DOKUZ EYLÜL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GENETİK ALGORİTMA İLE MAKSİMUM BAĞIMSIZ KÜME PROBLEMİNİN ÇÖZÜMÜ

Mehmet GENCER

Ağustos, 2017
İZMİR

GENETİK ALGORİTMA İLE MAKSİMUM BAĞIMSIZ KÜME PROBLEMİNİN ÇÖZÜMÜ

Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi
Bilgisayar Bilimleri Ana Bilim Dalı

Mehmet GENCER

Ağustos, 2017
İZMİR

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

MEHMET GENCER, tarafından **DOÇ.DR. MURAT ERŞEN BERBERLER** yönetiminde hazırlanan **"GENETİK ALGORİTMA İLE MAKSİMUM BAĞIMSIZ KÜME PROBLEMİNİN ÇÖZÜMÜ""** başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

Doç.Dr. Murat Erşen BERBERLER

Yönetici

Prof.Dr. Efendi Nasiboğlu

Jüri Üyesi

Yrd.Doç.Dr. Refet POLAT

Jüri Üyesi

Prof.Dr. Emine İlknur CÖCEN

Müdür

Fen Bilimleri Enstitüsü

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, iki yıl boyunca değerli bilgilerini benimle paylaşan, kıymetli zamanını ayırıp sabırla ve büyük bir ilgiyle bana faydalı olabilmek için elinden gelenden fazlasını yapan her sorun yaşadığımda yanına çekinmeden gidebildiğim, güler yüzünü ve samimiyetini benden esirgemeyen saygıdeğer danışman hocam Doç. Dr. Murat Erşen BERBERLER'e ve hayatımım her anında bana maddi manevi destek olan aileme teşekkür ederim.

Mehmet GENCER

GENETİK ALGORİTMA İLE MAKSİMUM BAĞIMSIZ KÜME PROBLEMİNİN ÇÖZÜMÜ

ÖZ

NP-Zor karmaşıklık sınıfına ait çizge problemlerinden biri olan Maksimum Bağımsız Küme (MIS) ile günlük hayatta sıklıkla görüntü işleme, harita işaretleme, moleküler biyoloji, zaman çizelgesi planlama(scheduling), vs. gibi alanlarda sıklıkla karşılaşılır. Bu çalışmada MIS problemine birden fazla uygun çözümün aynı anda oluşturulup kontrol edilebildiği meta sezgisel yöntemlerden birisi olan genetik algoritmalar yardımıyla kısa sürelerde optimuma yakın kalitede çözümler aranmıştır. Literatürdeki çalışmaların büyük bölümünden farklı olarak genetik algoritmanın başlangıç popülasyonu, rastgele belirlenmeyip çeşitli sezgisel yaklaşımlarla oluşturulmuştur. Farklı sezgisel yaklaşımlar ve bu yaklaşımların optimum değer bulamadığı ters örnekler incelenmiş, sezgisel yaklaşımların çözüme ulaşma olasılığı kaydırma operatörü yardımıyla farklı çözümler üretilerek arttırılmıştır. Burada kaydırma operatörü ile kastedilen yapı tepe derecelerine göre sıralı vektör üzerinde farklı çözümleri elde edebilmek için permütasyon uzayının küçük bir alt kümesinin seçilmesidir. Ayrıca çözüm uzayını etkin bir şekilde tarayabilmek amacıyla problemin boyutuna bağlı olarak büyüklüğü değişen başlangıç popülasyonu kullanılmıştır. Algoritma C dilinde kodlanmış olup farklı ayrıt yoğunluklarına sahip rastgele oluşturulmuş problemler üzerinde hesaplama denemeleri yapılmıştır. Çalışma zamanı ve amaç fonksiyonunun değerleri açısından algoritmanın verimli olduğu gözlenmiştir. Oluşturulan problemler klasik genetik algoritma ile de çözülerek, başarım oranları ve çözüm zamanları incelenmiş, sezgisel yaklaşımın etkisini gözlemlemek amacıyla bir biri ile karşılaştırılmıştır.

Anahtar kelimeler: Optimizasyon, Meta sezgisel algoritmalar, Genetik algoritma, Maksimum bağımsız küme problemi

SOLUTION TO THE MAXIMUM INDEPENDENT SET PROBLEM WITH GENETIC ALGORITHM

ABSTRACT

NP-hard graph problems that belong to complexity class which is one of Maximum independent set (MIS) in daily life, often with image processing, map marking, Molecular Biology, planning timeline(scheduling), etc in areas such as often is encountered. In this study, with the help of genetic algorithms which is a meta-heuristic solution approach in which more than one suitable solution of artificial intelligence techniques can be created and controlled at the same time, the solutions to the Maximum Independent Set were searched in a short period of time. Unlike most of the studies in the literature, the initial population of the genetic algorithm has not been determined at random and has been created with various heuristic approaches. Different heuristic approaches and their inverse examples in which these approaches are unable to find the optimum value have been examined, a heuristic approach designed in this process to increase the possibility of reaching to the best solution by the shifting operator, creating different solutions is ensured. Here on the scroll meant by sequential vector building operator in order to obtain different solutions according to the degree peak, is to select a small subset of the permutation space. In order to scan the solution space effectively depending on size of the problem varying size of the initial population was used. The algorithm is coded in the C language and the computational experiments on randomly produced problems in terms of runtime and objective function values have been performed and it was observed that the algorithm is efficient. Problems were solved with classical genetic algorithm, performance ratios and solution times were examined and compared with heuristic one to observe the effect of the heuristic approach.

Keywords: Optimization, Meta-heuristic algorithms, Genetic algorithms, Maximum independent set problem

İÇİNDEKİLER

Sayfa

SINAV SONUÇ FORMU ii
TEŞEKKÜR iii
ÖZ iv
ABSTRACT v
ŞEKİLLER LİSTESİ ix
TABLOLAR LİSTESİ x

BÖLÜM BİR – GİRİŞ 1

1.1 Problemin Matematiksel Modeli 1
1.2 Algoritma Karmaşıklığı, P ve NP Sınıflar 2
1.2.1 Problem, Algoritma ve Algoritmaların Karmaşıklığı 2
1.2.2 Asimptotik Notasyonlar 3
1.2.2.1 O Notasyonu 3
1.2.2.2 Ω Notasyonu 3
1.2.2.3 Θ Notasyonu 3
1.2.3 P ve NP Sınıflar, NP-tam Problemler 3
1.2.4 MIS Probleminin NP-Tamlığının İspatı 5

BÖLÜM İKİ – PROBLEMİN ÇÖZÜM YÖNTEMLERİ 6

2.1 Tam Çözüm Yöntemleri 6
2.1.1 Sayımlama 6
2.1.2 Matematiksel Modeli Baz Alan Optimum Garantili Çözüm Yöntemi 9
2.1.2.1 CPLEX 9
2.1.2.2 GAMS-CPLEX Hesaplama Denemeleri 11
2.1.3 GAMS-CPLEX ve Sayımlama Yöntemlerinin Karşılaştırılması 12
2.2 Sezgisel Çözüm Yaklaşımları 13
2.2.1 Tepe Dereceleri Metodu 13

2.2.2 MAX Algoritması 14
2.2.3 Komşu Tepe Dereceleri Toplamı Metodu 15
2.2.4 Problemin Tümleyen Problemi Üzerinden Çözüm Aranması 16
2.3 Kaydırma Yöntemi (Turlama Tekniği) 17
2.4 Meta-Sezgisel Çözüm Yaklaşımları: Genetik Algoritmalar 19

