c+1 to n do
    pop2[k,z]:=pop[j,z];
   j:=0;
   for i:=1 to n do
    if pop2[k,i]=1 then j:=j+1;
   if j=0 then
    pop2[k,random(n)+1]:=1;
  end;

{ Sort }
  for i:=1 to n do
   for j:=i+1 to n do
    if ff[i,2]<ff[j,2] then
    begin
     r:=ff[i,1];
     ff[i,1]:=ff[j,1];
     ff[j,1]:=r;
     r:=ff[i,2];
     ff[i,2]:=ff[j,2];
     ff[j,2]:=r;
    end;

{ Elitism }
  for i:=(p div 2)+1 to p do
   for j:=1 to n do
    pop2[i,j]:=pop[trunc(ff[i-(p div 2),1]),j];

{ Mutation }
{ Tercih edilmedi ! }

  if it<maxgeneration then goto generationnext;

son:

  kv:=0;

  for i:=1 to n do
```

```
 V[i]:=1;
for i:=1 to n do
 for j:=1 to n do
  graph2[i,j]:=graph[i,j];
g1s:=0;
for j:=1 to n do
 if result[j]=1 then
 begin
  V[j]:=0;
  for i:=1 to n do
   graph2[j,i]:=0;
  for i:=1 to n do
   graph2[i,j]:=0;
 end;

for i:=1 to n do
begin
 for j:=1 to n do
  if (Graph2[i,j] <> 0) then Distance[i,j]:=Graph2[i,j]
                        else Distance[i,j]:=MAX;
 Distance[i,i]:=0;
end;

for k:=1 to n do
 for i:=1 to n do
  for j:=1 to n do
   if (Distance[i,k] + Distance[k,j] < Distance[i,j]) then
    Distance[i,j]:=Distance[i,k] + Distance[k,j];

for i:=1 to n do
begin
 IsoV[i]:=0;
 sum:=0;
 for j:=1 to n do
  sum:=sum+Distance[i,j];
 if (sum=(n-1)*MAX) and (V[i]=1) then IsoV[i]:=1;
end;

es:='';

for i:=1 to n do
 if (result[i]=1) then
 begin
  inc(kv);
  es:=es+chr(64+i)+' ';
 end;

s:='';
form2.memo1.lines.add(s);
s:='k(G) = '+inttostr(kv);
form2.memo1.lines.add(s);

s:='';
form2.memo1.lines.add(s);

s:='Silinen Tepe(ler) : '+es;
form2.memo1.lines.add(s);
```

```
  s:='';
  form2.memo1.lines.add(s);
  for i:=1 to n do
   if IsoV[i]=1 then s:=s+chr(64+i)+' ';
  s:='İzole Tepe(ler)   : '+s;
  form2.memo1.lines.add(s);

  button4.visible:=true;
  label6.visible:=false;
  label7.visible:=false;
  spinedit2.visible:=false;
  spinedit3.visible:=false;

 end
    else showmessage('H A T A L I   V E R İ   G İ R İ Ş İ');
end;

procedure TForm1.Button4Click(Sender: TObject);   { Graf Sonra }
begin
 stringgrid1.options:=stringgrid1.options+[goediting];
 stringgrid1.visible:=false;
 form1.refresh;
 os:='s';
 graphdraw;
 button4.visible:=false;
 button5.visible:=true;
end;

procedure TForm1.Button5Click(Sender: TObject);   { Sonuçlar }
begin
 form2.show;
end;

procedure TForm1.SpinEdit2Change(Sender: TObject);
begin
 p:=spinedit2.value*n;
end;

procedure TForm1.SpinEdit3Change(Sender: TObject);
begin
 maxgeneration:=spinedit3.value;
end;

end.
```

Ek 2. Tepe Birleştirilmişlik Sayısını C Dili ile Hesaplayan Program Kaynak Kodu

