EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

MİNİMUM TEPE ÖRTÜSÜ PROBLEMİ ÜZERİNE

Onur UĞURLU

Tez Danışmanı : Prof. Dr. Urfat NURİYEV

İkinci Danışmanı : Yrd. Doç. Dr. Murat Erşen BERBERLER

Matematik Anabilim Dalı

Bilim Dalı Kodu : 619.03.03
Sunuş Tarihi : 09.01.2013

Bornova-İZMİR

2013

Onur UĞURLU tarafından YÜKSEK LİSANS TEZİ olarak sunulan **"MİNİMUM TEPE ÖRTÜSÜ PROBLEMİ ÜZERİNE"** başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 09.01.2013 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. Aysun AYTAÇ	**.....**
Üye	**:** Prof. Dr. Alpay KIRLANGIÇ	……....
Üye	**:** Yrd. Doç. Dr. Murat E. BERBERLER	**.....**
Üye	**:** Yrd. Doç. Dr. Mehmet KURT	….....

ÖZET

MİNİMUM TEPE ÖRTÜSÜ PROBLEMİ ÜZERİNE

UĞURLU, Onur

Yüksek Lisans Matematik Anabilim Dalı
Tez Danışmanı: Prof. Dr. Urfat NURİYEV
İkinci Danışmanı: Yrd. Doç. Dr. Murat Erşen BEBERLER
Ocak 2013, 43 sayfa

MİNİMUM TEPE ÖRTÜSÜ problemi NP-tam sınıfına ait bir teorik çizge optimizasyon problemidir. Problem bilgisayar bilimlerinde önemli bir rol oynamaktadır ve MİNİMUM TEPE ÖRTÜSÜ örneği olarak formüle edilebilen pek çok gerçek hayat uygulamasına sahiptir. MİNİMUM TEPE ÖRTÜSÜ problemleriyle karşılaşılan uygulama alanlarına örnek olarak iletişim ağları ve özellikle kablosuz ağlar ve bioinformatik verilebilir.

MİNİMUM TEPE ÖRTÜSÜ probleminin öneminden dolayı, birçok araştırmacı makul sürelerde kaliteli çözümler veren algoritmalar geliştirmeye odaklanmıştır. Literatürde MİNİMUM TEPE ÖRTÜSÜ problemi için kesin algoritmalar, yaklaşım algoritmaları, sezgisel algoritmalar ve evrimsel algoritmalar dahil bir çok algoritma önerilmiştir.

Bu tezde MİNİMUM TEPE ÖRTÜSÜ problemi ele alınmış, problem için geliştirilen çözüm yaklaşımları incelenmiş ve problem için yeni bir sezgisel algoritma ve yeni bir hibrid genetik algoritma önerilmiştir. Önerilen algoritmalar C++ dilinde kodlanmış ve kütüphane örnekleri üzerinde test edilerek literatür çalışmaları ile karşılaştırılmıştır. Hesaplama sonuçları önerilen algoritmaların kaliteli sonuçlar elde ettiğini göstermektedir.

Anahtar sözcükler: MİNİMUM TEPE ÖRTÜSÜ problemi, çizge kuramı, sezgisel algoritmalar, optimizasyon, NP-tam problemler, hibrid genetik algoritmalar

ABSTRACT

ON THE MINIMUM VERTEX COVER PROBLEM

UĞURLU, Onur

MSc in Department of Mathematics
Supervisor: Prof. Dr. Urfat NURİYEV
Co-Supervisor: Asist. Prof. Dr. Murat Erşen BERBERLER
January 2013, 43 pages

THE MINIMUM VERTEX COVER problem is a theoretical graph optimization problem which belongs to the class of NP-complete. The problem plays a central role in computer science and it has a plenty real life applications which can be formulated as instances of the MINIMUM VERTEX COVER. The communication networks, particularly in wireless networks and bioinformatics can be given as examples of such areas where the MINIMUM VERTEX COVER problem occurs.

Due to the importance of the MINIMUM VERTEX COVER problem, many researchers have focused on developing algorithms which give quality solutions in a reasonable time. In the literature, there are several algorithm have been proposed for MINIMUM VERTEX COVER problem such as exact algorithms, approximation algorithms, heuristic algorithms and evolutionary algorithms.

In this thesis, the MINIMUM VERTEX COVER problem has been studied, the solution approaches for the problem have been investigated, then a new heuristic algorithm and a new hybrid genetic algorithm have been proposed for the problem. The proposed algorithms are the algorithms have been implemented in C++, and have been tested on the benchmark instances, then the results have been compared with studies the literature works. The experimental results show that the proposed algorithms yield quality solutions.

Keywords: THE MINIMUM VERTEX COVER problem, graph theory, heuristic problems, optimization, NP-complete problems, hybrid genetic algorithms

TEŞEKKÜR

Bu çalışma süresince bilgi ve deneyimlerini esirgemeyen danışmanlarım Sayın Prof. Dr. Urfat NURİYEV'e ve Sayın Yrd. Doç. Dr. Murat Erşen BERBERLER'e, çalışmamda yardım ve önerilerini her zaman benimle paylaşan arkadaşım Gözde KIZILATEŞ'e, yaşantım boyunca desteklerini her zaman sürdüren aileme teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

Çizelge Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	Açıklama
$\delta(G)$	minimum tepe derecesi.
$\Delta(G)$	maksimum tepe derecesi.
$\overline{G}$	tümleyen çizge.
$\beta(G)$	minimum tepe örtüsünün boyutu.
$\alpha(G)$	maksimum bağımsız kümenin boyutu.
$\omega(G)$	maksimum klik boyutu.

Kısaltmalar

PA	Parçalama algoritması.
CBH	Continuous Based Heuristic.
QSH	Heuristic Based On Optimization Of A Quadratic Over A Sphere.
OCH	Optimized Crossover.

1.GİRİŞ

Matematik çok çeşitli alanlarıyla hayatımızda önemli bir yere sahiptir. Bu alanların en önemlilerinden biri yapısal modeller için kullanılan çizge kuramıdır. Çizge kuramının tarihsel yolculuğu 1736 yılında ünlü matematikçi Leonhard Euler tarafından yazılan Königsberg'in Yedi Köprüsü isimli makale ile başlar. Euler, problemi çözerken somut bir olayı modelleyip soyut bir şekle dönüştürerek çizge kuramının temellerini atmış ve ardından kuram Hamilton, Heawood, Whitney, Tutte, Ore, Erdös, Katona, Polya, Seshu ve Frank Harary gibi matematikçilerin eşsiz katkılarıyla bugünkü durumuna gelmiştir.

Çizge kuramı çok geniş bir çalışma alanına sahiptir. Özellikle somut problemlerin matematiksel olarak modellenip bilgisayara aktarılmasında çizge kuramının rolü büyüktür. Bununla birlikte çizgeler üzerine matematiksel anlamda geliştirilen birçok teorem ve algoritma yazılım dünyasında sıklıkla kullanılmaktadır. Basitçe bir çizge, tepeler olarak adlandırılan noktalar ve her biri bu tepeleri birleştiren, ayrıt olarak adlandırılan bileşenlerden oluşan bir yapıdır. Günlük hayatta "tepeler ve bu tepeler arasındaki ilişkilerin varlığını belirten ayrıtlar" şeklinde ifade edilebilen tüm problemler bu yöntem ile modellenebilir.

Çizge kuramının kullanım alanları gün geçtikçe artmaktadır. Fizik, kimya gibi temel bilimlerde, mühendislik uygulamalarında ve tıp alanında birçok problemin çözümü ve modellenmesi çizgelere dayandırılarak yapılmaktadır. Çizge kuramı yardımıyla modellenebilen eşleme, ulaşım, bilgisayar ağları ve akış problemleri genel olarak programlama problemleri olarak isimlendirilir.

Minimum tepe örtüsü problemi de en önemli çizge problemlerinden biridir. Kablosuz ağların daha geniş kapsama alanlarında yüksek performans gösterebilmesi için minimum sayıda kaynak seçilmesi, bir şehrin tüm yollarının görüntülenebilmesi için gerekli minimum sayıda kamera yerleştirilmesi ve bir müzenin güvenliği için minimum sayıda güvenlik elemanın atanması gibi birçok problem minimum tepe örtüsü problemi ile modellenerek çözülebilir.

Minimum tepe örtüsü problemi bilgisayar bilimlerinde oldukça önemli bir yere sahip olan bir çizge problemidir. Problem NP-tam sınıfına ait oluğu için polinom zamanda optimum çözüm bulmak imkansızdır. Problem, çözümü oldukça zor olmasına rağmen geniş uygulama alanı ve diğer çizge problemleriyle yakın bağlantısından dolayı araştırmacılar tarafından oldukça ilgi görmüştür. Literatürde minimum tepe örtüsü problemi için birçok çözüm yöntemi ve algoritma önerilmiştir. Buna rağmen problemin çözümüne yeni bir bakış açısı getirmek ya da var olan çözümleri geliştirebilmek, günlük hayattaki kullanımlarını iyileştirmesi yönünden önem taşımaktadır.

Bu tezde minimum tepe örtüsü problemi ele alınarak, problemin çözüm yöntemleri üzerinde araştırmalar yapılmış ve yeni sezgisel algoritmalar literatüre kazandırılmıştır.

Tezin izleyen bölümleri şu şekildedir:

Tezin ikinci bölümünde optimizasyon problemlerinin genel tanımı ve optimizasyon problemlerinin karmaşıklık sınıfları hakkında bilgi verilmiştir.

Üçüncü bölümünde temel çizge bilgileri ve tezin sonraki bölümlerinde kullanılacak olan bazı kavramlar verilmiştir.

Dördüncü bölümde minimum tepe örtüsü probleminin tanımı verilmiş, geçmiş çalışmalardan bahsedilmiş ve diğer çizge problemleri ile bağlantıları incelenmiştir.

Beşinci bölümde minimum tepe örtüsü problemi için yeni bir algoritma önerilmiş, önerilen algoritmanın performansı test edilmiş ve geçmiş çalışmalar ile kıyaslamalar yapılmıştır.

Altıncı bölümde genetik algoritmalar ve hibrid genetik algoritmalar incelenerek minimum tepe örtüsü problemine hibrid genetik algoritma ile çözüm önerisi getirilmiştir.

Yedinci bölümde tezde elde edilen sonuçlardan bahsedilmiştir.

Tezin son iki bölümünde ise kaynaklar dizini ve önerilen algoritmalardan ilkinin bilgisayar programının kaynak kodunun verildiği ek bulunmaktadır.

2. OPTİMİZASYON PROBLEMLERİ

Optimizasyon problemleri karar değişkenlerine bağlı olarak en uygun çözümün araştırıldığı problemlerdir. Optimizasyon problemlerinin amacı; amaç fonksiyonu olarak nitelendirilen fonksiyonun en küçüklenmesi ya da en büyüklenmesidir. Tepe örtüsü problemi göz önüne alındığında tepe örtüsü kümesinin eleman sayısının minimize edilmesi gibi amaçlar karar değişkenlerine bağlı fonksiyonlarla ifade edilir (Cura, 2008). Genel bir optimizasyon probleminin ifadesi aşağıdaki şekildedir:

Amaç fonksiyonu:

$$Enk \ (veya \ Enb) \ f(x) \tag{2.1}$$

Kısıtlar:

$$g_i(x) \geq b_i, \quad i = \overline{1,m} \tag{2.2}$$

$$h_i(x) \leq c_i, \quad i = \overline{1,n} \tag{2.3}$$

$$e_i(x) = d_i, \quad i = \overline{1,k} \tag{2.4}$$

$$x \in G \tag{2.5}$$

Bir sistemin davranışlarını yukarıdaki gibi ifadeler kullanarak tanımlayan modellere matematiksel model denir. Yukarıdaki modelde, (2.2, 2.3 ve 2.4) modelin kısıtlarını, (2.5) modelin karar değişkeni olan x'i, (2.1) ifadesi ise (2.2, 2.3 ve 2.4) kısıtları altında modelin amaç fonksiyonunu göstermektedir (Cura, 2008).

Bir optimizasyon probleminin amaç ve kısıt fonksiyonlarının tümü doğrusal fonksiyon ise bu probleme *doğrusal programlama problemi*, bu fonksiyonlardan herhangi biri doğrusal değil ise *doğrusal olmayan programlama problemi* denir. Karar değişkenleri negatif olmayan tamsayı değerler alan doğrusal programlama problemine *tamsayılı programlama problemi*, aksi halde *tamsayı olmayan programlama problemi* denir (Karaboğa, 2004).

2.1 Problemlerin Karmaşıklık Sınıfları

2.1.1 Algoritmik karmaşıklık

Bir algoritmanın *hesaplama karmaşıklığı*, algoritmanın ne kadar hızlı çalışacağı ve bilgisayarın belleğini ne kadar kullanacağı hakkında bilgi vermektedir. Bir algoritmanın hesaplama karmaşıklığı, hesaplamayı yapmak için gerekli zamanı ölçen *zaman karmaşıklığı* değerlendirilmesi ve hesaplama için gerekli bellek alanının ölçümü olarak *yersel karmaşıklık* olmak üzere iki farklı açıdan incelenmektedir. Genelde, zaman karmaşıklığı sadece karmaşıklık olarak adlandırılmaktadır (Nabiyev, 2009).

Kullanılan algoritmaların tipine göre tüm problemler eşdeğer değildir. Bir problemin algoritmik çözümlerinin sınıflandırılması, algoritmaların yürütülmesi ve gerekli işlemlerin sayısı temel alınarak gerçekleştirilmesi için gereken ölçüte *algoritmik karmaşıklık* denir (Nabiyev, 2009).

Algoritmaların çalışma zamanlarının belirlenmesinde sabit çalışma zamanlı emirler dikkate alınmamaktadır. Çalışma zamanının matematiksel özdeşliğinin yazılmasıyla bir algoritmanın göstereceği performans orantısal olarak tahmin edilebilir. Çalışma zamanının üst sınırı olan *büyük O* (Big O) notasyonu, algoritmanın en kötü durumdaki çalışma zamanının gösteriminde kullanılır ve büyük O notasyonu bir fonksiyonun asimptotik üst sınırını belirtir (Nabiyev, 2009).

2.1.2 P ve NP sınıflar

Bir algoritmanın *zaman karmaşıklığı*, belli boyutlardaki girdi değerleri için harcanan tüm hesaba dayalı adımların sayısını veren bir fonksiyondur. n girdi verisinin boyutunu belirtmek üzere, bir algoritmanın çalışma zamanı üstten herhangi bir $P(n)$ polinomu ile sınırlı ise buna *polinom sınırlı algoritma* denir. Bu tür algoritmalarla çözülebilen tüm problemler *P sınıfı* olarak tanımlanır (Nabiyev, 2005).