BÖLÜM ÜÇ – GENETİK ALGORİTMALAR 20

3.1 Genetik Algoritmaların Avantajları ve Sınırlılıkları 22
3.2 Kodlama 23
3.2.1 İkili düzende kodlama 24
3.2.2 Permütasyon (sıralı) kodlama 24
3.2.3 Değer kodlaması 24
3.3 Uygunluk Fonksiyonu 25
3.4 Başlangıç Populasyonunun Oluşturulması 25
3.5 Çoğalma operatörü 26
3.5.1 Rulet tekerleği 26
3.5.2 Turnuva Yöntemi 26
3.5.3 Sıralama Yöntemi 27
3.6 Çaprazlama Operatörü 27
3.6.1 Tek noktalı çaprazlama 27
3.6.2 İki noktalı çaprazlama 28
3.6.3 Tekdüze çaprazlama 28
3.7 Mutasyon Operatörü 28

BÖLÜM DÖRT – GENETİK ALGORİTMALAR İLE MAKSİMUM BAĞIMSIZ KÜME PROBLEMİNİN ÇÖZÜMÜ 29

4.1 Çözümün Kodlanması 29
4.2 Büyük Boyutlu Problemlerin Çözümünde Kullanılan Veri Yapıları 29
4.3 Başlangıç Popülasyonunun Oluşturulması 31
4.4 Kullanılan Çoğalma Yöntemleri 32
4.5 Kullanılan Çaprazlama Yöntemi 32

4.6 Kullanılan Mutasyon Yöntemi 33
4.7 Kullanılan Parametre Değerleri 33

BÖLÜM BEŞ – HESAPLAMA DENEMELERİ 34

5.1 GAMS-CPLEX Hesaplama denemeleri 34
5.2 Yoğun Çizgeler İçin Tam Çözüm Algoritması Hesaplama Denemeleri 36
5.3 Başlangıç Popülasyonu Sezgisel Yöntemle Oluşturulan GA için Hesaplama Denemeleri 38
5.4 Klasik Genetik Algoritmalar Hesaplama Denemeleri 40
5.5 Genetik Algoritmaların Bağıl Hata Tablolarına Göre Karşılaştırılması 42
5.6 Düşük Yoğunluklu Ek Problemler 44

BÖLÜM ALTI – SONUÇLAR 47

KAYNAKLAR 49

EKLER 53

EK 1: Metin Dosyası Şeklinde Problem Üreten Pascal Kodu 53
EK 2: Sayımlama C Kodu 55
EK 3: Metin Dosyasında Verilen Problemi GAMS Formatına Çeviren Pascal Kodu 58
EK 4: Yoğun Çizgeler İçin Tam Çözüm Algoritması C Kodu 60
EK 5: Başlangıç Populasyonu Rastgele Üretilen GA C Kodu 64
EK 6: Sezgisel Yöntemle Oluşturulan GA C Kodu 70
EK 7: Optimum Değerleri Bilinen Seyrek Problemler Üreten Pascal Kodu 81

ŞEKİLLER LİSTESİ

Sayfa

Şekil 2.1 Sayımlama için Hesaplama Denemeleri 7
Şekil 2.2 CPLEX Çözücüsü Problem Örneği 10
Şekil 2.3 GAMS-CPLEX Hesaplama Denemelerinde Elde Edilen Çalışma Süreleri 12
Şekil 2.4 GAMS ve Sayımlama Hesaplama Denemeleri Kıyası 12
Şekil 2.5 Tepe Dereceleri Metodu için Ters Örnek 14
Şekil 2.6 MAX Algoritması için Ters Örnek 14
Şekil 2.7 Komşu Tepe Dereceleri Toplamı Metodu için Ters Örnek 15
Şekil 2.8 Problemin Tümleyen Problemi için Ters Örnek 17
Şekil 2.9 Örnek Problemi Tümleyen Problem 17
Şekil 2.10 Turlama Tekniği Görselleştirme Örneği 18
Şekil 4.1 Bitişiklik Matrisi 30
Şekil 4.2 Bitişiklik Dizisi 30
Şekil 4.3 Bitişiklik Matrisi Alt Üçgen 30
Şekil 4.4 Bitişiklik Matrisi Dönüşüm Adımı 31

TABLOLAR LİSTESİ

Sayfa

Tablo 2.1 Sayımlama için Hesaplama Denemeleri 8
Tablo 2.2 GAMS-CPLEX Hesaplama Denemelerinde Elde Edilen Çalışma Süreleri 11
Tablo 5.1 GAMS Süreler 1 (sn) 35
Tablo 5.2 GAMS Süreler 2 (sn) 35
Tablo 5.3 GAMS Optimum Değerler 35
Tablo 5.4 Optimum Çözüm Üreten Algoritma Sonuçlar 36
Tablo 5.5 Optimum Çözüm Üreten Algoritma Süreler 1 37
Tablo 5.6 Optimum Çözüm Üreten Algoritma Süreler 2 37
Tablo 5.7 Problemlere Ait Bulunan Optimum Sonuçlar 37
Tablo 5.8 Genetik Algoritmalar Minimum Değerler 38
Tablo 5.9 Genetik Algoritmalar Maximum Değerler 38
Tablo 5.10 Genetik Algoritmalar Ortalama Değerler 1 39
Tablo 5.11 Genetik Algoritmalar Ortalama Değerler 2 39
Tablo 5.12 Genetik Algoritmalar Ortalama Süreler 1 40
Tablo 5.13 Genetik Algoritmalar Ortalama Süreler 2 40
Tablo 5.14 Klasik Genetik Algoritmalar Minimum Değerler 41
Tablo 5.15 Klasik Genetik Algoritmalar Maksimum Değerler 41
Tablo 5.16 Klasik Genetik Algoritmalar Ortalama Değerler 42
Tablo 5.17 Klasik Genetik Algoritmalar Ortalama Değerler 42
Tablo 5.18 Klasik Genetik Algoritmalar Ortalama Süreler 1 43
Tablo 5.19 Klasik Genetik Algoritmalar Ortalama Süreler 2 43
Tablo 5.20 Sezgisel Yöntemle Başlatılan GA Bağıl Hata 1 44
Tablo 5.21 Sezgisel Yöntemle Başlatılan GA Bağıl Hata 2 44
Tablo 5.22 Klasik Genetik Algoritmalar Bağıl Hata 1 45
Tablo 5.23 Klasik Genetik Algoritmalar Bağıl Hata 2 45
Tablo 5.24 Düşük Yoğunluklu Ek Problemler GA Ortalama Süreler 46
Tablo 5.25 Düşük Yoğunluklu Ek Problemler Optimum Değerler 46

BÖLÜM BİR
GİRİŞ

Basit bir G = (V, E) çizgesinde bağımsız küme, bu kümeye ait herhangi iki eleman (tepe) arasında bağ (ayrıt) olmaması şeklinde tanımlanır. Bağımsız küme, Korshunov (1974) tarafından "internal stable set" olarak isimlendirilmiş olsa da tepe kutulama olarak da bilinen NP-Zor tipi optimizasyon problemidir. Bağımsız küme (Independent Set), verilen çizge içinde seçilen bir alt kümedeki elemanların (tepelerin) komşuluk ilişkisinin olmama durumu ile açıklanır. En fazla eleman (tepe) sayısına sahip olan bağımsız küme MIS (Maksimum Bağımsız Küme) olarak adlandırılır. Bu kümenin eleman (tepe) sayısına bağımsızlık sayısı denir ve α(G) ile gösterilir. Maksimum elemana sahip olan bağımsız kümeyi bulma işlemi, verilen çizgenin problem kısıtlarına uyan bütün alt küme seçimlerinin incelenmesini gerektirir. Optimizasyon problemlerinden MIS birçok uygulama alanında karşılık bulmaktadır, örneğin; sinyal işleme analizi, sınıflandırma teorisi, ekonomi, zaman çizelgeleri, deneysel tasarım, bilgisayar görüsü vb. (Butenko, 2003)