```
#include <stdio.h> #include <stdlib.h> #include <string.h>

#define SS 1000 #define EBAS 32000

int Dijkstra(); int Hesapla(int cc);

int
M,ct,iilk=0,iiilk,iiiilk,ek1,ead,bas,tcon,s,sakla,CS[SS]
       ,FCN[SS],GRAF[SS][SS],GRAF1[SS][SS],GRAF2[SS][SS];
char
ROTA[SS][SS]={NULL},ROTA1[SS][SS]={NULL},ROTA2[SS][SS]
      ={NULL},ROTAS[SS][SS]={NULL};


void main() {

    int i,j,k,l,aa,bb,hh,gg,ilk,cvp,min,hedef,tg=0,tam=0
          ,tkrr=0,tepecon,vcon,ACS[SS],VC[SS],D[SS]={0};

    FILE *f;

    f=fopen("BM.txt","r");

    fscanf(f,"%d",&M);

    for(i=0;i<M;i++)
    {
     for(j=0;j<M;j++)
     {
         fscanf(f,"%d",&GRAF[i][j]);
         GRAF1[i][j]=GRAF[i][j];
         if (GRAF[i][j]==1) D[i]++;
     }
     if (D[i]==M-1) tg++;
    }
    fclose(f);

    if (tg==M)
    {
        tam=1;
        goto son;
    }

tkrar:
    tkrr++;
    if (tkrr==1)
    {
        for(i=0;i<M;i++)
```

```
        if ((i==0) || (min>D[i]))
        {
           min=D[i];
           bas=i;
        }
}
else
{
    bas=hedef;
    iilk=0;
}

if (min==1)
{
    printf("\n");
    printf("Connectivity= 1\n");
    printf("Atilacak Tepe= ");
    for(i=0;i<M;i++)
    if (GRAF[bas][i]==1) printf("%d",i);
    printf("\n");
    goto son;
}

ead=bas;
cvp=Dijkstra();

if (cvp==0)
{
    printf("Graf Baglantisiz...");
    goto son;
}

ilk=0;

for(i=0;i<M;i++)
{
    ead=bas;
    ct=0;

    if ((GRAF[ead][i]!=1) && (ead!=i))
    {
     aa=strlen(ROTA1[i]);
     bb=ROTA1[i][aa-1]-65;
     CS[ct]=bb;
     ct++;

     for(j=0;j<M;j++)
     {
         GRAF[bb][j]=0;
         GRAF[j][bb]=0;
         FCN[j]=1001;
     }

    iiilk=0;
    cvp=Dijkstra();

    if (cvp==0)
    {
```

```
            printf("\n");
            printf("Connectivity = 1 \n");
            printf("Atilacak Tepe=%d",CS[0]);
            printf("\n");
            goto son;
        }

        hh=0;
        ek1=0;
label1:

        for(k=0;k<M;k++)
            for(l=0;l<M;l++)
                GRAF2[k][l]=GRAF[k][l];

        aa=strlen(ROTA[i]);

             if (aa>2)
             {
                 gg=0;
                 iiiilk=0;
                 cvp=Hesapla(i);

                 for(k=0;k<M;k++)
                    for(l=0;l<M;l++)
                     ROTA[k][l]= ROTAS[k][l];

                 for(j=aa-1;j>1;j--)
                 {

                     if (hh==0)
                     {
                         bb=ROTA2[i][j]-65;
                     }
                        else
                     {
                            bb=ROTA[i][j]-65;
                     }

                     cvp=Hesapla(bb);

                     for(k=0;k<M;k++)
                        for(l=0;l<M;l++)
                         ROTA[k][l]= ROTAS[k][l];

                          if (cvp==0)
                          {
                              gg++;
                              CS[ct]=bb;
                              ct++;
                              goto label;
                          }
                 }

                 if (gg==0)
                 {
                     if (hh==0)
                     {
```

```
                        CS[ct]=ROTA2[i][1]-65;
                        ct++;
                    }
                    else
                    {
                        CS[ct]=ROTAS[i][1]-65;
                        ct++;
                    }
                }