Bir problemin karmaşıklığı polinomiyal biçimde tanımlanabiliyorsa, bu problem sonlu sayıda adımla polinomiyal zamanda çözülebilecek bir problemdir. Polinomiyal algoritmaları çalıştıran sanal modeller *deterministik Turing makineleridir* (DTM) ve P sınıfı, DTM ile polinomiyal zamanda çözülebilen problemleri içermektedir. Polinomiyal zamanda deterministik olmayan Turing makineleriyle (NDTM) çözülebilen problemler ise *NP* (Non-deterministic Polinomial) *sınıfını* oluşturmaktadırlar. (Nabiyev, 2009; Garey and Johnson, 1979). P sınıf problemleri NP sınıfın bir alt kümesi olarak gösterilebilir, $P \subset NP$. Çünkü NDTM ile çözülebilen tüm problemler DTM ile çözebilir (Cormen et al., 2001).

Polinomiyal zamanda çözülebilen problemler P sınıfını oluşmaktadır. Bu tarz problemler, *k* bir sabit sayı ve *n* problem girdisinin boyutu olmak üzere $O(n^k)$ zamanda çözülebilen problemlerdir. Polinomiyal zamanda doğrulanabilen problemler ise NP sınıfını oluşturmaktadır. Yani, problemin herhangi bir çözümü verildiğinde, problemin girdi boyutuna bağlı olarak bu çözümün polinomiyal zamanda doğrulanabildiği problemler sınıfıdır. Bu sınıfın üyesi olan algoritmalara deterministik olmayan algoritma denir.

Bir Π probleminin *NP-tam* sınıfına dahil olabilmesi için, öncelikle bu Π probleminin NP sınıfına ait olduğu ve sonrasında da NP sınıftaki her problemin polinomiyal zamanda bu probleme indirgenebileceği gösterilmelidir (Cormen et al., 2001). Eğer NP sınıftaki her problem polinomiyal zamanda bir probleme indirgenebiliyorsa ancak bu problemin NP sınıfına ait olduğu gösterilemiyorsa bu tip problemler NP-zor problemler sınıfına girer.

3. ÇİZGELER İLE İLGİLİ TEMEL KAVRAMLAR VE TANIMLAR

Bu bölümde, çizge kuramındaki temel kavramlar verilmiştir. Bu bölümde 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.

Tanım 3.1: V tepeler kümesi $V \neq \varnothing$ ve $E \subseteq V \times V$ ayrıtlar kümesini oluşturmak üzere $G(V,E)$ ifadesine *çizge (graf)* denir. Çizgenin tepe sayısı $|V| = n$ ve ayrıt sayısı $|E| = m$ şeklinde gösterilir.

Tanım 3.2: Bir G çizgesi üzerindeki u ve v tepeleri, ayrıtlar kümesinde bulunan bir ayrıtla ilişkilendiriliyorsa bu tepeler *komşu tepe* adını alır ve $e = \{u,v\}$ şeklinde gösterilir. Diğer bir anlatımla e ayrıtı u ve v tepelerini birbirine bağlar veya u ve v tepeleri e kenarının uç tepeleridir denir.

Tanım 3.3: Bir G çizgesinde herhangi bir $v \in V$ tepesine bitişik ayrıtların sayısına *v tepesinin derecesi* denir ve $\deg(v)$ ile gösterilir. G çizgesinin en küçük tepe derecesi $\delta(G)$, en büyük tepe derecesi ise $\Delta(G)$ ile gösterilir.

Tanım 3.4: Bir tepeden farklı bir tepeye varışta yürüyüş üzerindeki her tepe bir kez kullanılıyorsa bu yürüyüşe *yol* denir.

Tanım 3.5: Bir tepeden farklı bir tepeye varışta kullanılan ayrıt sayısına *yolun uzunluğu* denir.

Tanım 3.6: Bir G çizgesinde u ve v gibi iki tepe arasındaki yollar içinde uzunluğu minimum olanın uzunluğuna; u ve v nin *uzaklığı* denir ve $d(u,v)$ biçiminde gösterilir.

Tanım 3.7: Bir G çizgesindeki başlangıç ve bitiş tepeleri aynı olan ayrıta *bukle* denir.

Tanım 3.8: Bir G çizgesinde, herhangi bir v tepesi herhangi bir ayrıt ile bitişik değilse, bu tepeye *izole tepe* denir.

Tanım 3.9: Tek bir tepe içeren, ayrıt içermeyen bir çizgeye *aşikar çizge* denir.

Tanım 3.10: Bir G çizgesinin herhangi iki tepesi arasında birden fazla ayrıt varsa bu ayrıtlara *çok katlı ayrıt*, bu tür çizgelere ise *çok katlı çizge* denir.

Tanım 3.11: Bir G çizgesinin tepelerini birleştiren bütün e_i ayrıtlarının bir w_i ağırlığıyla birebir ilişkilendirilmesiyle oluşan çizgeye *ağırlıklandırılmış çizge* denir. Ağırlıklandırılmamış bir çizgenin ayrıtlarının ağırlığı 1 olarak alındığında çizge, ağırlıklandırılmış bir çizge olarak düşünülür.

Tanım 3.12: Bir $G(V,E)$ çizgesinde $V^* \subseteq V$ ve $E^* \subseteq E$ olacak biçimde tanımlanan $H=(V^*,E^*)$ çizgesine G çizgesinin bir *alt çizgesi* denir ve $H \subseteq G$ biçiminde gösterilir.

Tanım 3.13: $G(V,E)$ çizgesi birbirine bağlantısı bulunmayan alt çizgelerden oluşuyorsa, bu alt çizgelerin her birine çizgenin bir *bileşeni* denir.

Tanım 3.14: n sayıda tepeye sahip bir G çizgesinin tüm tepeleri birbiri ile bitişik ise bu çizgeye *tam çizge* denir ve K_n ile gösterilir.

Tanım 3.15: Bir çizgedeki v_0 tepesinden v_n tepesine uzunluğu n olan bir yol, (n+1) tepeden ve n ayrıttan oluşan, $(v_0, e_1, v_1, ..., v_{n-1}, e_n, v_n)$ şeklinde ifade edilen bir dizidir. Başka bir anlatımla, uç tepeleri 1, iç tepeleri 2 dereceli olan çizgeye *yol çizge* denir. n tepeli yol çizge P_n ile gösterilir.

Tanım 3.16: Tüm tepelerinin derecesi 2 ve tepe sayısı $n \geq 3$ olan kapalı bir yürüyüş *çevre çizge* olarak adlandırılır. n tepeli bir çevre çizgenin ayrıt sayısı n tane olup, bu çizge C_n ile gösterilir.

Tanım 3.17: *n* tepeli bir çizgenin bir tepesi *n*-1, diğer tüm tepeleri 1 dereceli ise bu çizgeye *yıldız çizge* denir ve $K_{1,n-1}$ ile gösterilir.

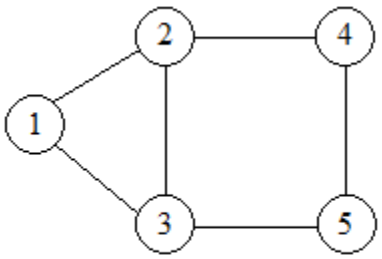
Tanım 3.18: *n* tepeli bir çevre çizgenin her bir tepesine, bu *n* tepeden farklı bir tek tepeden birer ayrıt eklenmesiyle elde edilen çizgeye *tekerlek çizge* denir ve bu çizge $W_{1,n}$ ile gösterilir. Tekerlek çizgeyi $K_1 + C_n$ şeklinde de ifade etmek mümkündür.

Tanım 3.19: Bir $G(V,E)$ çizgesi verildiğinde, $V_1 \cap V_2 = \phi$, $V_1 \cup V_2 = V$ olacak şekilde V kümesinin iki alt kümesi var ve E kümesindeki her bir ayrıt, V_1 kümesindeki bir tepe ile V_2 kümesindeki bir tepeyi birleştiriyorsa bu tür çizgelere *iki parçalı çizge* denir.

Tanım 3.20: G çizgesi, n tepesi olan basit bir çizge olsun. G çizgesinin *tümleyeni* olan $\overline{G}$ çizgesi, K_n tam çizgesinden G çizgesinin ayrıtlarının silinmesiyle elde edilir.

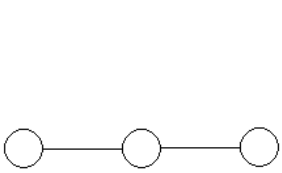
Tanım 3.21: $G(V,E)$ çizgesinde, $V = \{v_1, v_2, ..., v_n\}$ tepe kümesi E ayrıtlar kümesi olmak üzere, $A = \left[a_{ij}\right]_{nxn}$ matrisine *komşuluk matris* denir.

$$a_{ij} = \begin{cases} 1, & \text{eğer } v_i \text{ tepesi } v_j \text{ tepesine bitişik ise} \\ 0, & \text{aksi halde ya da i=j olduğunda} \end{cases}$$

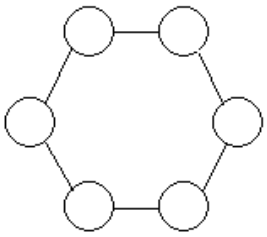


$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

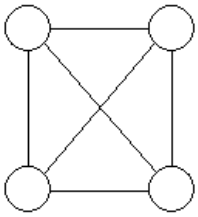
Şekil 3.1 Bir G çizgesi ve komşuluk matrisi



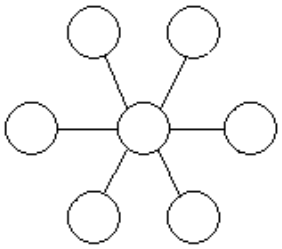
P_3 Yol Çizge



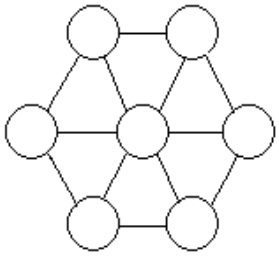
C_6 Çevre Çizge



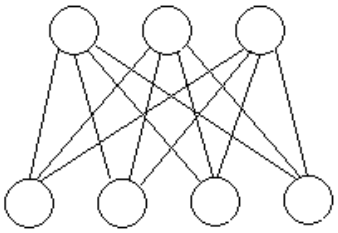
K_4 Tam Çizge



$K_{1,6}$ Yıldız Çizge



$W_{1,6}$ Tekerlerk Çizge



$K_{3,4}$ İki Parçalı Çizge

Şekil 3.2 Özel çizgeler

Tanım 3.22: Birleştirilmiş bir $G(V,E)$ çizgesini birleştirilmemiş bir çizge ya da sadece izole tepelerden oluşan bir çizge haline getirmek için çizgeden çıkarılması gereken en az tepe sayısına G çizgesinin *tepe birleştirilmişlik sayısı* adı verilir ve $\kappa(G)$ ile gösterilir.

Tanım 3.23: Silindiklerinde G çizgesini bağlantısız yapan tepelerin oluşturduğu kümeye *G 'nin ayıran kümesi* ya da *tepe kesim kümesi* denir.

Tanım 3.24: G çizgesinin ayrıt sayısının, G çizgesinin sahip olabileceği en büyük ayrıt sayına oranına *çizgenin yoğunluğu* denir.

4. MİNİMUM TEPE ÖRTÜSÜ PROBLEMİ VE İLİŞKİLİ OLDUĞU PROBLEMLER

4.1 Minimum Tepe Örtüsü Problemi

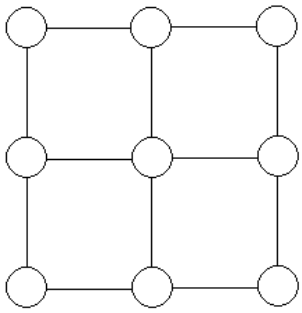
Minimum tepe örtüsü problemi bilgisayar bilimlerinde çok önemli bir yere sahip olan bir çizge problemidir. 1972 yılında Karp problemin NP-tam sınıfına ait olduğunu kanıtlamıştır (Karp, 1972). Problem NP-tam sınıfına ait olduğundan kesin çözümü için polinom zamanlı algoritmalar bulunamamaktadır. Bu tarz problemlere genel olarak iki farklı çözüm yaklaşımı vardır. Birinci yaklaşım küçük örnekler için üssel zamanlı optimal çözüm veren algoritmalar, ikinci yaklaşım ise optimale yakın çözümler bulan polinom zamanlı algoritmalardır (Kumar, 2009).

Problem, çözümünün çok zor olmasına rağmen geniş uygulama alanından dolayı araştırmacıların ilgisini çekmektedir. Bilgisayar ağları, çizelgeleme ve bioinformatik problemin uygulama alanlarına örnek olarak gösterilebilir. Tepe örtüsü problemi gerçek hayattaki uygulamalarının yanı sıra karmaşıklık teorisinde diğer problemlerin NP-tamlığını kanıtlamak için de kullanılır.

Tanım 4.1: $G(V,E)$ bir bağlantılı çizge, V tepeler kümesi ve E ayrıtlar kümesi olsun. Her bir (u,v) ayrıtı için $u \in V^*$ veya $v \in V^*$ (ya da her ikisi) koşulunu sağlayan V 'nin bir alt kümesi olan V^* kümesine *tepe örtüsü* denir. Minimum tepe örtüsünün eleman sayısı $\beta(G)$ ile gösterilir.

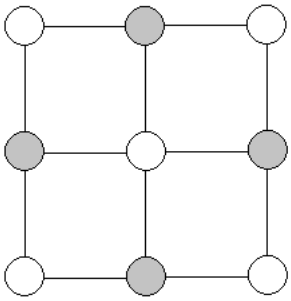
Tepe örtüsünün büyüklüğü örtü kümesinin içindeki tepe sayısı kadardır. Minimum tepe örtüsü problemi bu kümelerden en küçüğünün bulunmasıdır. Minimum tepe örtüsü probleminin iki versiyonu vardır: tanımlama versiyonu ve optimizasyon versiyonu. Tanımlama versiyonunda verilen boyuttan çok olmayan bir tepe örtüsünün olup olmadığı araştırılır, optimizasyon versiyonunda ise çizgenin minimum tepe örtüsü bulunmaya çalışılır.

Örneğin, iletilen paketlerin istatistiklerini tutarak bilgisayar ağının performansı arttırmak istiyoruz. Bilgisayar ağını, tepeler bilgisayarları, ayrıtlar ise aralarındaki bağları simgeleyecek şekilde modelleyebiliriz.



Şekil 4.1 Bilgisayar ağının çizge modeli

Bilgisayar ağlarında veri toplamak ağın performansını düşürerek ağdaki veri iletim hızını düşürür. Bu yüzden verilerin toplanması için en az sayıda tepe seçilmesi gerekmektedir. Bu problemin giderilmesi için çizgenin minimum tepe örtüsü bulunmalıdır. Bu şekilde bilgisayar ağındaki iletilen tüm paketler en az sayıda bilgisayar kullanılarak toplanabilir.