1.1 Problemin Matematiksel Modeli

V ile gösterilen tepeler kümesindeki elemanların sayısı n olsun, (|V|=n).
Maksimum Bağımsız Küme (Maximum Independent Set, MIS)

$$max f(x) = \sum_{i=1}^{n} x_i \tag{1.1}$$

öyle ki

$$x_i + x_j \leq 1, \quad \forall (x_i, x_j) \in E \tag{1.2}$$

ve

$$x_i \in \{0, 1\}, \quad i = 1, \ldots, n. \tag{1.3}$$

1.2 Algoritma Karmaşıklığı, P ve NP Sınıflar

1.2.1 Problem, Algoritma ve Algoritmaların Karmaşıklığı

Algoritmalar ardı ardına gelen işlemlerle problem çözme yollarıdır. Problem yanıt isteyen genel bir soru olarak kabul edilebilir. Soru tanımı içinde,

- soruya ilişkin tüm parametrelerin belirlenmesi
- yanıtın (çözümün) taşıması gereken tüm özellikler

olmalıdır.

Parametrelere özgün değerler atandığında problemin bir örneğini elde ederiz. Eğer problemin örneğini I ile gösterirsek, I'ların oluşturduğu kümeye $\Pi = \{I\}$ problem denir.

Örneğin; $\Pi : ax^2 + bx + c = 0$ genel 2. derece denklemin yazım şeklidir, burada a, b, c parametrelerine özgün değerler verilirse $I : 3x^2 - 4x + 7 = 0$ örnek problemi elde edilir.

İyi kötü algoritma ayrımının testi zaman ve bellek karmaşıklığına (time complexity, space complexity) dayandırılmaktadır. Eğer bir algoritmanın zaman karmaşıklığı büyüklük parametrelerinin bir polinomu ise iyi kabul edilir. Aksi halde, yani polinom sınırlı olmayan zaman karmaşıklığına sahip algoritmalar kötü kabul edilir. Bu tür algoritmalara üstel sınırlı algoritmalar denir.

Zaman karmaşıklığına benzer biçimde, hesaplamalar için gerekli bellek miktarı problemin büyüklük parametrelerinin bir fonksiyonu olarak yazılabilir ve bellek karmaşıklığı (space complexity) olarak isimlendirilir. (McConnell, 2001)

1.2.2 Asimptotik Notasyonlar

Asimptotik notasyonlar algoritmaların karmaşıklığını göstermek için kullanılan tanımlamalardır. En çok kullanılan 3 tanesi şunlardır. (Çölkesen, 2007)

1.2.2.1 O Notasyonu

Bağıntıyı üstten sınırlar; bu sınırlama $n > n_0$ için k çarpanı ile yapılır. Eğer $\exists n_0, k > 0$ varsa ve bağıntı $k.g(n)$'in altında kalıyorsa yani $f(n) \leq k.g(n)$ oluyorsa $f(n) = O(g(n))$ şeklinde gösterilir.

1.2.2.2 Ω Notasyonu

Bağıntıyı alttan sınırlar; bu sınırlama işlemi $n > n_0$ için k çarpanı ile yapılır. Eğer $\exists n_0, k > 0$ varsa ve bağıntı $k.g(n)$'in üstünde kalıyorsa yani $f(n) \geqslant k.g(n)$ oluyorsa $f(n) = \Omega(g(n))$ şeklinde gösterilir.

1.2.2.3 Θ Notasyonu

Bağıntıyı alttan ve üstten sınırlar; bu sınırlama $n > n_0$ için k_1 ve k_2 çarpanları ile yapılır. Eğer $\exists n_0, k_1, k_2 > 0$ varsa ve bağıntı $k_1.g(n)$ ile $k_2.g(n)$ fonksiyonları arasında kalıyorsa yani $k_1.g(n) \geqslant f(n) \geqslant k_2.g(n)$ oluyorsa $f(n) = \Theta(g(n))$ şeklinde gösterilir.

1.2.3 P ve NP Sınıflar, NP-tam Problemler

Algoritma kavramını formal olarak tanımlamak için hesaplama süresinin belirli bir modelinin seçilmesi gerekir. Bu model tek bantlı Deterministik Turing Makinesi (Deterministic Turing Machine) (DTM) olarak ele alınabilir. DTM sonsuz uzun bant, idare kısmı ve okuma-yazma yapan başlıktan oluşur. Eğer M programı için DTM

problemin bant üzerine yazılmış verilerinden oluşturulmuş kelimeleri baz alarak sonlu adımdan sonra durursa M programı Tanıma Problemini (Decision Problem, yalnızca "evet" veya "hayır" şeklinde iki yanıtı olan problemler) çözer denir. Öte yandan Deterministik Olmayan Turing Makinesi (Non Deterministic Turing Machine) (NDTM) ise DTM'den farklı olarak bulucu modülü içerir. NDTM çalışırken ilk etapta bulucu modül bant üzerinde bir çözüm önerir ve sonra bu çözümün tanıma probleminin koşullarına uygunluğunu kontrol eder. (Arora ve Safra, 1998)

Polinom sınırlı DTM ile çözülebilen tüm karar problemleri P sınıfını (Polynomial) oluşturur. Polinom sınırlı NDTM ile çözülebilen karar problemleri ise NP sınıfını (Nondeterministic Polynomial) oluşturur. Açıktır ki $P \subset NP$'dir. Çünkü NDTM polinom sınırlı DTM ile çözülebilen her problemi çözebilir. $P \in \Pi \Rightarrow NP \in \Pi$ olduğundan eğer A, Π problemini çözen algoritma ise o zaman A algoritması kontrol etabında kullanılarak deterministik olmayan bir algoritma ile Π problemi çözülebilir. Asıl sorun polinom sınırlı NDTM ile çözülebilen karar probleminin DTM ile polinom sınırlı zamanda çözülüp çözülemeyeceğidir. Yani $NP \stackrel{?}{\subset} P$ olduğu gösterilebilir mi, eğer öyleyse P = NP olacağından P $\stackrel{?}{=}$ NP sorusunun yanıtı bulunmuş olur. Çünkü eğer bu eşitlik varsa zor problemler sınıfındaki tüm problemlerin giriş verilerinin boyutuna bağlı olarak polinom zamanda çözülebileceği anlamına gelir. (Arora ve diğer., 1998)

Eğer NP sınıfındaki her problem Π problemine indirgenebiliyorsa Π problemi NP-zor (NP-hard) adını alır. Aynı zamanda Π'in kendisi de $\Pi \in NP$ ise NP-tam (NP-complete) adını alır. (Papadimitriou ve Steiglitz, 1982)

Bir Π problemin NP-tam'lığının ispatında aşağıdaki prosedür izlenir:

1. $\Pi \stackrel{?}{\in} NP$,
2. $\exists \Pi' \in NP - tam$,
3. $f : \Pi' \rightarrow \Pi$,
4. f'nin polinom zamanda hesaplanabileceği gösterilir.

1.2.4 MIS Probleminin NP-Tamlığının İspatı

Bir G çizgesi için maksimum bağımsız küme problemine ait çözüm MIS ile verilmiştir.