            }
            else
            {
                if (hh==0)
                {
                    CS[ct]=ROTA2[i][1]-65;
                }
                 else
                {
                    CS[ct]=ROTA[i][1]-65;
                }
                ct++;
            }

label:

     for(k=0;k<M;k++)
           for(l=0;l<M;l++)
                 GRAF[k][l]=GRAF2[k][l];

           for(j=0;j<M;j++)
           {
             GRAF[CS[ct-1]][j]=0;
             GRAF[j][CS[ct-1]]=0;
           }
     sakla=0;
     cvp=Dijkstra();

     if (cvp==1)
     {
         ek1++;
         hh++;
         goto label1;

     }

     if  (ilk==0)
           {
             ilk++;
             tepecon=ct;
             hedef=i;
             for(j=0;j<tepecon;j++)
             ACS[j]=CS[j];
           }

     if ((ct<tepecon) || ((ct==tepecon)
            && (strlen(ROTA1[i])<strlen(ROTA1[hedef]))))
           {
```

```
              tepecon=ct;
              hedef=i;
              for(j=0;j<tepecon;j++)
              ACS[j]=CS[j];
            }

      for(k=0;k<M;k++)
              for(l=0;l<M;l++)
                GRAF[k][l]=GRAF1[k][l];
         }

      }

       if (tkrr==1)
      {
        vcon=tepecon;
        for(i=0;i<tepecon;i++)
        VC[i]=ACS[i];
        goto tkrar;
      }
      else
      {
          if (tepecon<vcon)
          {
            vcon=tepecon;
            for(i=0;i<tepecon;i++)
            VC[i]=ACS[i];
          }
      }

      printf("\n");
      printf("Connectivity=%d\n",vcon);
      printf("Atilacak Tepeler= ");
      for(i=0;i<vcon;i++)
      printf(" %d ",VC[i]);

son:
      if (tam==1)
      {
          printf("\n");
          printf("Conncectivity=%d \n",M-1);
          printf("Atilacak Tepeler=");

          for(i=1;i<M;i++)
          printf("%d ",i);
      }

}

int Hesapla(int cc) {
   int  zz,i,j,h,kk,a,z,b,acon,cvp1;

      ead=bas;
      s=0;
      h=0;
      zz=0;
      kk=cc;
      h=h+ek1;
```

```
      for(i=0;i<M;i++)
      FCN[i]=101;

      for(i=0;i<M;i++)
         for(j=0;j<M;j++)
           GRAF[i][j]=GRAF2[i][j];
lab:
      h++;
      if (h==1)
      {
          a=strlen(ROTA2[kk]);
      }
      else
      {
          a=strlen(ROTA[kk]);
      }

      for(z=1;z<a;z++)
      {
         if (h==1)
         {
             b=ROTA2[kk][z]-65;
         }
         else
         {
             b=ROTA[kk][z]-65;
         }

         for(i=0;i<M;i++)
         {
             GRAF[i][b]=0;
             GRAF[b][i]=0;
         }

             FCN[zz]=b;
             zz++;
      }

      cvp1=Dijkstra();

      if (cvp1==1) goto lab;

      acon=s;

      if (iiiilk==0)
      {
           iiiilk++;
           tcon=acon;
           return 1;
      }

      if (tcon==acon)
      {
           return 1;
      }
      else
      {
```

```
            return 0;
      }

}

int Dijkstra() {

      char *ptr, ELEALINDI[SS]={0};
      int   i,j,ek,k1=0,k2=0,k3=0,EKM[SS];


      for(i=0;i<M;i++)
         for(j=0;j<M;j++)
            ROTA[i][j]=NULL;

      s++;
      for(i=0;i<M;i++)
      EKM[i] = EBAS;

      ead=bas;

      EKM[ead]=0;

      for(i=0;i<M;i++)
      {
         for(j=0;j<M;j++)
             if (!ELEALINDI[j])
                 if (GRAF[ead][j]!=0)
                     if (EKM[j]>GRAF[ead][j]+EKM[ead])
                     {
                        EKM[j]=GRAF[ead][j]+EKM[ead];