Şekil 4.2 Bilgisayar ağının tepe örtüsü çözümü

Minimum tepe örtüsü problemi ile günlük hayatta ve akademik çalışmalarda yoğun bir şekilde karşılaşırız. Problem en çok bilgisayar ağlarında ve bioinformatik biliminde karşımıza çıkmaktadır. Aşağıdaki problemler bu alanlar için birer örnek olarak verilebilir.

Haberleşme problemi, tüm servis alanına erişilmesini sağlamak için minimal sensör kümesi seçmektir. Sensörlerin minimal kümesini bulmak tepe örtüsü problemi olarak modellenebilir. Burada tepe örtüsü kümesi tüm sensörler arasında haberleşmeye olanak sağlanacak şekilde minimum sayıda sensörlerin kaynak olarak seçilmesiyle oluşturulur (Safar, 2007).

Biyolojide protein yapılarının hizalanması proteinlerin birbirlerine ne kadar benzediği sorusunun cevaplanmasını sağlar. Proteinlerin yapılarında güçlü fiziksel benzerlikler vardır ve bu benzerliklerin anlaşılması protein temelli ilaçlar geliştirilmesi için kilit bir rol oynamaktadır. Protein yapılarının hizalanması için kullanılan modellerden biri maksimum temas harita örtüşmesidir (maximum contact map overlap). Bu model uygun bir çizge üzerinde minimum tepe örtüsü probleminin bulunmasıyla çözülebilecek maksimum klik problemine indirgenebilir (Strickland et al, 2005).

4.2 Minimum Tepe Örtüsü Probleminin Matematiksel Modeli

Minimum tepe örtüsü problemi için karar değişkenleri $\forall v \in V$ için *v.* tepenin çözümde olup olmayacağına bağlı olarak aşağıdaki şekilde 0-1 türünden tanımlanabilir.

$$x_v = \begin{cases} 1, & v. \text{ tepe minimum tepe örtüsü içindeyse} \\ 0, & \text{aksi halde} \end{cases}$$

Bu durumda 0-1 minimum tepe örtüsü probleminde amaç fonksiyonu ve kısıtlar aşağıdaki şekildedir:

Amaç fonksiyonu:

$$\text{Enk Z} = \sum_{v \in V} x_v \qquad (4.2.1)$$

Kısıtlar:

$$x_u + x_v \geq 1 \quad \forall (u,v) \in E \qquad (4.2.2)$$

$$x_v = \{0,1\} \qquad \forall v \in V \tag{4.2.3}$$

Burada (4.2.2) kısıtı ile birbirine komşu olan herhangi iki tepeden en az birinin örtü kümesinde olması sağlanmaktadır.

4.3 Minimum Tepe Örtüsü Probleminin NP-Tamlığı

Bu bölümdeki teorem ve ispat Garey and Johnson'ın (1979) "Computer and Intractability" isimli kitabından alınmıştır.

Teorem 4.3.1: TEPE ÖRTÜSÜ problemi NP-tam'dır.

İspat: Deterministik olmayan bir algoritma tarafından çizgenin tepeler kümesinin sadece bir alt kümesinin seçileceği ve bu alt kümenin istenilen boyutta uygun bir çözüm olup olmadığının polinomiyal bir zamanda kontrol edilebileceği için, tepe örtüsü probleminin NP sınıfına ait olduğunu anlamak kolaydır.

Problemin NP-tam sınıfına ait olduğunu göstermek için, NP-tam bir problemden polinomiyal zamanda TEPE ÖRTÜ problemine bir dönüşüm bulunması gerekmektedir. Bu işlem için kullanılacak olan 3-GTP (3-lü GERÇEKLEŞTİREBİLİRLİK Tanıma Problemi) (3-Satisfiability) probleminin *tanıma* versiyonu aşağıdaki gibidir.

Koşul: $C = \{c_1, c_2, ..., c_m\}$ sonlu bir U kümesi üzerinde tüm $1 \le i \le m$ için $|c_i| = 3$ koşulunu sağlayan cümleciklerden (clauses) oluşan değişkenleri bir koleksiyon (collection) olsun.

Soru: U için öyle bir doğru (truth) atama (assignment) var mıdır ki C'deki tüm cümlecikleri gerçekleştirilebilir olsun?

TEPE ÖRTÜSÜ probleminin *tanıma* versiyonu da aşağıda verilmiştir.

Koşul: $G(V,E)$ bir çizge ve $K \le |V|$ bir pozitif tam sayı olsun.

Soru: G çizgesi için K dan büyük olmayan boyutta bir tepe örtüsü var mıdır? Başka bir değişle, her bir $\{u,v\} \in$ E ayrıtı için, u ve v den en az birini içeren, $|V'| \leq K$ olacak şekilde bir $V' \subseteq V$ tepe örtüsü var mıdır?

3-GTP probleminin tepe örtüsü problemine dönüştürülebilmesi için gereken yapı birçok bileşenden oluşturulacaktır. Bu yapıyı oluşturmak için doğru-seçim (truth-setting) bileşenleri, gerçekleştirilebilirlik testi (satisfaction testing) eden bileşen ve bu bileşenleri birbirine bağlamak için kullanılan iletişim (commumication) ayrıtları kullanılması gerekmektedir.

Her bir $u_i \in U$ değişkeni için, $Ti = (Vi, E_i)$ $Vi = \{u_i, \overline{u_i}\}$ $E_i = \{\{u_i, \overline{u_i}\}\}$ olacak şekilde iki tepe ve onları bağlayan bir ayrıttan oluşan bir doğru seçim bileşeni olacaktır. Herhangi bir tepe örtüsü u_i ve $\overline{u_i}$ den en az birini içermelidir.

Her bir $c_j \in C$ cümleciği için, 3 tepe ve 2 ayrıttan oluşan bir $S_j = (V'_j, E'_j)$ gerçekleştirilebilirlik testi bileşeni olacaktır.

$$V'_j = \{a_1[j], a_2[j], a_3[j]\}$$

$$E'_j = \{\{a_1[j], a_2[j]\}, \{a_1[j], a_3[j]\}, \{a_2[j], a_3[j]\}\}$$

Herhangi bir tepe örtüsü bu bileşendeki ayrıtların örtülmesi için bu bileşendeki tepelerin en az ikisini içermelidir.

Yapının oluşturulmasındaki en önemli kısım koleksiyondaki cümleciklerin iletişim ayrıtlarının oluşturulmasıdır. Her bir $c_j \in C$ cümleciği için, c_j deki üç literal x_j, y_j ve z_j olarak tanımlansın. O halde iletişim ayrıtları aşağıdaki gibi tanımlanabilir:

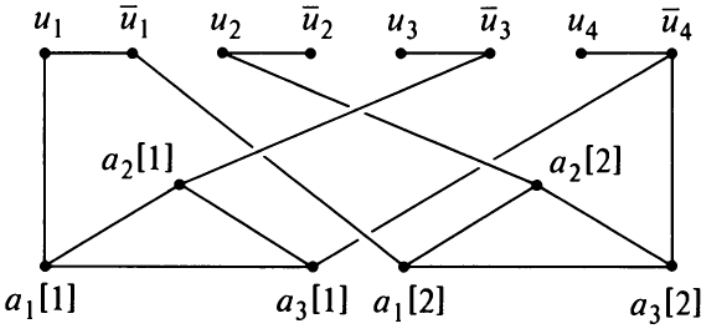
$$E''_j = \{\{a_1[j], x_j\}, \{a_2[j], y_j\}, \{a_3[j], z_j\}\}$$

Tepe örtüsü için oluşturulan yapı K ya $n + 2m$ değerinin atamasıyla ve $G(V,E)$ çizgesinin aşağıdaki gibi inşa edilmesiyle tamamlanır:

$$V = (\bigcup_{i=1}^{n} V_i) \cup (\bigcup_{j=1}^{m} V'_j)$$

$$E = (\bigcup_{i=1}^{n} E_i) \cup (\bigcup_{j=1}^{m} E'_j) \cup (\bigcup_{j=1}^{m} E''_j)$$

Şekil 4.3'de, $U = \{u_1, u_2, u_3, u_4\}$ değişkenleri ve $C = \{\{u_1, \overline{u_3}, \overline{u_4}\}, \{\overline{u_1}, u_2, \overline{u_4}\}\}$ koleksiyonundan elde edilen çizge örneği verilmiştir.



Şekil 4.3 3-GTP ile elde edilen tepe örtüsü örneği, $K = n + 2m = 8$.

Burada yapının polinomiyal zamanda oluşturulduğu açıktır. İspatın tamamlanması için gereken tek koşul "C ancak ve ancak G çizgesi K dan büyük olmayan boyutta bir tepe örtüsüne sahipse gerçekleştirilebilir" olduğunu göstermektir.

İlk olarak, $V' \subseteq V$ nin G çizgesi için $|V'| \leq K$ koşulunu sağlayan bir tepe örtüsü olduğunu varsayalım. Daha önce belirttiğimiz gibi V' her bir T_i den en az bir, her bir S_j den ise en az iki tepe içermelidir. Buradan tepe örtüsünün toplamda en az $K = n + 2m$ tepe içerir, çünkü V' her bir T_i den kesin olarak bir ve her bir S_j den ise kesin olarak iki tepe içermelidir. Elde edilen V' her bir doğru-seçim bileşeninden doğru atamalar (truth assignment) elde etmek için kullanılabilir, $t : U \to \{T, F\}$. Burada kullanılan tek bağlantılar aşağıdaki gibidir:

$$t(u_i) = \begin{cases} T & u_i \in V' \text{ ise} \\ F & \overline{u_i} \in V' \text{ ise} \end{cases}$$

Bu doğruluk atamasının her bir $c_j \in C$ için gerçekleştirebileceğini göstermek için $E"_j$ deki üç ayrıtı ele almalıyız. Bu ayrıtların sadece ikisi $V'_j \cap V"$ tarafından örtülebilir, bu yüzden bu ayrıtlardan biri V' deki V_i lerden biri ile örtülecektir. Bu da ilgili literalin, u_i ya da $\overline{u_i}$ olduğunu göstermektedir ve böylece t doğruluk ataması altında c_j doğrudur, yani c_j, t tarafından gerçekleştirilebilirdir. Bu durum tüm c_j ler için geçerli olduğundan t, C için gerçekleştirilebilir bir doğru atamadır.

Tersini düşünecek olursak t, C için gerçekleştirilebilir bir doğru atama olsun. Bu atamaya uygun tepe örtüsü V', her bir T_i den bir ve her bir S_j den ise iki tepe içersin. Öyle ki T_i nin V' deki elemanı eğer $t(u_i) = T$ ise u_i, eğer $t(u_i) = F$ ise $\overline{u_i}$ olsun. t tüm c_j leri gerçekleştirebildiği için $E"_j$ deki üç ayrıttan en az birinin örtüleceğini garanti eder. Bu nedenle, S_j alt çizgesinde kalan iki ayrıtın kalan ayrıtın bitiş noktasını V' ye sokmamız yeterli olacaktır ve bu da bize istenilen tepe örtüsünü vermektedir.

■

4.4 Minimum Tepe Örtüsü Problemi İçin Çözüm Yöntemleri

Optimizasyon problemlerinin çözüm yöntemleri kesin ve yaklaşık çözüm olmak üzere ikiye ayrılmaktadır. Kesin yöntemler optimal çözümü garanti eden ancak çalışma zamanı uzun süreler alan yöntemlerdir. Kısa sürelerden optimal olmayan kaliteli çözümler bulabilmek içinse yaklaşık yöntemler kullanılabilir. Yaklaşık çözümler optimal çözümleri garanti etmese de kısa sürelerde optimale yakın çözümler üretebilirler.

4.4.1 Kesin yöntemler

Temel NP-tam problemler için düşük üssel zamanlı algoritmalar bulmak birçok araştırmacının ilgisini çekmiştir. Bu algoritmalara yapılan küçük katkılar bile büyük önem taşımaktadır. Minimum tepe örtüsü problemi için de literatürde

birçok kesin algoritma önerilmiştir. Bunların çoğu sabit parametre (fixed parameter) algoritmalarıdır.

Parametre karmaşıklığı bilgisayar bilimlerinde hesaplama karmaşıklığının bir dalıdır. Parametre karmaşıklığı problemleri giriş parametrelerine göre sınıflara ayırmaktadır. Bu NP-zor problemlerin daha iyi bir şekilde sınıflandırılmasını sağlar. Bu alandaki ilk kitap Donwey ve Fellows (1999) tarafından yayınlanmıştır.

NP-tam problemler için geliştirilen kesin algoritmalar problemin boyutuna göre üssel olarak artmaktadır. Ancak parametreleri sabit olan bazı problemler için problemin boyutuna göre polinomiyal, problemin parametresine göre üssel algoritmalar geliştirilebilir. Bu tarz algoritmalara *sabit parametreli ulaşılabilir* (fixed parameter tractable) algoritmalar (ftp) denir. Küçük parametreli örneklerin çözümü için bu algoritmalar oldukça efektiftir. Bu algoritmaların en iyi sonuç verdiği problemlerden biri minimum tepe örtüsü problemidir.

Minimum tepe örtüsü problemi için (k minimum tepe örtüsünün boyutu olmak üzere) $O(kn + 2^k k^{2k+2})$ çalışma zamanına sahip ilk ftp algoritması Buss ve Goldsmith (1994) tarafından geliştirilmiştir, daha sonra Downey ve Fellows (1999) algoritmanın çalışma zamanını $O(kn + 2^k k^{2k})$ ya indirerek algoritmayı geliştirmiştir. Bu algoritmalarda 2 sınır değerini geçebilen ilk algoritma Balasubramanian ve diğerleri (1998) tarafından önerilen $O(kn + 1.324718^k k^k)$ çalışma zamanlı algoritmadır. Daha sonra bu algoritma Downey ve diğerleri (1999) tarafından $O(kn + 1.31951^k k^2)$ çalışma zamanına çekilerek geliştirilmiştir. Stege ve Fellows (1999), $O(kn + \max\{1.25542^k k^2, 1.2906^k k\})$ çalışma zamanlı bir ftp algoritması geliştirmiş ve bu alanda büyük bir iyileştirme yapmıştır. Niedermeier ve Rossmanith (2000) yukarıdaki algoritmaların k çarpımlarını ortadan kaldıracak genel bir yöntem geliştirmiştir. Örnek olarak bu teknik Stege ve Fellows un algoritmasının çalışma zamanını $O(kn + 1.2906^k)$ a indirgemiştir. Bu alanda bilinen en iyi süre üst sınırlı algoritmalardan biri Chen ve diğerleri (2001) tarafından yapılan $O(kn + 1.2852^k)$ çalışma zamanına sahip algoritmadır (Chen et al, 2001).

Minimum tepe örtüsü problemi için kullanılan kesin algoritmaların çoğu *dal-sınır* yöntemini kullanmaktadır. Dal-sınır algoritması optimal çözümü elde etmek için sitemli bir şekilde uygun çözümleri sayımlamaya dayanan bir yöntemdir. Ancak bu sayımlama işlemi tüm olasılıklar için değil sadece ümit veren olasılıklar için geçerlidir. Bu yüzden bu yöntem hızlı bir şekilde optimal çözüme ulaşmaktadır. Minimum tepe örtüsü problemi için geliştirilen birçok ftp algoritması bu yöntemi kullanmaktadır.