MIS = {<G, k> | G kümesine ait k eleman sayısına sahip maksimum bağımsız küme}

- İlk olarak çözümün polinom zamanda doğrulanabileceği gösterilerek MIS ∈ NP olduğu ispatlanmalıdır. MIS, P noktalar kümesinden oluşur. Bu küme için |P|=k olduğu doğrulanmalıdır. Sonrasında seçilen her u, v ∈ P ikilisi için (u, v)'nin G kümesinin bir ayrıtı olmadığı doğrulanmalıdır. Bu işlem P^2 zaman alır. Sonuç olarak MIS, polinom zamanda doğrulanmıştır. MIS ∈ NP.

- İkinci olarak MIS ∈ NP-Zor olduğu gösterilmelidir. Bu işlem için NP-Tam olduğu bilinen CLIQUE probleminin MIS problemine indirgenebildiği gösterilmelidir. (CLIQUE < p MIS). Herhangi bir CLIQUE problemini MIS problemine <G, k> ∈ CLIQUE ancak ve ancak f(<G, k>) ∈ MIS ile indirgeyen polinom zamanlı bir f dönüşümünün varlığı gösterilmelidir. f(<G, k>) = <G', k> dönüşümünde G', G çizgesinin tümleyenini göstermektedir. (u, v) ∈ G' ayrıtı ancak ve ancak (u, v) ∉ G olmalıdır. G' çizgesi, G çizgesi kullanılarak, V, G çizgesinin ayrıt sayısı olmak üzere, O(V^2) zamanda elde edilir. Eğer G çizgesi k boyutunda bir clique içeriyorsa, bu clique içindeki tepeler G' çizgesinde birbiri ile bağı olmayan tepeler olacaktır. Yani G' çizgesi için bu tepeler k boyutlu bağımsız bir küme oluşturur. Bu nedenle NP-Tam CLIQUE problemi polinom zamanda MIS problemine indirgenebilmiştir. Bu da MIS ∈ NP-Zor olduğunu gösterir.

- Sonuç olarak, MIS ∈ NP ve MIS ∈ NP-Zor, olduğu için MIS problemi NP-Tamdır.

BÖLÜM İKİ
PROBLEMİN ÇÖZÜM YÖNTEMLERİ

Maksimum bağımsız küme problemine bir çok farklı yöntem ile çözüm aranmıştır. Bu bölümde probleme ait çözüm yöntemlerinden tam çözüm yöntemleri, sezgisel yöntemler ve meta-sezgisel çözüm yöntemlerine örnekler verilecektir.

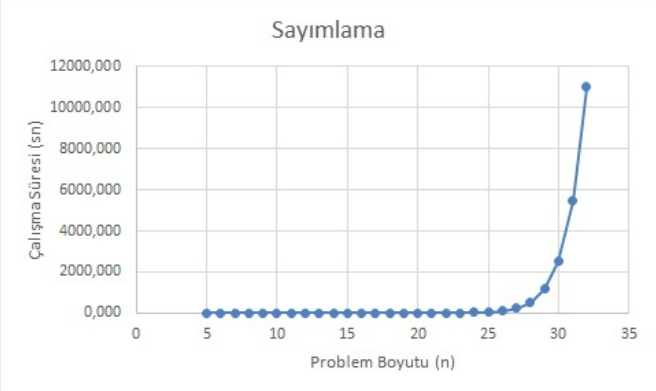
2.1 Tam Çözüm Yöntemleri

Problemlerin kesin (optimum) değerlerini bulmaya yarayan yöntemlerdir. Kesin çözüm yöntemleri, en iyi çözümü bulmayı garanti eder. Ancak, çalışma zamanı problemin boyutu ile birlikte çarpıcı biçimde artmaktadır. Genelde küçük ve orta boyutlu problemler pratik ve ispatlanabilir bir biçimde çözülebilmektedir. Bütün en iyileme problemlerini başarıyla çözen kesin çözüm yöntemi yoktur.

2.1.1 Sayımlama

Sayımlama tüm durumları saymaya dayanan bir yöntemdir. Bulunan tüm durumlara karşılık gelen amaç fonksiyonu değerleri hesaplanır ve bunlar arasında en iyi değere sahip durum çözüm olarak seçilir. (Loukakis, 1983) Maksimum Bağımsız Küme probleminin sayımlama yöntemi ile optimum çözümünün bulunması için alt küme inşası metodu kullanılmıştır. n tepe sayısını göstermek üzere boş küme hariç $2^n - 1$ adet alt küme, dizi veri yapısı ve 1 bitlik toplama benzetimi işlemi kullanılarak oluşturulmuştur. Örneğin n=3 seçildiğinde dizi [0,0,0] başlangıç pozisyonundan, ilk adımda elde ”1” alınarak ve her adımda ”1” eklenerek [1,1,1] durumuna kadar işlem devam eder. ”1” ekleme ile toplama işleminin manipüle edildiği kastedilmektedir, şöyle ki; ilk adımda [0+(elde=1),0,0] yapılarak [1,0,0] durumu ve ardından [1+(elde=1),0,0] yapılarak [0,1,0] durumu elde edilir. Burada elde 1+1=0 ve elde=1 alınarak 1 bitlik toplama işlemi simüle edilmektedir. (Loukakis ve Tsouros, 1981) n=5..32 arasında tepeler için ayrıt sayısı ⁒50 yoğunluğunda rastgele çizgeler

oluşturulmuş ve sayımlama yöntemiyle çözümleri ve çalışma süreleri Tablo 2.1 ile listelenmiştir. Ayrıca bu işlemin yapılmasında kullanılan C programı ekler bölümünde yer almaktadır. Bu yöntem değişken sayısı az olan problemler için kullanışlı olmasına karşın, değişken sayısı arttıkça ele alınan durumların sayısı üstel biçimde artar. Örneğin n boyutlu bir çizge için maksimum bağımsız küme probleminde; her bir tepenin çözüme alınıp alınmama durumuna bağlı olarak incelenmesi gereken durum sayısı 2^n'dir. Yani n değeri arttıkça işlem yoğunluğu artar ve büyük n değerleri için şu andaki mevcut bilgisayarlarla çözüm süresi yüz yıllar alabilir. Örneğin n=64 olduğunda 3,2 GHz hızında bir CPU'da yaklaşık olarak 182,79 yıl sürmektedir. Çözdürülen problemlere ait süreler Tablo 2.1 ve Şekil 2.1'de inclendiğinde bu artış net bir şekilde görülebilmektedir. (Alekseev ve Lozin, 2004)



Şekil 2.1 Sayımlama için Hesaplama Denemeleri

Tablo 2.1 Sayımlama için Hesaplama Denemeleri

n	Opt	Süre(sn)
5	3	0.000
6	3	0.000
7	3	0.000
8	4	0.000
9	3	0.000
10	4	0.000
11	4	0.000
12	5	0.000
13	5	0.000
14	4	0.000
15	4	0.000
16	5	0.015
17	5	0.078
18	5	0.177
19	6	0.328
20	5	0.812
21	5	1.890
22	6	4.954
23	6	8.547
24	5	23.373
25	6	49.175
26	6	117.105
27	7	248.987
28	6	512.578
29	6	118.,859
30	6	2507.264
31	7	5446.173
32	7	10998.956