                        strcpy(ROTA[j],ROTA[ead]);
                        ptr=ROTA[j];
                         while (*ptr!=NULL)
                            ++ptr;
                        *ptr='A'+ead;
                     }
          ek= EBAS;
          for(j=0;j<M;j++)
              if (!ELEALINDI[j])
                  if (EKM[j]<ek) {
                      ek=EKM[j];  ead=j;
                 }
         ELEALINDI[ead]=1;
      }

      if (sakla==0)
      {
       for(i=0;i<M;i++)
           for(j=0;j<M;j++)
             ROTAS[i][j]= ROTA[i][j];
       sakla++;
      }

      if (iilk==0)
      {
       for(i=0;i<M;i++)
```

```
            for(j=0;j<M;j++)
              ROTA1[i][j]= ROTA[i][j];
      iilk++;
      }

      if (iiilk==0)
      {
       for(i=0;i<M;i++)
            for(j=0;j<M;j++)
              ROTA2[i][j]= ROTA[i][j];
      iiilk++;
      }

      for(i=0;i<M;i++)
      {
            k2=0;k3=0;

            for(j=0;j<M;j++)
                 if (FCN[j]==i) k2++;

            for(j=0;j<ct;j++)
                 if (CS[j]==i)  k3++;

            if ((ELEALINDI[i]==0) && (k2==0) && (k3==0)) k1++;
      }

      if (k1!=0) return 0;

      return 1;

}
```

Ek 3. Tepe Birleştirilmişlik Probleminin GAMS ile Çözümü

```
GAMS Rev 148  x86/MS Windows
G e n e r a l   A l g e b r a i c   M o d e l i n g
S y s t e m C o m p i l a t i o n


   2  Set i / i1*i4 / ;
   3  Alias(i,j,k,p,q);
   4  Scalar M ;
   5  M=4*4*4;
   6
   7  * Distance between vertice
   8  Table d(i,j)
   9                              i1   i2   i3   i4
  10                         i1        1
  11                         i2   1         1    1
  12                         i3        1         1
  13                         i4        1     1 ;
  14
  15  Variables
  16  x(i,j,p,q)
  17  xx(i)
  18  y(i)
  19  z
  20  ;
  21
  22  binary variable x,y;
  23
  24  xx.lo(i)=1;
  25
  26  EQUATIONS
  27
  28  OBJ
  29  CONSTRAINT1(p,q)
  30  CONSTRAINT2(p,q)
  31  CONSTRAINT3(k,p,q)
  32  CONSTRAINT4
  33  CONSTRAINT6(i)
  34  CONSTRAINT7(i);
  35
  36
  37  OBJ..       z =E= SUM(i,xx(i)) ;
  38
  39  CONSTRAINT1(p,q)$(ord(p)<card(p) and ord(p)<ord(q))
      ..SUM(i$(ord(i) ne ord(p)),x(p,i,p,q)*d(p,i))=E=1;
  40
  41  CONSTRAINT2(p,q)$(ord(p)<card(p) and ord(p)<ord(q))
       ..SUM(i$(ord(i) ne ord(q)),x(i,q,p,q)*d(i,q))=E=1;
  42
```

```
 43  CONSTRAINT3(k,p,q)$(ord(p)<card(p) and ord(p)<ord(q)
     and ord(k) ne ord(p) and ord(k) ne ord(q))
     ..SUM(i,x(i,k,p,q)*d(i,k))=E=SUM(j,x(k,j,p,q)*d(k,j));
 44
 45  CONSTRAINT4.. SUM(p$(ord(p)<card(p)),SUM(q$(ord(q)>
     ord(p)), SUM((i,j), d(i,j)*x(i,j,p,q)*xx(i)*xx(j)/
     (xx(p)*xx(q))    )   )  )=G=M+1;
 46
 47  CONSTRAINT6(i).. xx(i)=G=M*y(i)+1;
 48
 49  CONSTRAINT7(i).. xx(i)=L=M*y(i)+1;
 50
 51  MODEL TBM /ALL/;
 52
 53  SOLVE TBM using MINLP minimizing z;