Minimum tepe örtüsü problemi için geliştirilen kesin algoritmaların bir kısmı da *dinamik programlama* tekniğini kullanmaktadır. Dinamik programlama optimumluk ilkesine dayalı çözümleme yöntemlerine verilen isimdir. Bu yöntem problemi çözmek için problemi alt problemlere ayırır. Başlangıçta alt problemlerin çözümü bulunur, daha sonra daha büyük alt problemlerin çözümü araştırılır ve en sonunda problemin kendisi çözülmüş olur. Minimum tepe örtüsü için kullanılan dinamik programla yöntemi genel olarak *ezberleme* (memorisation) (Robson, 1972) tekniğidir. Bu teknikle beraber üssel zamanlı karmaşıklığı ile birçok üssel zamanlı rekürsif algoritmanın zaman karmaşıklığı düşürülmüştür. Ezberleme tekniğinin arkasındaki en önemli fikir, eğer bir alt problem ile birçok defa karşılaşılıyorsa aynı hesaplamayı tekrar tekrar yapmamak için bu çözümlerin uygun şekillerde saklanmasıdır. Niedermeier and Rossmanith (2003) bu yöntemi kullanarak polinomiyal yer karmaşıklığını üssel yer karmaşıklığa çevirip küçük k değerleri için bilinen en hızlı algoritmayı geliştirmiştir. Ayrıca ezberleme tekniği ftp algoritmalarına uygulanarak bu algoritmaların çalışma zamanlarını düşürebilmektedir.

4.4.2 Yaklaşık yöntemler

Yaklaşık çözümler kesin algoritmalar gibi optimal çözüm garanti etmezler. Ancak pratikte, uzun zamanda kesin çözüm bulmak yerine kısa zamanda yaklaşık çözümler bulmak daha değerlidir.

Yaklaşık yöntemler içinde en çok kullanılan yöntemlerin başında *sezgisel* (heuristic) algoritmalar gelmektedir. Sezgisel algoritmalar, ele alınan problemin özelliğine göre tasarlanırlar. Genel olarak optimal çözümlere yaklaşarak oldukça

hızlı bir şekilde çözümler bulurlar. Sezgisel algoritmalar en kötü durum düşünüldüğünde optimal çözümden çok uzak sonuçlar bulabilir ve ya üssel çalışma zamanı gerektirebilir ancak iyi sezgisel algoritmalar en iyi yaklaşım algoritmaların performanslarını da geçebilmektedir (Elmas, 2007).

Minimum tepe örtüsü problemi için en çok bilinen sezgisellerden biri *açgözlü* (greedy) algoritmasıdır. Açgözlü algoritma tüm ayrıtlar silinene kadar çizgedeki en büyük dereceye sahip tepeyi bulup, o tepeyi çözüm kümesine atıp bitişik olduğu tüm ayrıtları silmektedir. Algoritmanın problemler özelinde birçok modifikasyonu bulunmaktadır (Clarkson, 1985). Minimum tepe örtüsü için kullanılan alternatif bir sezgisel ise, çizgede ayrıt kaldığı sürece, keyfi bir ayrıt seçerek bağlı olduğu tepeleri çözüm kümesine atmaktır. Bu yöntem ile her bir ayrıtın en az bir tepe tarafından örtülmesi garanti edilmektedir (Evans, 1998).

Çok büyük örnekler için sezgisel algoritmalar tatminkar sonuçlar bulamayabilir. Böyle durumlarda, algoritmanın en kötü durumda ne kadar yanılabileceğini bilmek çok önemlidir. En kötü durumda bile ne kadar yanılabileceği, yani performans garanti değeri bilinen sezgisel algoritmalara *yaklaşım algoritmaları* (approximation algorithm) denir. Genel olarak yaklaşım algoritmaları "α-yaklaşım algoritması" şeklinde gösterilir. Minimizasyon problemleri için α değeri 1 den büyük, maksimizasyon problemleri için ise α değeri 1 den küçüktür. Aşağıda birbirinden bağımsız bir şekilde Fanica Gavril ve Mihalis Yannakakis tarafından minimum tepe örtüsü problemi için önerilmiş bir yaklaşım algoritma verilmiştir.

Minimum Tepe Örtüsü İçin Bir Yaklaşım Algoritması:

1. Keyfi bir ayrıt seç.
2. Ayrıtın uç tepelerinin bağlı olduğu ayrıtları çizgeden sil ve bu tepeleri çözüm kümesine ekle.
3. Çizgede hala ayrıt varsa Adım 1'ye dön.

Seçilen her bir ayrıt en az bir bağlı olduğu tepe tarafından örtülmektedir bu yüzden elde edilen sonuç minimum tepe örtüsünün optimal çözümün en fazla iki katı

büyüklüğünde olabileceğini görmek açıktır. Bu bilgiler algoritmanın 2-yaklaşım algoritması olduğunu göstermektedir.

$P \neq NP$ olduğu sürece, minimum tepe örtüsü problemi için 1.1666 dan düşük bir yaklaşım oranı bulunamamaktadır (Hastad, 1997). Ayrıca Dinur ve Safra (2005) minimum tepe örtüsü problemi için 1.3606 yaklaşım oranına sahip olan bir algoritmanın geliştirilebileceğini ispatlamıştır.

Literatürde minimum tepe örtüsü için birçok yaklaşım algoritması önerilmiştir (Pitt (1985), Halperin (2000), Gandhi ve Halperin (2003)) .Problem için literatürdeki en iyi yaklaşım algoritmaları Monien ve Speckenmeyer (1985) ve Bar-Yehuda ve Even (1985) tarafından önerilmiştir. Bu algoritmalar $2-\frac{\log\log|V|}{2\log|V|}$ yaklaşım oranına sahiptir. Daha sonra Karakostas (2009) bu yaklaşım oranını $2-\frac{1}{\sqrt{\log|V|}}$ e indirgenmiştir.

NP-tam problemler için kullanılan yaklaşık yöntemlerden biri de *yerel aramadır* (local search). Yerel arama optimale yakın çözümler vermek için yaygın olarak kullanılan bir yöntemdir. Yerel arama yönteminin çalışma prensibi; uygun bir çözümden başlayıp, akılcı hamleler ile çözümün komşu çözümlerine ulaşarak yerel optimal çözüm bulmasıdır. Yerel aramalarının komşuluk sınırları genişletilirse daha iyi sonuçlar elde edilebilir ancak çalışma zamanı da komşulukların genişletildiği ölçüde artar (Katayama et al, 2005). Literatürde minimum tepe örtüsü problemi için birçok yerel arama önerilmiştir. Bunlara örnek olarak; Andrade (2008), Richter ve diğerleri (2007), Pullan ve Hoos (2006), Katayama ve diğerleri (2005) ve Battiti ve Protasi (2001) verilebilir.

NP-tam problemler için kullanılan yaklaşık yöntemlerden bir diğeri ise *evrimsel algoritmalardır* (evolutinary algorithms). Evrimsel algoritmalar *sezgiüstü* (metaheuristic) algoritmalar olarak bilinirler. Evrimsel algoritmalar tek bir çözümle değil, çözümlerin oluşturduğu bir popülasyon ile ilgilenir. Popülasyon çeşitli operatörler vasıtasıyla değişime uğrayarak yeni çözümler oluşturur. Evrimsel algoritmaların en çok kullanılan alt dallarından biri genetik algoritmalardır. Genetik algoritmalara tezin 6. Bölümde detaylı bir şekilde

değinilecektir. Minimum tepe örtüsü problemi içinde evrimsel algoritmalar ile oldukça iyi sonuçlar bulunmaktadır. Literatürde bu alandaki çalışmalara örnek olarak Khuri ve Thomas (1994), Aggarwal ve diğerleri (1997), Evans (1998), Xu ve Ma (2006) ve Oliveto ve Yao (2007) verilebilir.

4.5 Minimum Tepe Örtüsü Probleminin İlişkili Olduğu Problemler

Minimum tepe örtüsü problemi iki önemli çizge problemiyle yakından ilişkilidir. Bir çizgenin minimum tepe örtüsünü bulmak aynı zamanda çizgenin maksimum bağımsız kümesinin ve çizgenin en büyük kliğinin bulunmasını sağlar.

Tanım 4.2: $G(V,E)$ bir bağlantılı çizge, V tepeler kümesi ve E ayrıtlar kümesi olsun. Her bir $u,v \in V^*$ ikilisi için $(u,v) \notin E$ koşulunu sağlayan V 'nin bir alt kümesi olan V^* kümesine *bağımsız küme (independent set)* denir. Maksimum bağımsız kümenin eleman sayısı $\alpha(G)$ ile gösterilir. Maksimum bağımsız küme problemi bu kümelerinin en büyük boyutlusunun bulunmasıdır.

Tanım 4.3: $G(V,E)$ bir bağlantılı çizge, V tepeler kümesi ve E ayrıtlar kümesi olsun. Her bir $u,v \in V^*$ ikilisi için $(u,v) \in E$ koşulunu sağlayan V 'nin bir alt kümesi olan V^* kümesine *klik* (clique) denir. Maksimum klik'in eleman sayısı $\omega(G)$ ile gösterilir. Bir başka deyişle, bir klik G çizgesinin bir tam alt çizgesidir.

Maksimum bağımsız küme, minimum tepe örtüsü ve maksimum klik arasında aşağıdaki bağlantı açıktır (Garey and Johnson, 1979).

Lemma 4.1: $G(V,E)$ bir bağlantısız çizge, $V^* \subseteq V$ olsun. Aşağıdaki ifadeler birbirine denktir:

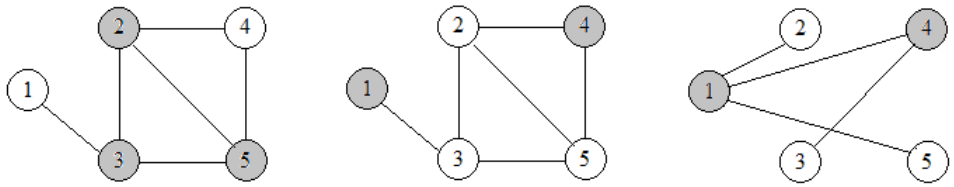
V^*, G için bir tepe örtüsüdür.

$V - V^*, G$ için bir bağımsız kümedir.

$V - V^*, G$ in tümleyeni olan $\overline{G}$ çizgesinde bir kliktir.

Sonuç olarak minimum tepe örtüsü probleminin çözümü maksimum bağımsız kümenin tümleyenini alarak elde edilebilir. Maksimum klik probleminin çözümü ise G çizgesinin tümleyeni alınarak maksimum bağımsız kümesi bulunarak elde edilebilir. Bu üç problem arasındaki ilişki aşağıdaki gibi özetlenebilir.

$$\begin{aligned} &\beta(G)+\alpha(G)=n \\ &\beta(G)+\omega(\overline{G})=n \\ &\alpha(G)=\omega(\overline{G}) \end{aligned} \qquad (4.4.1)$$



(a) (b) (c)

Şekil 4.4 Tepe örtüsü probleminin diğer çizge problemleriyle olan ilişkisi (a) G çizgesinin minimum tepe örtüsü (b) G çizgesinin maksimum bağımsız kümesi (c) G çizgesinin tümleyenindeki maksimum klik

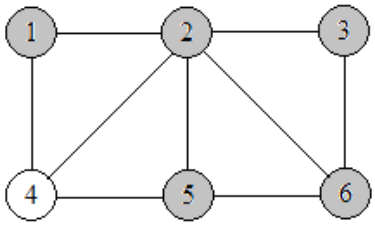
5. MİNİMUM TEPE ÖRTÜSÜ PROBLEMİ İÇİN YENİ BİR ALGORİTMA

Çalışmanın bu bölümünde minimum tepe örtüsü probleminin çözümü için bir algoritma önerilmiş ve algoritmanın karmaşıklığı incelenmiştir.

5.1 Parçalama Algoritması

Algoritmanın anlatımına geçmeden önce algoritmada kullanılan birkaç tekniğin tanımı verilecektir.

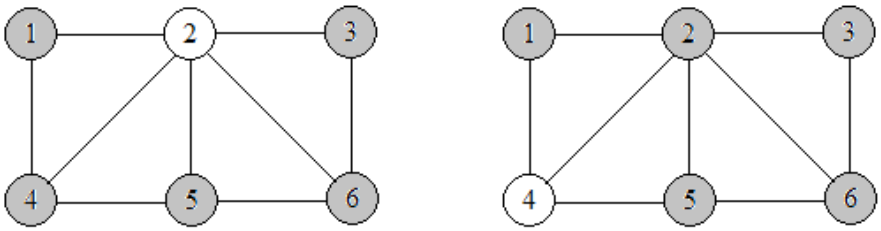
Tanım 5.1: Eğer tepe örtüsü kümesinde bulunan bir tepenin tüm komşuları da tepe örtüsü kümesindeyse, bu tepe çözüm kümesinden silinir. Bu tip tepelere *silinebilir tepeler* denir.



Şekil 5.1 Silinebilir tepe içeren çizge örneği

Şekil 5.1'deki 6 numaralı tepenin tüm komşu tepeleri çözüm kümesi içinde olduğundan 6 numaralı tepe silinebilir bir tepedir. 6 numaralı tepe çözüm kümesinden çıkarıldıktan sonra çizge için minimum tepe örtüsü elde edilmiş olur.

Tanım 5.2: Tepe örtüsü kümesinde bulunan bir tepenin sadece bir komşusu hariç tüm komşuları tepe örtüsü kümesindeyse, bu tepe çözüm kümesinden çıkarılıp çözüm kümesinde olmayan komşusu çözüm kümesine eklenebilir. Bu tip tepelere *kaydırılabilir tepeler* denir.



(a) (b)

Şekil 5.2 Kaydırılabilir tepe içeren çizge örneği (a) G_1 çizgesi (b) G_2 çizgesi

Şekil 5.2'de G_1 çizgesinde silinebilir tepe bulunmamaktadır. 4 numaralı tepenin 2 numaralı tepe hariç tüm komşuları çözüm kümesindedir. Bu durumda 4 numaralı tepe çözüm kümesinden çıkarılıp 2 numaralı tepe çözüm kümesine eklenerek G_2 çizgesi elde edilir. Yeni oluşan G_2 çizgesinde 6 numaralı tepenin tüm komşu tepeleri çözüm kümesinde olduğu için bu tepe çözüm kümesinden çıkarılır. 6 numaralı tepe çözüm kümesinden çıkarıldıktan sonra G çizgesi için yine minimum tepe örtüsü elde edilmiş olur.