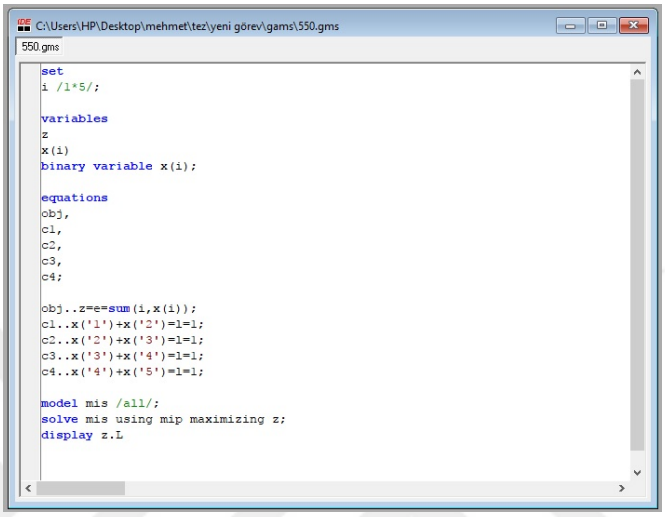
2.1.2 Matematiksel Modeli Baz Alan Optimum Garantili Çözüm Yöntemi

Maksimum bağımsız küme probleminin çözümü için, problemin matematiksel modelini baz alan optimum garantili çözüm yöntemini kullanmak icin GAMS yazılımı kullanılmıştır. Genel Cebirsel Modelleme Sistemi (General Algebraic Modeling System (GAMS)), matematiksel programlama ve optimizasyon için kullanılan üst düzey bir modelleme sistemidir. GAMS bir derleyici ve entegre edilmiş bir yüksek performanslı çözücüden oluşur. Bu sistem karmaşık, büyük ölçekli modelleme problemleri için tasarlanmıştır. Kullanıcıya modelleri inceleme ve modeller üzerinde değişiklik yaparak yeni durumlara adapte edebilme olanağı sağlar. GAMS ilk olarak optimizasyon uzmanları Dr. Anthony Brooke ve Dr. Alexander Meeraus tarafından dünya bankası için geliştirilmiştir. Programın en temel özelliği güçlü bir çözücü olmasının yanı sıra kullanım kolaylığı ve esnekliğidir. CPLEX, Lingo, DICOPT gibi çözücü algoritmaları bünyesinde bulundurarak bu çözücüler üzerinden doğrusal, doğrusal olmayan, karma tamsayı problemleri çözebilmekte ve çözüm yöntemini kolaylıkla değiştirebilmektedir. MIS probleminin çözümünde bu çözücü algoritmalardan CPLEX kullanılmıştır.

2.1.2.1 CPLEX

CPLEX; doğrusal programlama, karesel programlama, karesel koşullu programlama ve karma tamsayı programlaması problemlerini çözen esnek ve yüksek performanslı matematiksel programlamaya dayalı çalışan bir çözücüdür. Problemlerin çözümünde dal-sınır yöntemini kullanır. (Danna ve diğer., 2005) Dal-sınır yöntemi optimal çözüme ulaşmak için sistemli bir şekilde uygun çözümleri sayımlamaya dayanır. Sayımlama yöntemindeki gibi tüm durumlar değil, durumların daha küçük bir kısmı incelenir, ümit vermeyen durumlar sayımlama aşamasında kesip atılır. Dal-sınır algoritmasının avantajı hızlı olması, verilmiş her hangi bir tamsayılı problemin özelliklerinin göz önünde bulundurulması ve yöntemin içinde başka tamsayılı metotların kullanılmasına imkan vermesidir. Hesaplama sürecinde ortaya

çıkan çözümler genellikle iyi yaklaşık çözümlerdir. Dezavantajı ise bilgisayar belleğine daha çok gerek duymasıdır. Ayrıca değişken sayısının artması hesaplama süresini üstel olarak artırır ve bulunan çözümün optimal çözüm olduğunu ispatlamak çalışma zamanından daha fazla zaman gerektirebilir. (Papadimitriou ve Steiglitz, 1982)



Şekil 2.2 CPLEX Çözücüsü Problem Örneği

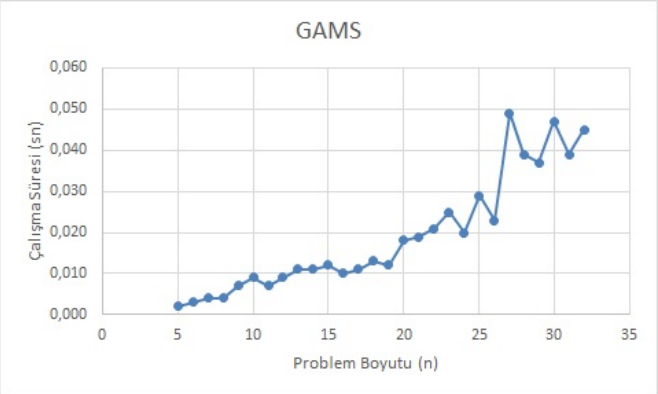
Metin dosyası yani txt formatında oluşturulmuş ve içinde verilen çizgeye ait bitişiklik matrisi bilgisi olan problem dosyaları, pascal dilinde yazılmış olan kod ile GAMS formatına çevrilmiştir. GAMS çözücüsüne ait problem örneği Şekil 2.2'de verilmiştir. Set ile tanımlanan küme problemin boyutu olan n=5 değerine göre tanımlanmıştır. Değişkenler kısmında yer alan z amaç fonksiyonunu, x(i)'ler karar değişkenlerini tutar ve 0-1 değerlerini alabilir. Denklemler kısmında problemin kısıtlarını tutan değişkenler yer almaktadır. Devamında yer alan denklemler ile problemin bitişiklik ilişkisi ifade edilir. Model ifadesi probleme ait olan matematiksel modelin isimin verildiği komut olup, çöz kısmı ise hangi optimizasyon yöntemi ile çözümün yapılacağını gösterir. En alt satırda ise problem çözüldükten sonra raporlama için değişken ve format verilmiştir. (Puchinger ve Raidl, 2005)

2.1.2.2 GAMS-CPLEX Hesaplama Denemeleri

GAMS paketi içinde yer alan ve dal sınır yöntemini kullanan CPLEX çözücüsü Maksimum Bağımsız Küme problemini çözmek için kullanılmıştır. Sayımlama yönteminde karşılaşılan sürelere göre çok daha kısa zamanda çözümlere ulaşılmıştır. Elde edilen çalışma süreleri Tablo 2.1 ile verilmiş ve Şekil 2.3'de görselleştirilmiştir.

Tablo 2.2 GAMS-CPLEX Hesaplama Denemelerinde Elde Edilen Çalışma Süreleri

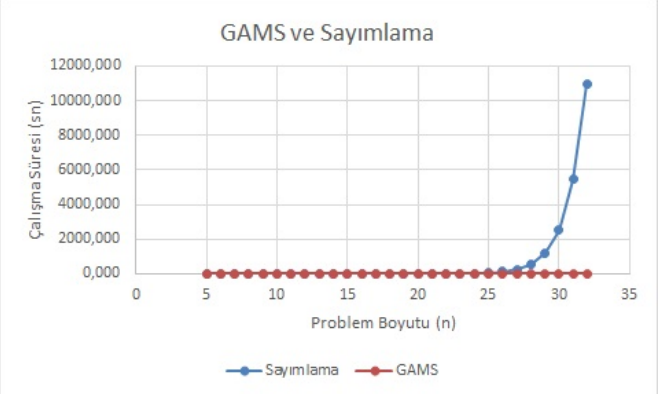
n	Süre(sn)
5	0.002
6	0.003
7	0.004
8	0.004
9	0.007
10	0.009
11	0.007
12	0.009
13	0.011
14	0.011
15	0.012
16	0.010
17	0.011
18	0.013
19	0.012
20	0.018
21	0.019
22	0.021
23	0.025
24	0.020
25	0.029
26	0.023
27	0.049
28	0.039
29	0.037
30	0.047
31	0.039
32	0.045



Şekil 2.3 GAMS-CPLEX Hesaplama Denemelerinde Elde Edilen Çalışma Süreleri

2.1.3 GAMS-CPLEX ve Sayımlama Yöntemlerinin Karşılaştırılması

Sayımlama ve GAMS-CPLEX çözücüsü ile çözdürülen problemlere ait çalışma zamanları kıyaslanmıştır. Bu durumun sebebi CPLEX çözücüsünün dal-sınır yöntemini kullanması ve problem için kullandığı ağaca ait çözüme girmediğinden emin olduğu tepelerden çok hızlı biçimde kurtulabilmesidir. Problemlerin çözdürülmesiyle elde edilen çalışma zamanları Tablo 2.1 ve Tablo 2.2 ile verilmiş ve iki yöntemin çalışma zamanları arasındaki ilişki Şekil 2.4 ile görselleştirilmiştir.