COMPILATION TIME     =        0.000 SECONDS      3 Mb  WIN225-148
May 29, 2007 GAMS Rev 148  x86/MS Windows
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Equation Listing    SOLVE TBM Using MINLP From line 53


---- OBJ  =E=

OBJ..  - xx(i1) - xx(i2) - xx(i3) - xx(i4) + z =E= 0 ; (LHS = -4,
INFES = 4 ***)


---- CONSTRAINT1  =E=

CONSTRAINT1(i1,i2)..  x(i1,i2,i1,i2) =E= 1 ; (LHS = 0, INFES = 1
***)

CONSTRAINT1(i1,i3)..  x(i1,i2,i1,i3) =E= 1 ; (LHS = 0, INFES = 1
***)

CONSTRAINT1(i1,i4)..  x(i1,i2,i1,i4) =E= 1 ; (LHS = 0, INFES = 1
***)

REMAINING 3 ENTRIES SKIPPED


---- CONSTRAINT2  =E=

CONSTRAINT2(i1,i2)..  x(i1,i2,i1,i2) + x(i3,i2,i1,i2) +
x(i4,i2,i1,i2) =E= 1 ;

      (LHS = 0, INFES = 1 ***)

CONSTRAINT2(i1,i3)..  x(i2,i3,i1,i3) + x(i4,i3,i1,i3) =E= 1 ;

      (LHS = 0, INFES = 1 ***)

CONSTRAINT2(i1,i4)..  x(i2,i4,i1,i4) + x(i3,i4,i1,i4) =E= 1 ;

      (LHS = 0, INFES = 1 ***)
```

```
REMAINING 3 ENTRIES SKIPPED


---- CONSTRAINT3  =E=

CONSTRAINT3(i1,i2,i3)..  - x(i1,i2,i2,i3) + x(i2,i1,i2,i3) =E= 0 ;
(LHS = 0)

CONSTRAINT3(i1,i2,i4)..  - x(i1,i2,i2,i4) + x(i2,i1,i2,i4) =E= 0 ;
(LHS = 0)

CONSTRAINT3(i1,i3,i4)..  - x(i1,i2,i3,i4) + x(i2,i1,i3,i4) =E= 0 ;
(LHS = 0)

REMAINING 9 ENTRIES SKIPPED


---- CONSTRAINT4  =G=

CONSTRAINT4..  (1)*x(i1,i2,i1,i2) + (1)*x(i1,i2,i1,i3) +
(1)*x(i1,i2,i1,i4)