Önerilen algoritma (Ugurlu, 2012a) çizgedeki minimum dereceye sahip tepeyi bulup tepeyi izole tepe haline getirir ve bu tepenin tüm komşularını çözüm kümesine ekler. Burada amaç çizgeyi parçalayarak daha küçük çizge parçaları elde etmektir. Bu işlem çizgedeki tüm tepeler izole tepe olana kadar çizgenin tüm bileşenleri için devam etmektedir. Elde edilen çözüm kümesi minimum tepe örtüsü problemi için bir başlangıç çözümü oluşturur. Başlangıç çözümü elde edildikten sonra çözüm iyileştirilmeye çalışılır. Çözüm kümesinde önce silinebilir tepe olup olmadığı araştırılır, eğer bulunamazsa kaydırılabilir tepeler yardımıyla silinebilir tepeler yaratılmaya çalışılır. Çözüm üzerinde bu iki optimizasyon yapılıp çözüm kümesinde silinebilir ve kaydırılabilir tepe kalmayınca, çözüm kümesinin maksimum bağımsız küme ve minimum tepe örtüsü elemanları birbirleriyle değişince çözüm kümesinin uygunluğunu bozmayan bir alt kümesi bulunur. Bulunan alt kümenin elemanları birbiriyle değiştirildikten sonra tekrar silinebilir ve kaydırılabilir tepeler araştırılır.

Ashay Dharwadker'ın Ekim 2011 de yayınladığı kitapta silinebilir ve kaydırabilir tepe kavramlarını kullanmıştır. Dharwadker önerdiği algoritmanın tüm bilinen örnekler üzerinde optimal sonuç bulduğunu iddia etmektedir. Dharwadker'ın algoritması tepe sayısı ve tüm tepelerin ikili kombinasyonu kadar başlangıç çözümü bularak bu çözümler üzerinde silinebilir tepe ve kaydırılabilir tepeler aramaktadır. Burada önerilen kaydırma işleminin sadece bir r tamsayısı kadar yapılma koşulu vardır. Başlangıç çözümü çok geniş olduğu için algoritma bilinen örnekler üzerinde tam sonuç verse bile 450 tepeli olan çizgelerde dahi çözüm bulması günler sürmektedir (Dharwadker, 2011).

Bu tezde önerilen algoritmanın diğer çalışmadan farkı; kaydırma işleminin daha akılcı yapılması ve silinebilir ve kaydırılabilir tepeler kalmayınca tekrar bu tip tepeler oluşturulması için çözüm kümesinin alt kümelerinde değişimler yapılmasıdır.

Parçalama Algoritması

Girdi: Aşikar olmayan, basit ve bağlantılı bir $G=(V,E)$ çizgesi

Çıktı: Çizgenin minimum tepe örtüsü değeri ($\beta(G)$).

1. Birleştirilmişlik matrisini oku ve çözüm kümesi $=\phi$ al.
2. Çizgenin en düşük dereceli tepesini bul, tepeye bağlı olan tüm ayrıtları sil ve tepenin komşularını çözüm kümesine ata.
3. Çizgede hala ayrıt var ise Adım 2'e git.
4. Silinebilir tepeleri çözüm kümesinden çıkar.
5. Kaydırılabilir tepelerin sayısını hesapla. Eğer bu tepelerden biri kaydırıldığında kaydırılabilir tepelerin sayısı artıyorsa bu tepeyi kaydır ve Adım 4'e git.
6. Terse çevirme işlemi çözüm kümesinin eleman sayısını arttırmıyorsa bu işlemi uygula ve Adım 4'e git.
7. Çözüm kümesinin boyutunu $\beta(G)$ ye ata.
8. Son.

5.2 Parçalama Algoritmasının Karmaşıklığı

Algoritmanın en kötü durum (*worst-case complexity*) karmaşıklığı şu şekilde hesaplanabilir. $G(V,E)$ bir yol çizge olsun. Çizgedeki minimum tepe derecesine sahip olan tepeyi bulmanın maliyeti $O(n)$ dir. Bulunan tepeden sonra bitişiklik matrisin güncellenmesi için gereken maliyette $O(n)$ eşittir. Burada başlangıç çözümü için en kötü durum çizgenin yol çizge olmasıdır. Çizge yol çizge olduğu için parçalama işlemi $n/2$ kere tekrar edilecektir. Böylece başlangıç çözümü için gereken toplam maliyet $n/2(O(n)+O(n))=O(n^2)$ e eşittir. Silme işlemi en kötü durumda maksimum bağımsız kümenin eleman sayısı kadar olabilir, her bir silme için en fazla kaydırma işleminin karmaşıklığı da maksimum bağımsız kümenin eleman sayısına eşit olabilir. Bu durumda iyileştirme teknikleri de $O(\alpha(G)^3)$ eşit olmaktadır. Sonuç olarak algoritmanın genel karmaşıklığı $O(n^2+\alpha(G)^3)$ tür.

5.3 Parçalama Algoritmasının Hesaplama Denemeleri

Bu bölümde dördüncü bölümde anlatılan parçalama algoritmasının sonuçları ile literatürdeki bu problem için geliştirilmiş algoritmaların bulduğu sonuçlar karşılaştırılmıştır.

Önerilen algoritma C++ dilinde kodlanmıştır. Program Linux ortamında gcc derleyicisi ile –O1 optimizasyonu kullanılarak derlenmiştir. Hesaplama denemeleri 2.6 GHz Intel Core 2 Duo CPU işlemci ve 2 GB RAM e sahip işletim sistemi 32-bit Ubuntu olan bir bilgisayar üzerinde yapılmıştır. Algoritmanın çalışma zamanı tüm algoritmalarda ortak olduğu için çizge okumayı ve bitişiklik matrisinin oluşturulmasını içermemektedir.

Parçalama algoritmasının literatürdeki algoritmalarla kıyaslaması DIMACS klik örnekleri üzerinde yapılmıştır. Kütüphanedeki çizgeler klik örnekleri olduğu için parçalama algoritması çizgelerin tümleyenleri üzerinde çalıştırılmıştır. Örneklerin isim boyut, yoğunluk, maksimum klik değerleri DIMACS internet sitesi üzerinden alınmıştır.

Önerilen algoritma maksimum klik problemi için önerilen “Continuous Based Heuristic” (CBH) ve maksimum bağımsız küme için önerilen “Heuristic Based On Optimization Of A Quadratic Over A Sphere” (QSH) ile karşılaştırılmıştır.

Çizelge 5.1 ve Çizelge 5.2’de, Parçalama Algoritmasının (PA), CBH ve QSH’ın performans denemeleri kıyaslanmıştır. Tablodaki her bir satır için, ilk sütun çizgenin ismini, ikinci sütun çizgenin boyutunu (tepe sayısını), üçüncü sütun çizgenin yoğunluğunu ve dördüncü sütun da çizgenin minimum tepe örtüsü değerini göstermektedir. Kalan sütunlar sırasıyla CBH, QSH ve PA’nın bulduğu çözümleri ve son sütun da PA’nın çalışma süresini göstermektedir

Çizelge 5.1 PA nın DIMACS problemleri üzerinde literatürdeki çalışmalar ile kıyaslanması (1)

$\overline{G}$	**|V|**	**Yoğunluk**	$\beta(G)$	**CBH**	**QSH**	**PA**	**Süre(sn)**
brock200_1	200	0.745	179	180	179	180	0.02
brock200_2	200	0.496	188	188	188	190	0.02
brock200_3	200	0.605	185	186	185	187	0.02
brock200_4	200	0.658	183	184	183	185	0.01
brock400_1	400	0.748	373	377	373	377	0.08
brock400_2	400	0.749	371	376	371	377	0.08
brock400_3	400	0.748	369	377	369	378	0.10
brock400_4	400	0.749	367	376	367	377	0.08
brock800_1	800	0.649	777	780	783	781	0.40
brock800_2	800	0.651	776	781	776	780	0.40
brock800_3	800	0.649	775	780	775	781	0.37
brock800_4	800	0.650	774	781	774	782	0.35
c-fat200-1	200	0.077	188	188	188	188	0.00
c-fat200-2	200	0.163	176	176	176	176	0.01
c-fat200-5	200	0.426	142	142	142	142	0.02
c-fat500-1	500	0.036	486	486	486	486	0.05
c-fat500-2	500	0.374	474	474	474	474	0.08
c-fat500-5	500	0.073	436	436	436	436	0.18
c-fat500-10	500	0.186	374	374	374	374	0.39
hamming6-2	64	0.905	32	32	32	32	0.00
hamming6-4	64	0.349	60	60	60	60	0.00
hamming8-2	256	0.969	128	128	128	128	0.30
hamming8-4	256	0.639	240	240	240	240	0.00

Çizelge 5.2 PA nın DIMACS problemleri üzerinde literatürdeki çalışmalar ile kıyaslanması (2)

$\overline{G}$	|V|	Yoğunluk	$\beta(G)$	CBH	QSH	PA	Süre(sn)
johnson8-2-4	28	0.556	24	24	24	24	0.00
johnson8-4-4	70	0.768	56	56	56	56	0.00
johnson16-2-4	120	0.765	112	112	112	112	0.00
johnson32-2-4	496	0.879	480	480	480	480	0.40
keller4	171	0.649	160	161	160	160	0.00
keller5	776	0.751	749	755	752	752	0.91
MANN_a9	45	0.927	29	29	29	29	0.00
MANN_a27	378	0.990	252	257	253	252	3.92
p_hat300-1	300	0.244	292	292	293	292	0.03
p_hat300-2	300	0.489	275	275	276	275	0.05
p_hat300-3	300	0.744	264	264	267	264	0.09
p_hat500-1	500	0.253	491	491	491	491	0.07
p_hat500-2	500	0.505	464	465	467	464	0.28
p_hat500-3	500	0.752	450	451	454	451	0.59
p_hat700-1	700	0.249	689	691	692	691	0.16
p_hat700-2	700	0.498	656	656	658	656	1.05
p_hat700-3	700	0.748	638	640	641	638	2.04
p_hat700-3	700	0.748	638	640	641	638	2.04
san200_0.7_1	200	0.700	170	185	170	170	0.17
san200_0.7_2	200	0.700	182	188	182	185	0.07
san200_0.9_1	200	0.900	130	154	130	130	0.38
san200_0.9_2	200	0.900	140	164	140	140	0.07
san200_0.9_3	200	0.900	156	170	165	156	0.05
san400_0.5_1	400	0.500	387	392	391	391	0.10
san400_0.7_1	400	0.700	360	380	360	378	4.05
san400_0.7_2	400	0.700	370	385	370	381	0.52
san400_0.7_3	400	0.700	378	386	384	383	0.10
san400_0.9_1	400	0.900	300	350	300	300	6.48
sanr200_0.7	200	0.700	182	182	185	182	0.01
sanr200_0.9	200	0.900	158	159	163	159	0.06
sanr400_0.5	400	0.500	387	388	389	388	0.04
sanr400_0.7	400	0.700	379	380	382	381	0.08

Çizelge 5.1 ve Çizelge 5.2'den anlaşılacağı gibi PA, QSH neredeyse aynı kalitede, CBH tan ise daha kaliteli sonuçlar vermiştir. Çizelgelerden QSH ın brock ve san çizge ailelerinde, PA'nın ise p_hat ve MANN_a çizge ailelerinde diğer algoritmalara göre daha iyi sonuçlar verdiği anlaşılmaktadır.

Parçalama algoritması literatür çalışmalarının yanında ayrıca BHOSLIB kütüphanesi örnekleri üzerinde test edilmiştir. BHOSLIB kütüphanesi örnekleri maksimum bağımsız küme ve minimum tepe örtüsü problemleri için oluşturulmuş çizge örneklerinden oluşur. Bu kütüphanenin örnekleri önce birbirinden farklı *n* klik oluşturur, daha sonra bu klikler arasında rastgele ayrıtlar yerleştirir.

Çizelge 5.3 ve 5.4'de, Parçalama algoritmasının BHOSLIB kütüphane çizgeleri üzerinde performansı gösterilmiştir. Tablodaki her bir satır için, ilk sütun çizgenin ismini, ikinci sütun çizgenin boyutunu (tepe sayısını) ve üçüncü sütun da çizgenin minimum tepe örtüsü değerini göstermektedir. Kalan üç sütun sırasıyla PA'nın bulduğu çözümleri, PA'nın çalışma süresini ve bulunan çözümün hata oranını göstermektedir.

Çizelge 5.3'teki hata oranı (*bağıl hata*) aşağıdaki şekilde hesaplanmaktadır:

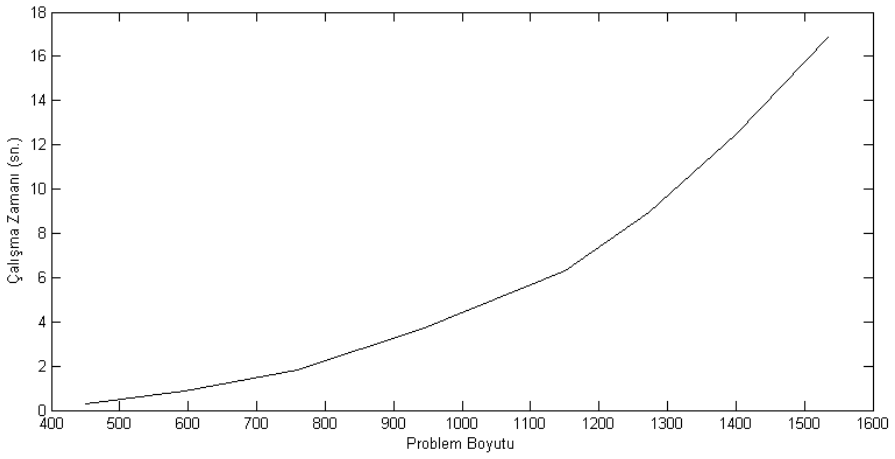
$$Hata\ Oranı = \left(\frac{bulunan\ çözüm - optimum\ çözümü}{optimum\ çözümü} \right) * 100$$

Çizelge 5.3 PA nın BHOSLIB problemleri üzerinde performansı (1)