Şekil 2.4 GAMS ve Sayımlama Hesaplama Denemeleri Kıyası

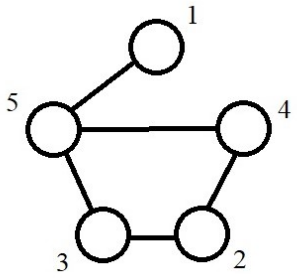
2.2 Sezgisel Çözüm Yaklaşımları

Hızlı olmaları sebebiyle tercih edilen ve optimum garantisi olmayan ancak makul bir hata oranı ile çözümler üretebilen çözüm yaklaşımlarıdır. Aç gözlü sezgisel yaklaşımlar, MIS problemi için polinom zamanda maksimal bağımsız küme bulmak için kullanılabilirler. En çok bilinen dizi temelli sezgisel metotlar MAX, MIN ve VO (Tepe dereceleri dizisi) yöntemleridir. MAX ve MIN yöntemleri dinamik yapıda olup tepe dereceleri her adımda güncellenmektedir. (Boppana ve Halldórsson, 1992) Bu yaklaşımlardan MAX ve VO yaklaşımlarının karşılaştığı (optimum çözüm bulamadığı) ters örnek ile birlikte izleyen bölümde verilecektir. MIS problemi için yeni dizi temelli sezgisel yötemlerde mevcuttur. Ancak bu yaklaşımlar genellikle bazı özel çizge grupları üzerinde makul sonuçlar üretebilmektedir. (Lê ve diğer., 2017)

2.2.1 Tepe Dereceleri Metodu

Tepe dereceleri (VO) algoritmasıyla (Mahadev ve Reed, 1999), G çizgesi içinde yer alan tepeler, derecelerine göre küçükten büyüğe doğru sıralanarak bir dizi oluşturulur. Dizinin işaretlenmemiş ilk elemanı çözüme alınır, çözüme alınan tepe ve o tepeye ait komşu tepeler işaretlenir. Bu işleme çizgede yer alan tüm tepeler işaretlenene kadar devam edilir. Bu algoritma Şekil 2.5 ile verilen, ters örnek olarak karşılaşılan 5 tepeli G çizgesi üzerinde çalıştırılmıştır. Başlangıçta çözüm kümesi (ÇK) ve yasaklı tepeler kümesi (YK) boş küme olarak ayarlanır. G çizgesinin tepe dereceleri sırasıyla [1, 2, 2, 2, 3] olarak hesaplanır ve 1 indisli tepe dizideki en küçük değer olan 1'e sahip olduğu için ÇK={1} çözüme dahil edilir. Bu aşamada 1 tepesine bitişik olan 5 indisli tepe de yasaklı tepe olarak YK=5 işaretlenir. Ardından güncellenmiş [1*, 2, 2, 2, 3*] dizisindeki 2 indisli tepe dizideki en küçük değer olan 2'ye sahip olduğu için ÇK={1, 2} çözüme dahil edilir ve 2 tepesine bitişik olan 3 ve 4 indisli tepeler de yasaklı tepeler olarak YK={3 ,4 ,5} işaretlenir. Tüm tepeler üzerinde işlem yapıldığı için [1*, 2*, 2*, 2*, 3*] algoritma sonlanır ve çözüm kümesinin eleman sayısı 2 algoritmanın sonucu olarak raporlanır. Oysa ki bu örnek

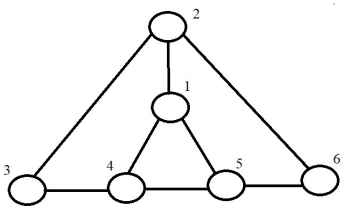
için en iyi çözüm 1, 3 ve 4 tepelerini içeren kümenin eleman sayısı olan 3 olacaktı.



Şekil 2.5 Tepe Dereceleri Metodu için Ters Örnek

2.2.2 MAX Algoritması

MAX algoritması verilen çizgede tepe derecelerine göre çözüm bulan dinamik bir algoritmadır. İlk adımda en büyük dereceye sahip olan tepe ve ayrıtları çizgeden silinir. Tekrar tepe dereceleri sayılır. Atma ve yeniden derece hesaplama işlemine çizge içinde birbirine bağı (ayrıt) olan tepe kalmayana kadar devam edilir. Kalan bağımsız tepeler ise çözüm olarak verilir. (Griggs, 1983) Bu algoritma Şekil 2.6'da



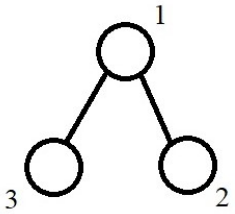
Şekil 2.6 MAX Algoritması için Ters Örnek

verilen, ters örnek olarak karşılaşılan 6 tepeli G çizgesi üzerinde çalıştırılmıştır. Başlangıçta çözüm kümesi (ÇK) ve yasaklı tepeler kümesi (YK) boş küme olarak ayarlanır. G çizgesinin tepe dereceleri sırasıyla [3, 3, 2, 3, 3, 2] olarak hesaplanır ve 1 indisli tepe dizideki en büyük değer olan 3'e sahip olduğu için 1 tepesi çizgeden silinir ve yasaklı tepe YK={1} olarak işaretlenir. Bu aşamada 1 tepesine bitişik olan 2, 4 ve 5 indisli tepelerin dereceleri birer azaltılır. Ardından güncellenmiş [*, 2, 2, 2, 2, 2] dizisindeki 2 indisli tepe dizideki en büyük değer olan 2'ye sahip olduğu için 2 tepesi çizgeden silinir ve yasaklı tepe YK={1 ,2} olarak isaretlenir. 2 tepesine bitişik

olan 3 ve 6 indisli tepelerin dereceleri birer azaltılır. Ardından güncellenmiş [*, *, 1, 2, 2, 1] dizisindeki 4 indisli tepe dizideki en büyük değer olan 2'ye sahip olduğu için 4 tepesi çizgeden silinir ve yasaklı tepe YK={1 ,2 ,4} olarak isaretlenir. 4 tepesine bitişik olan 3 ve 5 indisli tepelerin dereceleri birer azaltılır. Ardından güncellenmiş [*, *, 0, *, 2, 1] dizisindeki 5 indisli tepe dizideki en büyük değer olan 2'ye sahip olduğu için 5 tepesi çizgeden silinir ve yasaklı tepe YK={1 ,2, 3, 4} olarak isaretlenir. 5 tepesine bitişik olan 6 indisli tepelenin derecesi bir azaltılır. Son olarak [*, *, 0, *, *, 0] dizisinde derecesi 0 değerinden farklı olan ve yasaklı olarak işaretlenmeyen tepe kalmadığı için işlem sonlandırılır. 0 değerine sahip tepeler çözüm kümesine alınır ÇK={3, 6}. Çözüm kümesinin eleman sayısı 2 algoritmanın sonucu olarak raporlanır. Oysa ki bu örnek için en iyi çözüm 1, 3 ve 6 tepelerini içeren kümenin eleman sayısı olan 3 olacaktı.