      + (1)*x(i1,i2,i2,i3) + (1)*x(i1,i2,i2,i4) + (1)*x(i1,i2,i3,i4

      + (1)*x(i2,i1,i1,i2) + (1)*x(i2,i1,i1,i3) + (1)*x(i2,i1,i1,i4

      + (1)*x(i2,i1,i2,i3) + (1)*x(i2,i1,i2,i4) + (1)*x(i2,i1,i3,i4

      + (1)*x(i2,i3,i1,i2) + (1)*x(i2,i3,i1,i3) + (1)*x(i2,i3,i1,i4

      + (1)*x(i2,i3,i2,i3) + (1)*x(i2,i3,i2,i4) + (1)*x(i2,i3,i3,i4

      + (1)*x(i2,i4,i1,i2) + (1)*x(i2,i4,i1,i3) + (1)*x(i2,i4,i1,i4

      + (1)*x(i2,i4,i2,i3) + (1)*x(i2,i4,i2,i4) + (1)*x(i2,i4,i3,i4

      + (1)*x(i3,i2,i1,i2) + (1)*x(i3,i2,i1,i3) + (1)*x(i3,i2,i1,i4

      + (1)*x(i3,i2,i2,i3) + (1)*x(i3,i2,i2,i4) + (1)*x(i3,i2,i3,i4

      + (1)*x(i3,i4,i1,i2) + (1)*x(i3,i4,i1,i3) + (1)*x(i3,i4,i1,i4

      + (1)*x(i3,i4,i2,i3) + (1)*x(i3,i4,i2,i4) + (1)*x(i3,i4,i3,i4

      + (1)*x(i4,i2,i1,i2) + (1)*x(i4,i2,i1,i3) + (1)*x(i4,i2,i1,i4

      + (1)*x(i4,i2,i2,i3) + (1)*x(i4,i2,i2,i4) + (1)*x(i4,i2,i3,i4

      + (1)*x(i4,i3,i1,i2) + (1)*x(i4,i3,i1,i3) + (1)*x(i4,i3,i1,i4

      + (1)*x(i4,i3,i2,i3) + (1)*x(i4,i3,i2,i4) + (1)*x(i4,i3,i3,i4

      + (0)*xx(i1) + (0)*xx(i2) + (0)*xx(i3) + (0)*xx(i4) =G= 65 ;

      (LHS = 0, INFES = 65 ***)


---- CONSTRAINT6  =G=
```

```
CONSTRAINT6(i1)..  xx(i1) - 64*y(i1) =G= 1 ; (LHS = 1)

CONSTRAINT6(i2)..  xx(i2) - 64*y(i2) =G= 1 ; (LHS = 1)

CONSTRAINT6(i3)..  xx(i3) - 64*y(i3) =G= 1 ; (LHS = 1)

REMAINING ENTRY SKIPPED


---- CONSTRAINT7  =L=

CONSTRAINT7(i1)..  xx(i1) - 64*y(i1) =L= 1 ; (LHS = 1)

CONSTRAINT7(i2)..  xx(i2) - 64*y(i2) =L= 1 ; (LHS = 1)

CONSTRAINT7(i3)..  xx(i3) - 64*y(i3) =L= 1 ; (LHS = 1)

REMAINING ENTRY SKIPPED
```



```
---- x

x(i1,i2,i1,i2)
                (.LO, .L, .UP = 0, 0, 1)
        1       CONSTRAINT1(i1,i2)
        1       CONSTRAINT2(i1,i2)
       (1)      CONSTRAINT4

x(i1,i2,i1,i3)
                (.LO, .L, .UP = 0, 0, 1)
        1       CONSTRAINT1(i1,i3)
        1       CONSTRAINT3(i2,i1,i3)
       (1)      CONSTRAINT4

x(i1,i2,i1,i4)
                (.LO, .L, .UP = 0, 0, 1)
        1       CONSTRAINT1(i1,i4)
        1       CONSTRAINT3(i2,i1,i4)
       (1)      CONSTRAINT4

REMAINING 45 ENTRIES SKIPPED

---- xx

xx(i1)
                (.LO, .L, .UP = 1, 1, +INF)
       -1       OBJ
       (0)      CONSTRAINT4
        1       CONSTRAINT6(i1)
        1       CONSTRAINT7(i1)

xx(i2)
                (.LO, .L, .UP = 1, 1, +INF)
```

```
       -1       OBJ
       (0)      CONSTRAINT4
        1       CONSTRAINT6(i2)
        1       CONSTRAINT7(i2)

xx(i3)
                (.LO, .L, .UP = 1, 1, +INF)
       -1       OBJ
       (0)      CONSTRAINT4
        1       CONSTRAINT6(i3)
        1       CONSTRAINT7(i3)

REMAINING ENTRY SKIPPED

---- y

y(i1)
                (.LO, .L, .UP = 0, 0, 1)
      -64       CONSTRAINT6(i1)
      -64       CONSTRAINT7(i1)

y(i2)
                (.LO, .L, .UP = 0, 0, 1)
      -64       CONSTRAINT6(i2)
      -64       CONSTRAINT7(i2)

y(i3)
                (.LO, .L, .UP = 0, 0, 1)
      -64       CONSTRAINT6(i3)
      -64       CONSTRAINT7(i3)

REMAINING ENTRY SKIPPED

---- z

z
                (.LO, .L, .UP = -INF, 0, +INF)
        1       OBJ

GAMS Rev 148  x86/MS Windows
G e n e r a l   A l g e b r a i c   M o d e l i n g
S y s t e m Model Statistics    SOLVE TBM Using MINLP From line 53