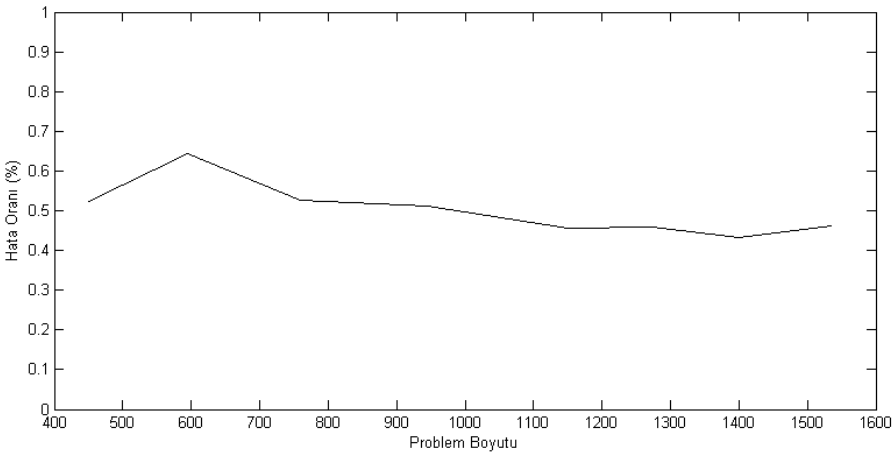
G	|V|	$\beta(G)$	PA	Süre(sn)	Hata Oranı
frb30-15-1	450	420	422	0.27	0.476
frb30-15-2	450	420	422	0.28	0.476
frb30-15-3	450	420	423	0.28	0.714
frb30-15-4	450	420	422	0.28	0.476
frb30-15-5	450	420	422	0.30	0.476
frb35-17-1	595	560	564	0.85	0.714
frb35-17-2	595	560	564	0.58	0.714
frb35-17-3	595	560	563	0.83	0.536
frb35-17-4	595	560	564	1.17	0.714
frb35-17-5	595	560	563	0.81	0.536
frb40-19-1	760	720	723	2.38	0.417
frb40-19-2	760	720	724	1.68	0.556
frb40-19-3	760	720	723	2.01	0.417
frb40-19-4	760	720	725	1.08	0.694
frb40-19-5	760	720	724	1.86	0.556

Çizelge 5.4 PA nın BHOSLIB problemleri üzerinde performansı (2)

G	|V|	$\beta(G)$	PA	Süre(sn)	Hata Oranı
frb45-21-1	945	900	905	3.41	0.556
frb45-21-2	945	900	904	3.60	0.444
frb45-21-3	945	900	905	3.48	0.556
frb45-21-4	945	900	904	4.53	0.444
frb45-21-5	945	900	905	3.60	0.556
frb50-23-1	1150	1100	1104	5.17	0.364
frb50-23-2	1150	1100	1106	6.49	0.545
frb50-23-3	1150	1100	1105	5.31	0.455
frb50-23-4	1150	1100	1105	6.47	0.455
frb50-23-5	1150	1100	1105	8.03	0.455
frb53-24-1	1272	1219	1225	10.8	0.492
frb53-24-2	1272	1219	1224	7.77	0.410
frb53-24-3	1272	1219	1223	9.25	0.328
frb53-24-4	1272	1219	1225	9.01	0.492
frb53-24-5	1272	1219	1226	7.96	0.574
frb56-25-1	1400	1344	1350	14.29	0.446
frb56-25-2	1400	1344	1350	14.44	0.446
frb56-25-3	1400	1344	1349	10.72	0.372
frb56-25-4	1400	1344	1350	12.00	0.446
frb56-25-5	1400	1344	1350	10.91	0.446
frb59-26-1	1534	1475	1482	15.68	0.475
frb59-26-2	1534	1475	1482	17.39	0.475
frb59-26-3	1534	1475	1482	16.88	0.475
frb59-26-4	1534	1475	1481	15.85	0.407
frb59-26-5	1534	1475	1482	18.59	0.475



Şekil 5.3 Problem boyutu ve çalışma zaman arasındaki ilişki



Şekil 5.4 Problem boyutu ve hata oranı arasındaki ilişki

Şekil 5.3 ve 5.4'de algoritmanın problem boyutu ile performansı arasındaki ilişki gösterilmiştir. Problemin boyutlarının artmasına rağmen hata oranının artmaması algoritmanın efektifliğini göstermektedir.

6. HİBRİD GENETİK ALGORİTMALAR

Bu bölümde genetik algoritmalar ve hibrid genetik algoritmalar incelenerek, minimum tepe örtüsü problemi için yeni bir hibrid genetik algoritma önerilmiştir.

6.1 Genetik Algoritmalar

6.1.1. Genetik algoritmaların tarihçesi ve temelleri

Genetik Algoritmalar öğrenme işleminde doğadaki evrime dayalı bir yaklaşım kullanan Evrimsel Hesaplamaların (Evolutionary Computing) bir alt bölümüdür. "Genetik Algoritmalar" terimi bilimsel literatürde ilk defa J. D. Bagley'in bir çalışmasında ortaya atılmıştır (Bagley 1967). Daha sonra 1975'te yayınlanan J. Holland'ın "Doğal ve Yapay Sistemlerde Uyum (Adaptation in Natural and Artificial Systems)" kitabında Genetik algoritmaların temelleri atılmış ve teorik esasları gösterilmiştir (Holland 1975). J. Holland'ın öğrencisi olan D. E. Goldberg, çok sayıda kollara ayrılan gaz borularındaki gaz akışını düzenlemek için genetik algoritmaları kullanarak, genetik algoritmaların gerçek hayatta kullanılabilir olduğunu göstermiştir (Goldberg 1983).

6.1.2. Genetik operatörler

Genetik algoritma için çözümler problemin tipine uygun şekilde kodlanmalıdır. Kodlama türleri arasında en çok tercih edilen 0-1 kodlamadır. Çözümler kodlandıktan sonra genetik algoritma için başlangıç popülasyonun oluşturulması gerekmektedir. Bir popülasyon kromozom topluluğu, kromozom ise genler topluluğu olarak açıklanabilir. Başlangıç popülasyonu rastgele ya da hızlı sezgisel çözüm yöntemleri ile oluşturulabilir.

Başlangıç çözümü oluşturulduktan sonra yeniden üreme, çaprazlama ve dönüşüm (mutasyon) ile popülasyon iyileştirilir.

Yeniden üreme operatörü, doğadaki doğal seleksiyon işleminin genetik algoritmalar tarafınca uyarlanmasıdır. Bu işlem güçlü bireylerin hayatta kalmasını ve güçlü çocukların oluşturulmasını sağlamaktadır (Sakawa, 2002).

Çaprazlama genetik algoritma süresince kullanılan operatörler arasında en önemli olan operatördür ve amacı popülasyonda bireyler arasındaki çeşitliliğin sağlanmasıdır (Sakawa, 2002; Vural, 2005). Bu işlem yeni bireylerin oluşturma sürecinde yer alır. Problem çözümleri için birçok çaprazlama yöntemi üretilmiştir (Sakawa, 2002). Genel olarak tüm çaprazlama operatörlerinde, iki birey bir çiftleşme havuzundan alınarak karşılıklı gen değişimiyle gerçekleşir (Deb, 2001).

Dönüşüm operatörü nesiller boyunca çeşitliliğinin sağlanması ve yerel optimumlara takılmaması için kullanılır. Dönüşüm operatörü bir kromozomun genlerinden birini ya da bir kaçını değiştirir. Dönüşüm operatörünün en önemli özelliklerinden biri dönüşüm olasılığının iyi bir şekilde ayarlanmasıdır. Dönüşüm olasılığı bir genin dönüşüme uğrama olasılığını temsil etmektedir. Bir popülasyon içinde dönüşümün beklenen değeri dönüşüm olasılığının o popülasyondaki gen sayısıyla çarpımı ile elde edilir (Vural, 2005). Dönüşüm olasılığı genel olarak 0.001 ile 0.01 arasında kullanılır.

6.2 Hibrid Genetik Algoritmalar

Hibrid genetik algoritmalar gerçek hayat problemlerinin çözümlerinde gittikçe önem kazanmaya başlamıştır. Genetik algoritmalar diğer teknikler ile işbirliği oluşturarak hibrid yapılar oluşturmaya çok uygundurlar.

Diğer tüm global optimizasyon algoritmaları gibi genetik algoritmaların performansı da çözüm uzayının genişletilmesi ve çözüm uzayındaki en iyi çözümün bulunması arasındaki dengeye dayanır. Genetik algoritmaların en güçlü yönü bu iki işlemini de kombine etmesinden gelir. Ancak bu özellikler genetik algoritmalar için teorik olarak doğru olsa da pratikte problemler olabilir. Çünkü Holland popülasyon büyüklüğünün sınırsız olduğunu ve uygunluk fonksiyonun çözümün uygunluğunu tam olarak yansıtacağını varsaymıştır (El-Mihoub et al, 2006).

Limitli popülasyonundan dolayı, genetik algoritma iyi arama bölgelerini kötü olarak, kötü arama bölgelerini de iyi olarak düşünebilir. Yerel iyileştirme teknikleri farklı bölgelerin temsilini adil bir şekilde yaparak, algoritmanın yerel optimuma takılma ihtimalini düşürür (Kotecha and Gambhava, 2003).

Genetik algoritmalar hızlı bir şekilde global optimumun bulunduğu bölgeyi tespit etse de, göreceli olarak bölge içindeki global optimum noktasını bulması zaman alabilir. Genetik algoritma operatörleri çözümlerin etrafında küçük hareketler yapamadığından dolayı, global optimumun bölgesi bulunsa bile bu nokta bulunamayabilir. Genetik algoritmanın yerel optimizasyon yöntemi ile kombine edilmesi global optimumun bulunmasını hızlandırır. Genel olarak hibrid genetik algoritmalar çözümün kalitesini arttırma ve global optimum çözümün bulunma zamanın kısaltılması açısından önemlidir.

6.3 Minimum Tepe Örtüsü Problemi İçin Yeni Bir Hibrid Genetik Algoritma

Genetik algoritmanın başlangıç popülasyonu açgözlülük yöntemi ile oluşturulmuştur. Bu problem için uygunluk fonksiyonun değeri, çözüm kümesi içindeki eleman sayısına eşittir. Uygunluk fonksiyonun değerine göre bireyler sıralanır. En küçük değerin en çok şansa, en büyük değerinde en az şansa sahip olduğu şekilde rastgele sıralama yaparak tek nokta çaprazlama tekniği uygulanır. Bu işlem birey sayının yüzde sekseni kadar yapılır ve diğer bireyler ise doğrudan yeni nesle geçer. Çaprazlamadan sonra her bir bireye 4. Bölümde tanımları verilen silme ve kaydırma işlemleri uygulanarak çözümler geliştirilir.

Silme ve kaydırma operatörlerinden sonra çözüme herhangi bir tepe eklemeden bir tepe çıkarmak çözümün uygunluğunu bozacaktır. Bu yüzden mutasyon operatörü değiştirilerek sadece çözüme eleman eklemektedir. Bu yaklaşım ilk bakışta çözümün kalitesini düşürecek gibi görünse de popülasyonun çeşitliliğini arttırarak yerel optimumlara takılma riskini azaltmaktadır.

Önerilen hibrid genetik algoritma (Ugurlu, 2012b) (HGA), Balas ve Niehaus (1996) tarafından maksimum klik için önerilen “The CliqMerge Method” ve

Aggarwal ve diğerleri (1997) tarafından maksimum bağımsız küme için önerilen "Optimized Crossover" (OCH) yöntemleri ile karşılaştırılmıştır.

Çizelge 6.1'de HGA, CliqMerge ve OCH'nin performans kıyaslamaları verilmiştir. Tablodaki her bir satır için, ilk sütun çizgenin ismini, ikinci sütun çizgenin boyutunu (tepe sayısını), üçüncü sütun çizgenin minimum tepe örtüsü değerini göstermektedir. Kalan sütunlar sırasıyla OCH, CliqMerge ve HGA'nın bulduğu sonuçları göstermektedir.

Çizelge 6.1 Hibrid algoritmanın geçmiş çalışmalar ile kıyaslanması

$\overline{G}$	**N**	$\beta(G)$	**OCH**	**CliqMerge**	**HGA**
brock200-2	200	188	189	189	189
brock200-4	200	183	184	184	184
brock400-2	400	371	376	375	376
brock400-4	400	367	376	375	376
brock800-2	800	776	781	779	780
brock800-4	800	774	781	779	780
keller4	171	160	160	160	160
keller5	776	749	752	749	750
MANN-a27	378	252	252	252	253
MANN-a45	1035	690	692	691	693

7. SONUÇ

Çizge problemlerinin en önemlilerinden biri olan minimum tepe örtüsü probleminin incelendiği bu tezde, problem için yeni algoritmalar önerilmiş ve bu algoritmaları esas alan C++ dilinde bilgisayar programı oluşturulmuştur. Geliştirilen program uluslararası problem kütüphanelerinden alınan test problemleri üzerinde denenmiş ve literatürdeki çalışmalar ile kıyaslanmıştır. Elde edilen sonuçlar birlikte ulusal ve uluslararası kongrelerde sunulmuştur.

KAYNAKLAR DİZİNİ

Aggarwal, C.C., Orlin, J.B. and Tai, R.P., 1997, Optimized crossover for the independent set problem, Operations Research, 45(2):226-234pp.

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

Andrade, D.V.,. Resende, M.G.C. and Werneck, R.F., 2008, Fast local search for the maximum independent set problem, In: International Workshop on Experimental Algorithms(WEA).

Balasubramanian, R., Fellows, M.R. and Raman, V., 1998, An improved fixed parameter algorithm for vertex cover, *Inform. Process. Lett.* 65: 163–168pp.

Balas E. and Niehaus W., 1996, Finding Large Clique by Bipartite Matching, DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 26.

Bar-Yehuda, R., and Even, S. (1985), A local-ratio theorem for approximating the weighted vertex cover problem, in Analysis and Design of Algorithms for Combinatorial Problems, (25), 27–46pp.

Battiti, R. and Protasi, M., 2001, Reactive local search for the maximum clique problem, Algorithmica, 29(4):610–637pp.

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

Buss, J.F. and Goldsmith. J., 1993, Nondeterminism within P, SIAM Journal on Computing, 22(3):560-572.

Chen, J., Kanj, I.A. and Jia, W., 1999, Vertex cover: further observations and further improvements, Journal of Algorithms, 313-324pp.

Clarkson, K.L., 1985, Modification of the greedy algorithm for vertex cover, Info. Proc. Letters, 16:23-26pp.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C., 2001, Introduction To Algorithms, The MIT Press, Cambridge, 1180p.

Cura, T., 2008, Modern Sezgisel Teknikler ve Uygulamaları, Papatya Yayıncılık, İstanbul, 173s.

Deb, K., 2001, Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons Ltd., England.

KAYNAKLAR DİZİNİ (devam)

Dharwadker, A., 2011, The Vertex Cover Algorithm, Createspace.

Dinur, I. and Safra, S., 2005, On the hardness of approximating minimum vertex cover, Annals of Mathematics, 162: 439-485pp.

Downey, R.G. and Fellows, M.R., 1999, Parameterized complexity, Springer Verlag, New York

Downey, R.G., Fellows, M.R. and Stege, U., 1998, Parameterized complexity: A framework for systematically confronting computational intractability, *in* "Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future" (F. Roberts, J. Kratochvil, and J. Nesetril, Eds.), *AMS-DIMACS Proceedings Series*, 49:49–99pp.

Elmas, Ç., 2007, Yapay Zeka Uygulamalar , Seçkin Yayıncılık, Ankara.

El-Mihoub, T.A., Hopgood, A.A., Nolle, L. and Battersby, A., 2006, Hybrid genetic algorithms: a review, Engineering Letters, 13:2.