2.2.3 Komşu Tepe Dereceleri Toplamı Metodu

Verilen çizgede bir tepenin komşu tepe dereceleri toplamı, ilgili tepenin var olan tüm komşularına ait derece değerlerinin toplanması ile bulunur. Tüm tepeler için bulunan toplamlar ilgili tepelerin indisleri gözetilerek bir dizi oluşturulur. Bu dizinin elemanları, değerlerinin büyüklüğüne göre küçükten büyüğe sıralanırlar. Çözüm üretilirken bu dizide yer alan ilk tepe sorgusuz çözüme eklenir. Alınan tepe ve o tepeye ait komşu tepeler işaretlenir. Sonraki adımda dizi içinde işaretlenmemiş en küçük değere sahip tepe çözüme eklenir. Çözüme alınan tepe ve o tepeye ait komşu tepeler işaretlenir. Çizgede işaretlenmemiş tepe kalmayınca işlem sonlandırılır. Bulunan çözüm çizgeye ait bir maksimal bağımsız küme olarak verilir. Bu algoritma

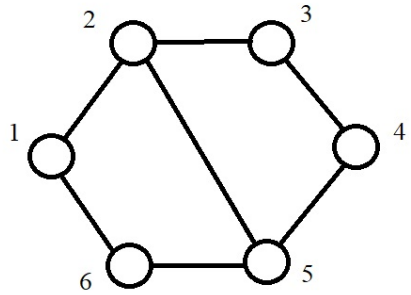


Şekil 2.7 Komşu Tepe Dereceleri Toplamı Metodu için Ters Örnek

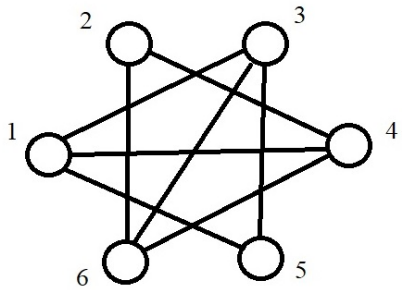
grafiği Şekil 2.7'de verilen, ters örnek olarak karşılaşılan 3 tepeli G çizgesi üzerinde çalıştırılmıştır. Başlangıçta çözüm kümesi (ÇK) ve yasaklı tepeler kümesi (YK) boş küme olarak ayarlanır. G çizgesinin komşu tepe dereceleri sırasıyla [2, 2, 2] olarak hesaplanır ve 1 indisli tepe dizideki en küçük değer olan 2'ye sahip olduğu için ÇK={1} çözüme dahil edilir. Bu aşamada 1 tepesine bitişik olan 2 ve 3 indisli tepeler de yasaklı tepeler olarak YK={2, 3} işaretlenir. Tüm tepeler üzerinde işlem yapıldığı için [2*, 2*, 2*] algoritma sonlanır ve çözüm kümesinin eleman sayısı 1 algoritmanın sonucu olarak raporlanır. Oysa ki bu örnek için en iyi çözüm 2 ve 3 tepelerini içeren kümenin eleman sayısı olan 2 olacaktı.

2.2.4 Problemin Tümleyen Problemi Üzerinden Çözüm Aranması

Maksimum bağımsız küme probleminin karakteristiğinden dolayı ayrıt yoğun çizgelerde karar verme ve yasaklı tepeleri eleme işlemi çok hızlı şekilde yapılabilmektedir. Bu sebeple mümkün olan durumlar daha hızlı denenebilmektedir. Fakat ayrıt yoğunluğu az olan çizgelerde komşuluk ilişkisi zayıf olduğundan bir tepe çözüme alındığı anda yasaklı olarak işaretlenebilen tepe sayısı çok çok az olmaktadır. Bu da deneme miktarını arttırıp problemin makul sürelerde optimum garantili olarak çözülememesine sebep olmaktadır. Yoğun çizgelerde verimli çalışan algoritmaların çokluğu az yoğun çizgelerde bir dönüşüm işlemiyle yoğun hale getirilip çözümün dönüşüm sonrası aranma fikrini düşündürmektedir. Bu işlem verilen çizgenin tümleyen çizgesi üzerinden, MIS probleminin tamamlayıcı problemi olan tepe örtüsü problemine ait çözüm aranarak bulunabilir. Verilen çizgede, var olan ayrıtlar silinip, olmayan ayrıtlar çizilerek çizgenin tümleyeni elde edilir. Üretilen bu çizge üzerinde tepe örtüsü problemine uygun sonuç aramak yerine, maksimum bağımsız küme aranıp, bulunan değer tepe sayısından çıkarılarak sonuca ulaşılır. (Butenko, 2003) Bahsedilen yöntem Şekil 2.8'de verilen çizge, ters örnek olarak karşılaşılan G çizgesi üzerinde çalıştırıldığında elde edilen tümleyen çizge Şekil 2.9 olmaktadır. Şekil 2.8'de verilen G çizgesinde bulunan MIS = {1, 3, 5} olarak bulunabilirken, Şekil 2.9'de verilen G' çizgesinde MIS = {1, 2} olarak bulunmakta eleman sayısı ve



Şekil 2.8 Problemin Tümleyen Problemi için Ters Örnek

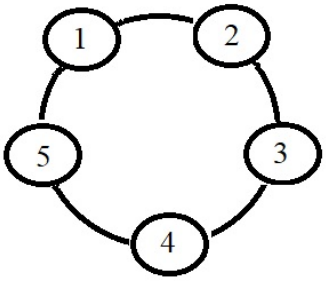


Şekil 2.9 Örnek Problemi Tümleyen Problem

amaç fonksiyonun değeri 2 olarak elde edilmektedir. Bahsedilen yöntem bazı özel problemler için MIS sonucunu vermektedir.

2.3 Kaydırma Yöntemi (Turlama Tekniği)

Anlatılan sezgisel tekniklerde genellikle karar verme anında karşılaşılan sorun (çözümün optimum sonuca ulaşmasını engelleyen faktör) birbirleri ile aynı özellikte olan tepelerden hangisinin ilk olarak çözüme alınmasıdır. Problemin yapısı gereği bir tepe çözüme eklendiğinde diğer tepelerin çözüme alınmasına engel olabilir. (Komşuluk ilişkisi olan tepeler çözümde yer alamazlar.) Eşitlik anında iyi çözüme ulaşmanın engellenmemesi için çözüm oluşturma dizileri üzerinde manipulasyon yapan bir teknik geliştirilmiştir. Bu tekniğe turlama tekniği adı verilmiştir. Turlama tekniği; indis vektörünü bir dairesel kuyruğa dönüştürerek her adımda farklı bir başlangıç noktasından başlayarak çözüm uzayını akıllıca taramaktan ibarettir. Örneğin [1,2,3,4,5] şeklinde elde edilmiş olan indis vektörüne turlama tekniği uygulandığında aşağıdaki sıralamalar elde edilir.