MODEL STATISTICS

BLOCKS OF EQUATIONS           7     SINGLE EQUATIONS           34
BLOCKS OF VARIABLES           4     SINGLE VARIABLES           57
NON ZERO ELEMENTS           145     NON LINEAR N-Z             52
DERIVATIVE POOL              66     CONSTANT POOL              16
CODE LENGTH               1,394     DISCRETE VARIABLES         52


GENERATION TIME      =        0.000 SECONDS      4 Mb  WIN225-148
May 29, 2007


EXECUTION TIME       =        0.000 SECONDS      4 Mb  WIN225-148
```

May 29, 2007 GAMS Rev 148 x86/MS Windows
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m
Solution Report SOLVE TBM Using MINLP From line 53

S O L V E S U M M A R Y

MODEL	TBM	OBJECTIVE	z
TYPE	MINLP	DIRECTION	MINIMIZE
SOLVER	BARON	FROM LINE	53

**** SOLVER STATUS 1 NORMAL COMPLETION **** MODEL STATUS 8
INTEGER SOLUTION **** OBJECTIVE VALUE 68.0000

RESOURCE USAGE, LIMIT	0.110	1000.000
ITERATION COUNT, LIMIT	0	10000
EVALUATION ERRORS	0	0

GAMS/BARON Jun 1, 2007 WIN.BA.NA 22.5 012.000.000.VIS P3PC

Branch And Reduce Optimization Navigator Nikolaos Sahinidis and
Mohit Tawarmalani The Optimization Firm, LLC.

Total time elapsed	000:00:00,	in seconds: 0.12
on parsing	000:00:00,	in seconds: 0.00
on preprocessing	000:00:00,	in seconds: 0.11
on navigating	000:00:00,	in seconds: 0.02
on relaxed	000:00:00,	in seconds: 0.00
on local	000:00:00,	in seconds: 0.00
on tightening	000:00:00,	in seconds: 0.00
on marginals	000:00:00,	in seconds: 0.00
on probing	000:00:00,	in seconds: 0.00

Total no. of BaR iterations:	1
Best solution found at node:	-1
Max. no. of nodes in memory:	1

Solution = 68 best solution found during preprocessing Best
possible =61.8181818182 Absolute gap=6.1818181818 optca = 1E-9
Relative gap = 0.10000 optcr = 0.1

	LOWER	LEVEL	UPPER	MARGINAL
---- EQU OBJ	.	.	.	1.000

---- EQU CONSTRAINT1

	LOWER	LEVEL	UPPER	MARGINAL		LOWER	LEVEL	UPPER	MARGINAL
i1.i2	1.000	1.000	1.000	.	i1.i3	1.000	1.000	1.000	.
i2.i3	1.000	1.000	1.000	.	i2.i4	1.000	1.000	1.000	.
i3.i4	1.000	1.000	1.000	.					

---- EQU CONSTRAINT2

```
         LOWER     LEVEL     UPPER    MARGINAL

i1.i2     1.000     1.000     1.000      . i1.i3   1.000     1.000
1.000      . i1.i4     1.000     1.000     1.000     . i2.i3 1.000
1.000     1.000      . i2.i4     1.000     1.000     1.000 . i3.i4
1.000     1.000     1.000      .

---- EQU CONSTRAINT3

            LOWER     LEVEL     UPPER    MARGINAL

i1.i2.i3     .         .         .         . i1.i2.i4     . .
.         . i1.i3.i4      .         .         .        . i2.i1.i3
.         .         .         . i2.i1.i4      . .         .
. i2.i3.i4      .         .         .         . i3.i1.i2      .
.         .         . i3.i1.i4      . .                 . i3.i2.i4
.         .         .         . i4.i1.i2      .         .         .
. i4.i1.i3      . .         .         .         . i4.i2.i3      .         .
.         .

                       LOWER     LEVEL     UPPER    MARGINAL

---- EQU CONSTRAIN~    65.000   393.000     +INF       .

---- EQU CONSTRAINT6

      LOWER     LEVEL     UPPER    MARGINAL

i1     1.000     1.000     +INF       . i2     1.000     1.000 +INF
. i3     1.000     1.000     +INF       . i4     1.000 1.000
+INF       .