Evans, I.K., 1998, Evolutionary algorithms for vertex cover, Proc. of Evolutionary Programming VII, 1447: 377-386pp.

Gandhi, R., Halperin, E., Khuller, S., Kortsarz, G. and Srinivasan, A., 2003, An improved approximation algorithm for vertex cover with hard capacities, to appear in, in: Proc. of International Colloquium on Automata Languages and Programming.

Garey, M.R. and Johnson, D.S., 1979, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco.

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

Halperin, E., 2000, Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs, In ACM-SIAM Symposium on Discrete Algorithms, 329-337pp.

Hastad, J., 1997, Some optimal inapproximability results, In ACM Symposium on the Theory of Computing, 1-10pp.

Karaboğa, D., 2004, Yapay Zeka Optimizasyon Algoritmaları, Atlas Yayın Dağıtım, İstanbul, 199s.

Karakostas G. 2009, A Better Approximation Ratio for the Vertex Cover Problem, ACM Transactions on Algoritmhs, 5(4),.

KAYNAKLAR DİZİNİ (devam)

Karp. R. M., 1972, Reducibility among combinatorial problems, In Complexity of Computer Computations, 85-103pp.

Katayama, K., Hamamoto, A. and Narihisa, H., 2005, An effective local search for the maximum clique problem, Information Processing Letters, 95:503–511pp.

Khuri, S and Thomas, B., 1994, An evolutionary heuristic for the minimum vertex cover problem, KI-94 Workshops, 86-90pp.

Kotecha, K. and Gambhava, N., 2003, A hybrid genetic algorithm for minimum vertex cover problem, The First Indian International Conference on Artificial Intelligence, 904-913pp.

Kumar, K.V.R., 2009, Choosing the efficient algorithm for vertex cover problem, Computer Science and Engineering Department Thapar University.

Monien, B. and Speckenmeyer, E., 1985, Ramsey Numbers and an Approximation algorithm for the Vertex Cover Problem, Acta Informatica, , 22, 115-123pp.

Nabiyev, V.V., 2005, Yapay Zeka, Seçkin Yayıncılık, Ankara, 764s.

Nabiyev, V.V., 2009, Teoriden Uygulamalara Algoritmalar, Seçkin Yayıncılık, Ankara, 824s.

Niedermeier, R. and Rossmanith, P., 2000, A general method to speed up fixed-parameter tractable algorithms, *Inform. Process. Lett.* 73:125–129pp.

Niedermeier, R. and Rossmanith, P., 2003, On efficient fixed-parameter algorithms for weighted vertex cover, Journal of Algorithms, 47(2):63-77pp.

Oliveto, P.S. and Yao, X., 2007, Evolutionary algorithms and the vertex cover problem, Evolutionary Computation, 1870-1877pp.

Pitt, L., 1985, A simple probabilistic approximation algorithm for vertex cover, Tech. Rep. YaleU/DCS/TR-404, Department of Computer Science, Yale University.

Pullan, W.J. and Hoos, H.H., 2006, Dynamic local search for the maximum clique problem, J. Artificial Intelligence Research, 25:159–185pp

KAYNAKLAR DİZİNİ (devam)

Richter, S., Helmert, M. and Gretton, C., 2007, A stochastic local search approach to vertex cover, In Proceedings of the 30th German Conference on Artificial Intelligence (KI), 412–426pp.

Robson, J.M., 2001, Finding a maximum independent set in time O(2n=4), Technical Report1251(01).

Safar, M., 2007, Modeling the communication problem in wireless sensor networks as a vertex cover, Computer Systems and Applications, 592-598pp.

Sakawa, M., 2002. Genetic Algorithms and Fuzzy Multi objective Optimization, Kluwer Academic Publishers.

Stege, U. and Fellows, M., 1999, An improved fixed-parameter-tractable algorithm for vertex cover, Technical Report 318, Department of Computer Science, ETH Zurich.

Strickland, D.M., Barnes, E. and Sokol, J.S., 2005, Optimal protein structure alignment using maximum cliques, Operations Research, 53: 389-402pp.

Ugurlu, O., 2012a, A new heuristic algorithm for unweighted minimum vertex cover, Proceeding of the fourth International Conference "Problems of Cybernetics and Informatics, Section VII, "Control and Optimization, Baku, Azerbaijan, 69-73pp.

Ugurlu, O., 2012b, A new heuristic mutation method for solving vertex cover problem, 8th International Symposium of Statistics, Abstracts Book, Eskisehir, Turkey, 381-382pp.

Xu, X. and Ma, J., 2006, An efficient simulated annealing algorithm for the minimum vertex cover problem, Neurocomputing, 69: 913-916pp.

Vural, M., 2005. Genetik Algoritma Yöntemi ile Toplu Üretim Planlama, Yüksek Lisans Tezi, T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

Benchmarks with Hidden Optimum Solutions for Graph Problems (BHOSLIB), available at: www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm **(**Erişim tarihi**:** 01 Ocak 2013)

DIMACS clique benchmarks, Benchmark instances made available by electronic transfer at dimacs.rutgers.edu, **(**Erişim tarihi**:** 01 Ocak 2013) Rutgers Univ., Piscataway. NJ.

ÖZGEÇMİŞ

5 Nisan 1988 tarihinde Ankara'nın Keçiören ilçesinde doğdu. Lise eğitimini Aldemir Atilla Konuk Anadolu Lisesinde tamamladı. 2007 yılında girdiği üniversite sınavında Ege Üniversitesi Matematik bölümünü kazandı. 2011 yılında Bilgisayar Bilimleri Ağırlık Matematikçi unvanını alarak mezun oldu.

YAYINLAR

Makaleler

A) Uluslararası Science Citation Index ve Sosial Science Citation Indexce taranan dergilerde yayınlanmış makaleler

1. Isıklı S., **Ugurlu O.**, Kizilates G., Kitis O., Ozan E., Eker C., Coburn K, Gonul A. S., "Altered Hippocampal Formation Shape In First-Episode Depressed Patients at Five Year Follow-Up", Journal of Psychiatric Research, vol. 47 (1), 2013, 50-55.

B) Diğer bilimsel dergilerde yayınlanmış makaleler

1. Kizilates G., Berberler M. E., **Ugurlu O.**, "A New Solution Approach for the Integer Fair Division Problem", World Applied Sciences Journal, 13 (12), 2444-2450, 2011.

2. Berberler M. E., **Uğurlu O.**, Kızılateş G., " Macar Algoritmasının Sıfırları Kapatma Alt Yordamı Üzerine", Pamukkale University Journal of Engineering Sciences, vol. 2, 85-94, 2012.

Kongreler

A) Uluslararası bilimsel toplantılarda yayınlanmış bildiriler:

1. **Ugurlu O.**, Berberler M. E., Kizilates G., Kurt M., "A New Algorithm for Finding Minimum Vertex Cut Set", Proceeding of the fourth International

Conference “Problems of Cybernetics and Informatics”, Section VII, "Control and Optimization" , p. 145-148, Baku, Azerbaijan, September 12-14, 2012.

2. **Ugurlu O.**, “A New Heuristic Algorithm for Unweighted Minimum Vertex Cover”, Proceeding of the fourth International Conference “Problems of Cybernetics and Informatics”, Section VII, "Control and Optimization", p. 69-73, Baku, Azerbaijan, September 12-14, 2012.

3. **Ugurlu O.**, “A New Heuristic Mutation Method for Solving Vertex Cover Problem”, 8th International Symposium of Statistics, Abstracts Book, p. 381-382, Eskisehir, Turkey, October 11-13, 2012.

4. **Ugurlu O.**, Nuriyeva F., “A New Heuristic Algorithm for Rainbow Vertex Connection”, Theoretical and Applied Aspects of Cybernetics, “Proceedings Of The 2nd International Scientific Conference of Students and Young Scientists Theoretical and Applied Aspects of Cybernetics”, p.74-78, Kiev, Ukraine, November 12-16, 2012.

B) Ulusal bilimsel toplantılarda yayınlanmış bildirileri

1. Işıklı S., **Uğurlu O.**, Kızılateş G., Durmuşoğlu T. E., Çınar C., Ozan E., Eker M., Kitiş O., Gönül A. S., “Depresyonda Hipokampus Alt Bölgelerinin Değişimi 5 Yıllık İzlem Çalışması”, 47. Ulusal Psikiyatri Kongresi, Özet Kitabı, p. 355, Antalya, Türkiye, Ekim 26-30, 2011.

2. **Uğurlu O.**, “Minimum Tepe Örtüsü Problemi için Yeni bir Çözüm Yaklaşımı”, 25. Ulusal Matematik Sempozyumu, Özet Kitapçığı, p. 88, Niğde, Türkiye, Eylül 7-10, 2012.

3. Atlamaz M., Yılmaz A, **Uğurlu O.**, Bilgi M. M., Gönül A. S., “Geri Bildirimler Karar Vermeyi Nasıl Etkiliyor? Teknolojik Geri Bildiriminin İnsan Geri Bildirimi ile Karşılaştırılması”, 48. Ulusal Psikiyatri Kongresi, Özet Kitabı, p. 231, Bursa, Türkiye, Ekim 5-8, 2012.

EKLER

Ek 1 Parçalanma Algoritmasının C++ Programlama Dili Kodu

Ek 1 Parçalama Algoritmasının C++ Programlama Dili Kodu

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

#define n 1000

int  a[n+2][n+2]={0};
int  aa[n+2][n+2]={0};

int  k[n+2][n+2]={0};
int  d[n+1]={0};

int  vcset[n+1]={0};
int  svc[n+1]={0};
int  bestvc[n+1]={0};

int add[n+1]={0};
int erase[n+1]={0};
int use[n+1]={0};

int shf1[n+1]={0};
int shf2[n+1]={0};
int shf3[n+1]={0};

char c,s4[5];

int i,j,z,jj,edge,p1,p2,p3,zz;
int min,hangi;
int ii,ii1,m;
int s1,s2,s3,ss1,ss2;
int i2,j2,z2,han,sayi,max,sf1,sf2,sf3;
int iter;
int tsayi,ti=0,e1,e2,e3,sin;
```

```
int main()
{

FILE *f;

        f=fopen("D1000.txt","r");

        fscanf(f,"%c %s %d %d \n",&c,&s4,&i,&edge);

        if (i!=n)
        {
          printf("Boyut uyumsuz !");
          return 0;
        }

        for (zz=1;zz<=edge;zz++)
        {
           fscanf(f,"%s",&p1);
          fscanf(f,"%d",&p2);
          fscanf(f,"%d",&p3);

          a[p2][p3]=1;
          a[p3][p2]=1;

          aa[p2][p3]=1;
          aa[p3][p2]=1;
        }

        fclose(f);

        for (i=1;i<=n;i++)
          for (j=1;j<=n;j++)
              if (a[i][j]==1)
                      {
```

```
                        k[i][0]++;
                        k[i][k[i][0]]=j;
                        aa[i][0]++;
                }


        clock_t start, end;
         start = clock();


tekrar:
    min=10000;
        for(i=1;i<=n;i++)
          if ((aa[i][0]<min) && (use[i]==0) && (vcset[i]==0))
          {
                min=aa[i][0];
                hangi=i;
          }


        use[hangi]=1;
        for(i=1;i<=n;i++)
        {
          if (aa[hangi][i]==1)
          {
                aa[i][hangi]=0;
                aa[hangi][i]=0;

                aa[hangi][0]--;
                aa[i][0]--;


                if (vcset[i]==0)
                {
                        vcset[0]++;
```

```
                    vcset[i]=1;

                    for(j=1;j<=k[i][0];j++)
                    {
                        a[i] [k[i][j]]++;
                        a[k[i][j]] [i]++;

                        if (vcset[k[i][j]]==0) aa[k[i][j]][0]--;
                    }
                }
            }
        }

        for(i=1;i<=n;i++)
          for(j=i+1;j<=n;j++)
                if (a[i][j]==1) goto tekrar;

again:
        for(i=1;i<=n;i++)
        {
          if (vcset[i]==1)
          {
                for(j=1;j<=k[i][0];j++)
                    d[i]=d[i]+a[i][k[i][j]];
          }
        }

        for(i=1;i<=n;i++)
          for(j=i+1;j<=n;j++)
                aa[i][j]=0;
tkrr:

        for(i=1;i<=n;i++)
          if (vcset[i]==1)
          {
```

```
                if (k[i][0]*3==d[i])
                {
                        for(j=1;j<=k[i][0];j++)
                        {
                                a[i][k[i][j]]--;
                                a[k[i][j]][i]--;
                                aa[i][k[i][j]]=1;
                                aa[k[i][j]][i]=1;
                                d[k[i][j]]--;
                        }
                        vcset[i]=0;
                        vcset[0]--;
                        d[i]=0;
                }
          }

        for(i=n;i>0;i--)
        {
         if (vcset[i]==1)
          {
                if (d[i]==k[i][0]*3-1)
                {
                        for(j=1;j<=k[i][0];j++)
                                if (a[i][k[i][j]]==2)
                                {
                                        hangi=k[i][j];
                                        for(z=1;z<=k[hangi][0];z++)
                                                if (a[hangi][k[hangi][z]]==2 &&
vcset[k[hangi][z]]==1 && d[k[hangi][z]]== k[k[hangi][z]][0]*3-1 &&
i!=k[hangi][z] && a[i][k[hangi][z]]==0)
                                                {
                                                        vcset[0]--;

                                                        vcset[i]=0;
                                                        for(jj=1;jj<=k[i][0];jj++)
                                                        {
```

```
                                                    a[i][k[i][jj]]--;
                                                    a[k[i][jj]][i]--;
                                                    if    (vcset[k[i][jj]]==1)
d[k[i][jj]]--;
                                                }

                                                vcset[k[hangi][z]]=0;

        for(jj=1;jj<=k[k[hangi][z]][0];jj++)
                                                {

        a[k[hangi][z]][k[k[hangi][z]][jj]]--;

        a[k[k[hangi][z]][jj]][k[hangi][z]]--;
                                                    if
(vcset[k[k[hangi][z]][jj]]==1) d[k[k[hangi][z]][jj]]--;
                                                    }

                                                vcset[hangi]=1;
                                                for(jj=1;jj<=k[hangi][0];jj++)
                                                {

        a[hangi][k[hangi][jj]]++;

        a[k[hangi][jj]][hangi]++;
                                                    if
(vcset[k[hangi][jj]]==1) d[k[hangi][jj]]++;

        d[hangi]=d[hangi]+a[hangi][k[hangi][jj]];
                                                }
                                                    goto tkrr;
                                            }
                                    }
                            }
                    }
          }
        }

        for(i=1;i<=n;i++)
```

```
if ( (vcset[i]==1) && (d[i]==k[i][0]*3-2) )
{
	for(j=0;j<=n;j++)
	{
		shf1[j]=0;
		shf2[j]=0;
	}

	z=0;
	for(j=1;j<=k[i][0];j++)
		if (a[i][k[i][j]]==2)
		{
			z++;
			if (z==1) s1=k[i][j];
			if (z==2) s2=k[i][j];
		}

	for(z=1;z<=k[s1][0];z++)
	{
		if ((vcset[k[s1][z]]==1) && (d[k[s1][z]]==k[k[s1][z]][0]*3-
1))
		{
			shf1[0]++;
			shf1[shf1[0]]=k[s1][z];
		}
	}

	for(z=1;z<=k[s2][0];z++)
	{
		if ((vcset[k[s2][z]]==1) && (d[k[s2][z]]==k[k[s2][z]][0]*3-
1))
		{
			shf2[0]++;
			shf2[shf2[0]]=k[s2][z];
		}
	}
```