---- EQU CONSTRAINT7

      LOWER     LEVEL     UPPER    MARGINAL

i1     -INF      1.000     1.000      . i2     -INF    1.000 1.000
. i3     -INF      1.000     1.000      . i4     -INF 1.000
1.000      .

---- VAR x

               LOWER     LEVEL     UPPER    MARGINAL

i1.i2.i1.i2      .         1.000     1.000      EPS i1.i2.i1.i3 .
1.000     1.000      EPS i1.i2.i1.i4      .         1.000 1.000
EPS i1.i2.i2.i3      .         .         1.000    EPS i1.i2.i2.i4
.         .         1.000      EPS i1.i2.i3.i4 .         .
1.000      EPS i2.i1.i1.i2      .         . 1.000      EPS
i2.i1.i1.i3      .         .         1.000      EPS i2.i1.i1.i4
.         .         1.000      EPS i2.i1.i2.i3 .         .
1.000      EPS i2.i1.i2.i4      .         . 1.000      EPS
i2.i1.i3.i4      .         .         1.000      EPS i2.i3.i1.i2
.         .         1.000      EPS i2.i3.i1.i3 .         1.000
1.000      EPS i2.i3.i1.i4      .         . 1.000      EPS
i2.i3.i2.i3      .         1.000     1.000      EPS i2.i3.i2.i4
.         .         1.000      EPS i2.i3.i3.i4 .         .
1.000      EPS i2.i4.i1.i2      .         . 1.000      EPS
```

```
i2.i4.i1.i3      .          .         1.000      EPS i2.i4.i1.i4
.         1.000     1.000      EPS i2.i4.i2.i3 .          .
1.000      EPS i2.i4.i2.i4      .         1.000 1.000      EPS
i2.i4.i3.i4      .        1.000     1.000      EPS i3.i2.i1.i2
.          .         1.000      EPS i3.i2.i1.i3 .          .
1.000      EPS i3.i2.i1.i4      .          . 1.000      EPS
i3.i2.i2.i3      .          .         1.000      EPS i3.i2.i2.i4
.          .         1.000      EPS i3.i2.i3.i4 .         1.000
1.000      EPS i3.i4.i1.i2      .          . 1.000      EPS
i3.i4.i1.i3      .          .         1.000      EPS i3.i4.i1.i4
.          .         1.000      EPS i3.i4.i2.i3 .          .
1.000      EPS i3.i4.i2.i4      .          . 1.000      EPS
i3.i4.i3.i4      .          .         1.000      EPS i4.i2.i1.i2
.          .         1.000      EPS i4.i2.i1.i3 .          .
1.000      EPS i4.i2.i1.i4      .          . 1.000      EPS
i4.i2.i2.i3      .          .         1.000      EPS i4.i2.i2.i4
.          .         1.000      EPS i4.i2.i3.i4 .          .
1.000      EPS i4.i3.i1.i2      .          . 1.000      EPS
i4.i3.i1.i3      .          .         1.000      EPS i4.i3.i1.i4
.          .         1.000      EPS i4.i3.i2.i3 .          .
1.000      EPS i4.i3.i2.i4      .          . 1.000      EPS
i4.i3.i3.i4      .          .         1.000      EPS

---- VAR xx

       LOWER     LEVEL     UPPER    MARGINAL

i1     1.000     1.000     +INF       1.000 i2     1.000    65.000
+INF       1.000 i3     1.000     1.000     +INF     1.000 i4 1.000
1.000     +INF       1.000

---- VAR y

       LOWER     LEVEL     UPPER    MARGINAL

i1       .         .         1.000      EPS i2     .         1.000
1.000      EPS i3       .         .        1.000    EPS i4       . .
1.000      EPS

                       LOWER     LEVEL     UPPER    MARGINAL

---- VAR z              -INF     68.000     +INF        .


**** REPORT SUMMARY :         0     NONOPT
                              0 INFEASIBLE
                              0  UNBOUNDED
                              0     ERRORS
```