```
                if ((shf1[0]!=0) && (shf2[0]!=0))
                for(j=1;j<=shf1[0];j++)
                        for(z=1;z<=shf2[0];z++)
                        {
                                if  ((a[shf1[j]][shf2[z]]==0)  &&  (a[shf1[j]][i]==0)
&& (a[i][shf2[z]]==0))
                                {

                                vcset[0]--;

                                vcset[shf1[j]]=0;
                                d[shf1[j]]=0;
                                for(jj=1;jj<=k[shf1[j]][0];jj++)
                                {
                                        a[shf1[j]][k[shf1[j]][jj]]--;
                                        a[k[shf1[j]][jj]][shf1[j]]--;
                                        if (vcset[k[shf1[j]][jj]]==1) d[k[shf1[j]][jj]]--
;
                                }

                                vcset[s1]=1;
                                for(jj=1;jj<=k[s1][0];jj++)
                                {
                                        a[s1][k[s1][jj]]++;
                                        a[k[s1][jj]][s1]++;
                                        if (vcset[k[s1][jj]]==1) d[k[s1][jj]]++;
                                        d[s1]=d[s1]+a[s1][k[s1][jj]];
                                }

                                vcset[i]=0;
                                d[i]=0;
                                for(jj=1;jj<=k[i][0];jj++)
                                {
                                        a[i][k[i][jj]]--;
                                        a[k[i][jj]][i]--;
```

```
                        if (vcset[k[i][jj]]==1) d[k[i][jj]]--;
                    }

                    vcset[shf2[z]]=0;
                    d[shf2[z]]=0;
                    for(jj=1;jj<=k[shf2[z]][0];jj++)
                    {
                        a[shf2[z]][k[shf2[z]][jj]]--;
                        a[k[shf2[z]][jj]][shf2[z]]--;
                        if (vcset[k[shf2[z]][jj]]==1) d[k[shf2[z]][jj]]-
-;
                    }

                    vcset[s2]=1;
                    for(jj=1;jj<=k[s2][0];jj++)
                    {
                        a[s2][k[s2][jj]]++;
                        a[k[s2][jj]][s2]++;
                        if (vcset[k[s2][jj]]==1) d[k[s2][jj]]++;
                        d[s2]=d[s2]+a[s2][k[s2][jj]];
                    }

                    goto tkrr;
                    }
                }
            }


        for(i=1;i<=n;i++)
          if ( (vcset[i]==1) && (d[i]==k[i][0]*3-3) )
          {
                for(j=0;j<=n;j++)
                {
                    shf1[j]=0;
                    shf2[j]=0;
```

```
        shf3[j]=0;
    }

    z=0;
    for(j=1;j<=k[i][0];j++)
        if (a[i][k[i][j]]==2)
        {
            z++;
            if (z==1) s1=k[i][j];
            if (z==2) s2=k[i][j];
            if (z==3) s3=k[i][j];
        }

    for(z=1;z<=k[s1][0];z++)
    {
        if ((vcset[k[s1][z]]==1) && (d[k[s1][z]]==k[k[s1][z]][0]*3-
1))
        {
            shf1[0]++;
            shf1[shf1[0]]=k[s1][z];
        }
    }

    for(z=1;z<=k[s2][0];z++)
    {
        if ((vcset[k[s2][z]]==1) && (d[k[s2][z]]==k[k[s2][z]][0]*3-
1))
        {
            shf2[0]++;
            shf2[shf2[0]]=k[s2][z];
        }
    }

    for(z=1;z<=k[s3][0];z++)
    {
```

```
                if ((vcset[k[s3][z]]==1) && (d[k[s3][z]]==k[k[s3][z]][0]*3-
1))
                {
                        shf3[0]++;
                        shf3[shf3[0]]=k[s3][z];
                }
        }

        if ((shf1[0]!=0) && (shf2[0]!=0) && (shf3[0]!=0))
        for(j=1;j<=shf1[0];j++)
                for(z=1;z<=shf2[0];z++)
                        for(jj=1;jj<=shf3[0];jj++)
                {
                        if ((a[shf1[j]][shf2[z]]==0) &&
(a[shf1[j]][shf3[jj]]==0) && (a[shf3[jj]][shf2[z]]==0) && (a[shf1[j]][i]==0) &&
(a[i][shf2[z]]==0) && (a[i][shf3[jj]]==0))
                        {

                                e1=shf1[j];
                                e2=shf2[z];
                                e3=shf3[jj];

                                vcset[0]--;

                                vcset[i]=0;
                                d[i]=0;
                                for(sin=1;sin<=k[i][0];sin++)
                                {
                                        a[i][k[i][sin]]--;
                                        a[k[i][sin]][i]--;
                                        if (vcset[k[i][sin]]==1) d[k[i][sin]]--;
                                }

                                vcset[e1]=0;
                                d[e1]=0;
                                for(sin=1;sin<=k[e1][0];sin++)
```

```
{
        a[e1][k[e1][sin]]--;
        a[k[e1][sin]][e1]--;
        if              (vcset[k[e1][sin]]==1)
d[k[e1][sin]]--;
}

vcset[e2]=0;
d[e2]=0;
for(sin=1;sin<=k[e2][0];sin++)
{
        a[e2][k[e2][sin]]--;
        a[k[e2][sin]][e2]--;
        if              (vcset[k[e2][sin]]==1)
d[k[e2][sin]]--;
}

vcset[e3]=0;
d[e3]=0;
for(sin=1;sin<=k[e3][0];sin++)
{
        a[e3][k[e3][sin]]--;
        a[k[e3][sin]][e3]--;
        if              (vcset[k[e3][sin]]==1)
d[k[e3][sin]]--;
}

vcset[s1]=1;
for(sin=1;sin<=k[s1][0];sin++)
{
        a[s1][k[s1][sin]]++;
        a[k[s1][sin]][s1]++;
        if              (vcset[k[s1][sin]]==1)
d[k[s1][sin]]++;
        d[s1]=d[s1]+a[s1][k[s1][sin]];
}
```

```
                                        vcset[s2]=1;
                                        for(sin=1;sin<=k[s2][0];sin++)
                                        {
                                                a[s2][k[s2][sin]]++;
                                                a[k[s2][sin]][s2]++;
                                                if              (vcset[k[s2][sin]]==1)
d[k[s2][sin]]++;
                                                d[s2]=d[s2]+a[s2][k[s2][sin]];
                                        }

                                        vcset[s3]=1;
                                        for(sin=1;sin<=k[s3][0];sin++)
                                        {
                                                a[s3][k[s3][sin]]++;
                                                a[k[s3][sin]][s3]++;
                                                if              (vcset[k[s3][sin]]==1)
d[k[s3][sin]]++;
                                                d[s3]=d[s3]+a[s3][k[s3][sin]];
                                        }

                                        goto tkrr;

                                }
                        }
        }

        for(i=1;i<=n;i++)
          for(j=i+1;j<=n;j++)
                  if (a[i][j]==3) ii++;

        for(i=1;i<=n;i++)
          if ( (vcset[i]==1) && (d[i]==k[i][0]*3-1) )
          {
                  for(j=1;j<=k[i][0];j++)
```

```
if (a[i][k[i][j]]==2)
{
        for(z=1;z<=k[i][0];z++)
        {
          a[i][k[i][z]]--;
          a[k[i][z]][i]--;
          if (vcset[k[i][z]]==1) d[k[i][z]]--;
        }

                vcset[i]=0;
                d[i]=0;

                vcset[k[i][j]]=1;
                m=k[i][j];

         for(z=1;z<=k[m][0];z++)
         {
          a[m][k[m][z]]++;
          a[k[m][z]][m]++;
          if (vcset[k[m][z]]==1) d[k[m][z]]++;
                  d[m]=d[m]+a[m][k[m][z]];
                }

                ii1=0;
                for(z=1;z<=n;z++)
                  for(j=z+1;j<=n;j++)
                                  if (a[z][j]==3) ii1++;

 if (ii1>ii)
 {
                goto tkrr;
 }

 for(z=1;z<=k[i][0];z++)
 {
```

```
                    a[i][k[i][z]]++;
                    a[k[i][z]][i]++;
                    if (vcset[k[i][z]]==1) d[k[i][z]]++;
                    d[i]=d[i]+a[i][k[i][z]];
                }

                vcset[i]=1;

                for(z=1;z<=k[m][0];z++)
                {
                    a[m][k[m][z]]--;
                    a[k[m][z]][m]--;
                    if (vcset[k[m][z]]==1) d[k[m][z]]--;
                }

                vcset[m]=0;
                d[m]=0;

                ii=0;
                for(z=1;z<=n;z++)
                    for(j=z+1;j<=n;j++)
                        if (a[z][j]==3) ii++;
                }
        }

        for(i=1;i<=n;i++)
            if ( (vcset[i]==1) && (d[i]==k[i][0]*3-2) )
            {
                for(j=1;j<=k[i][0];j++)
                    if ((vcset[j]==1) && (i!=j) && (a[i][j]==0) &&
(d[j]==k[j][0]*3-2) )
                    {
                        m=0;
                        for(z=1;z<=k[i][0];z++)
                            if (a[i][k[i][z]]==2)
```

```
                {
                        m++;
                        if (m==1) s1=k[i][z];
                        if (m==2) s2=k[i][z];
                }

        m=0;
        for(z=1;z<=k[j][0];z++)
                if (a[j][k[j][z]]==2)
                {
                        m++;
                        if (m==1) ss1=k[j][z];
                        if (m==2) ss2=k[j][z];
                }


        if      ((s1==ss1)      &&      (s2==ss2)      &&
(aa[s1][s2]==0))
        {
                aa[s1][s2]=1;

                vcset[i]=0;
                for(jj=1;jj<=k[i][0];jj++)
                {
                        a[i][k[i][jj]]--;
                        a[k[i][jj]][i]--;
                        if              (vcset[k[i][jj]]==1)
d[k[i][jj]]--;
                }

                vcset[j]=0;
                for(jj=1;jj<=k[j][0];jj++)
                {
                        a[j][k[j][jj]]--;
                        a[k[j][jj]][j]--;
```

```
                                        if          (vcset[k[j][jj]]==1)
d[k[j][jj]]--;
                                    }

                                    vcset[s1]=1;
                                    for(jj=1;jj<=k[s1][0];jj++)
                                    {
                                        a[s1][k[s1][jj]]++;
                                        a[k[s1][jj]][s1]++;
                                        if          (vcset[k[s1][jj]]==1)
d[k[s1][jj]]++;
                                        d[s1]=d[s1]+a[s1][k[s1][jj]];
                                    }

                                    vcset[s2]=1;
                                    for(jj=1;jj<=k[s2][0];jj++)
                                    {
                                        a[s2][k[s2][jj]]++;
                                        a[k[s2][jj]][s2]++;
                                        if          (vcset[k[s2][jj]]==1)
d[k[s2][jj]]++;
                                        d[s2]=d[s2]+a[s2][k[s2][jj]];
                                    }

                                    goto tkrr;
                                }
                            }
                        }

        iter++;

        ti++;

        if (iter==1)
        {
          for(i=0;i<=n;i++)
```

```
  {
        svc[i]=vcset[i];
        use[i]=0;
  }
  bestvc[0]=n;
  tsayi=n-vcset[0];
}

if (bestvc[0]>vcset[0])
{
  for(i=0;i<=n;i++)
  {
        bestvc[i]=vcset[i];
  }
}

if (iter!=1)
{

  for(i=0;i<=n;i++)
  {
        vcset[i]=svc[i];
        d[i]=0;
  }

  for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        {
              if (a[i][j]!=0) a[i][j]=1;
        }

  for(i=1;i<=n;i++)
  {
        if (vcset[i]==1)
        {
```

```
                for(j=1;j<=k[i][0];j++)
                {
                        a[i] [k[i][j]]++;
                        a[k[i][j]] [i]++;
                        aa[i][j]=0;
                }
        }
  }

  for(i=1;i<=n;i++)
  {
        if (vcset[i]==1)
        {
                for(j=1;j<=k[i][0];j++)
                        d[i]=d[i]+a[i][k[i][j]];
        }

  }

  for(i=0;i<=n;i++)
  {
        erase[i]=0;
        add[i]=0;
        use[i]=0;
  }
}

s1=0;
for(i=1;i<=n;i++)
{
  if (vcset[i]==0)
  {
        s1++;
        if (s1==ti) hangi=i;
  }
```

```
}

erase[0]++;
erase[erase[0]]=hangi;
use[hangi]=1;

for(i=1;i<=k[hangi][0];i++)
{
  s1=1;
  for(j=1;j<=add[0];j++)
  {
            if (a[add[j]][k[hangi][i]]!=0)
            {
                      s1=0;
            }
  }

  if (s1==1)
  {
            add[0]++;
            add[add[0]]=k[hangi][i];
  }
}

for(i=1;i<=add[0];i++)
{
  s1=add[i];
  for(j=1;j<=k[s1][0];j++)
  {
            if ((vcset[k[s1][j]]==0) && (use[k[s1][j]]==0))
            {
                      use[k[s1][j]]=1;
                      erase[0]++;
                      erase[erase[0]]=k[s1][j];
            }
```

```
  }
}

for(i=1;i<=add[0];i++)
{
  s1=add[i];
  vcset[s1]=0;
  for(jj=1;jj<=k[s1][0];jj++)
  {
                    a[s1][k[s1][jj]]--;
                    a[k[s1][jj]][s1]--;
                    if (vcset[k[s1][jj]]==1) d[k[s1][jj]]--;

  }
}

for(i=1;i<=erase[0];i++)
{
  s1=erase[i];
  vcset[s1]=1;
  for(jj=1;jj<=k[s1][0];jj++)
  {
          a[s1][k[s1][jj]]++;
          a[k[s1][jj]][s1]++;
          if (vcset[k[s1][jj]]==1) d[k[s1][jj]]++;
          d[s1]=d[s1]+a[s1][k[s1][jj]];
  }
}

vcset[0]=vcset[0]+erase[0]-add[0];

if (iter<=tsayi) goto tkrr;

end = clock();
double taym;
```

```
        printf("\n Son Cozum VC =  %d\n",bestvc[0]);
        taym = (double)(end - start) / CLOCKS_PER_SEC;
        printf( "%10.6f seconds\n", taym );

return 0;

}
```