Şekil 2.10 Turlama Tekniği Görselleştirme Örneği

Böylece indisleme için kullanılan sıralama kriterinin hatalı seçiminden veya sıralama esnasında karşılaşılan eşitlik durumlarında öncelik-sonralık tercihinden kaynaklanan olumsuz durumların sayısı azaltılmış olur. Eşitlik durumu ile kast edilen en basit sıralama algoritmasında bile karşılaşılabilen, ele alınan elemanların eşit olduğu durumda hangi indise sahip olanın önde olacağı problemidir.

$$[1, 2, 3, 4, 5]$$
$$[2, 3, 4, 5, 1]$$
$$[3, 4, 5, 1, 2]$$
$$[4, 5, 1, 2, 3]$$
$$[5, 1, 2, 3, 4]$$

MIS problemi alt küme seçim problemi olmasına rağmen bu probleme çözüm getirmeye çalışan bir sezgisel algoritma çözüme alınacak tepeleri bir sıraya göre ele almalıdır eğer çözüme dair çok ince bir görüşe sahip değilse, çünkü algoritmada tercih edilen kritere göre sıralanan tepelerin aynı değere sahip olması durumunda sadece indislere bakılmakta ve bu durum optimumdan uzak çözümlerin bulunmasına neden olmaktadır. Sıra gözetildiği için de n! olası durumun hepsini değerlendirme zorunluluğu olmasa da hatırı sayılır bir bölümünü algoritmanın denemesi gerekir ki izleyen satırlarda bu işin nasıl yapılacağı açıklanacaktır.

Tepe dereceleri kullanılarak manipüle edilmiş olan indis vektörünün [1,2,3,4,5] şeklinde olduğu kabul edilerek turlama tekniği ile yukarıda görülen 5 adet ana dal elde edilmiştir. Çözüm uzayını verimli taramak adına bu 5 ana dalın alt dallarının

oluşturulması için artım miktarının değişken olarak alınması yöntemi kullanılmıştır. Burada artım miktarı 1 olduğunda vektör 1,2,3,4,5 sırasını verirken; artım miktarı 2 olduğunda 1,3,5,2,4 ve artım miktarı 3 olduğunda 1,4,3,5,2 sırasını verecek olup artım miktarları 1..n aralığında seçilmiştir. Dolayısıyla algoritmanın karmaşıklığını belirleyen iki ana bölümde de kullanılan bu yapıda dışarıda vektörün başlangıç noktası döngüsü, içeride de artım miktarı döngüsü bulunmaktadır.

2.4 Meta-Sezgisel Çözüm Yaklaşımları: Genetik Algoritmalar

Genetik algoritmalar en iyi olanın yaşamasını ve doğal seçilim mekanizmasını temel alan algoritmalardır. Bu yöntemde bireylerin popülasyonuyla işe başlanır, her birey çözüm uzayında bir noktayı temsil eder. Her nesilde her bir birey için amaç fonsksiyonunun sonucu değerlendirilir ve daha iyi olan bireyler seçilerek yeni bir popülasyon elde edilir. Özel olarak, yeni bir nesil şu üç aşamayla elde edilir: kalite (sağlamlık) ölçümü, seçilim ve yeni bireylerin oluşturulması (Cura, 2008). Sayımlama yöntemi ve GAMS-CPLEX çözücüsü ile çözdürülmüş olan problemler, yazılan genetik algoritma ile de çözüdürülmüştür. Problemler optimum değerleri ile hesaplanmış olup, çalışma zamanları her bir problem için 0 sn olarak elde edilmiştir.

BÖLÜM ÜÇ
GENETİK ALGORİTMALAR

Genetik algoritmalar, doğal süreçlerin bilgisayarda simule edilmesiyle ortaya çıkan arama ve optimizasyon algoritmalarıdır. Darwin'in çevre şartlarına uyum sağlayan bireylerin hayatta kalması ilkesini benimseyen genetik algoritmalar ilk olarak, Michigan Üniversitesi'nde John Holland tarafından geliştirilmiştir. (Holland, 1975) Holland, doğal sistemlerin süreçlerini açıklamak ve bu sistemlere ait adımları içeren yapay sistem yazılımları tasarlamak amacıyla bu çalışmaları yapmıştır. Bu sayede hem doğal hem de yapay sistemlerde önemli yeniliklere neden olmuştur. Bu yöntemde bireylerin popülasyonları ile işe başlanır, ikili dizinin temsil ettiği birey parametre uzayı R^p'de bir noktayı temsil eder. Her nesilde her bir birey için amaç fonksiyonunun değeri onun uygunluğu olarak değerlendirilir ve daha uygun olan bireyler seçilerek yeni bir popülasyon elde edilir. Böylece bireylerin uygunluğuna dayalı yeni çözümler oluşturulur. Yüksek uygunluk değerine sahip bireyler sıklıkla seçildiği için daha uygun olan bireylerin popülasyona katılması yönünde baskı vardır. Birkaç nesil sonra en iyi bireyin optimal çözümü temsil etmesi veya en azından yaklaşması beklenir. (Goldberg, 1989)

Evrimsel algoritma paradigmalarından biri olan genetik algoritmalar, fonksiyon optimizasyonu problemlerinin ve özellikle bileşi optimizasyon problemlerinin çözümünde, klasik arama yöntemlerine göre önemli üstünlükleri nedeniyle yaygın olarak kullanılmaktadır. Geniş çözüm uzaylarının klasik arama yöntemleriyle taranması hesaplama zamanını arttırmaktadır. Ancak bu tip problemlere, genetik algoritmalar ile kabul edilebilir sürede, optimale yakın çözümler bulunabilmektedir. Genetik algoritmalar özellikle çözüm uzayının geniş, süreksiz ve karmaşık olduğu problem tiplerinde başarılı sonuçlar vermektedir. Genetik algoritmalar rastsal arama yöntemleri olmalarına karşın klasik rastsal arama yöntemlerinden farklıdırlar. Bu farklılıklar değişken ile değil, değişken kümesinin kodlanmış biçimi ile çalışmalarından, tek bir noktadan değil, noktalar kümesinden arama yapmalarından, olasılık içeren kurallar kullanmalarından, türev ve benzeri yardımcı bilgi

kullanmamalarından ve yalnızca amaç fonksiyonu bilgisine gereksinim duymalarından kaynaklanmaktadır. Genetik algoritmalar doğal evrim sürecinden esinlenilerek oluşturulduğu için kullanılan terminoloji de evrimsel kuramın ki ile hemen hemen aynıdır. Doğada bireyler birbirleri ile rekabet halindedir, rakiplerine oranla çevreye daha iyi uyum sağlayan bireylerin diğer bireylere göre hayatta kalma şansı daha fazladır. Bu bireylerin genlerinde kodlanmış olan özellikler bir sonraki nesle geçer ve uyum sağlama yeteneği az olan bireylere göre gelecek nesillerde daha fazla temsil edilirler. Doğada bir gen, kendi başına anlamlı genetik bilgi taşıyan en küçük birimdir. Bir genin alabileceği değerlerin her birine allel denir. Genler kromozomları oluştururlar ve genetik algoritmalarda kromozomlar diziler ile ifade edilmektedir. Dizilerin her biri, bir problemin olası çözümlerine ait kodlanmış bilgiyi içermektedir. Olası çözümlerin tutulduğu diziler topluluğuna da popülasyon adı verilir. Dizilerin çevreye uyum sağlayabilme gücü dizilerde saklanan bilgiye bağlıdır ve uyum değeri olarak adlandırılır. Bu değeri ölçen ilişkiye ise uyum fonksiyonu adı verilir. (Goldberg ve Deb, 1991)

GA'ların çalışma prensibi kısaca şöyle özetlenebilir:

Adım 1. Olası çözümlerin kodlandığı bir çözüm kümesi oluşturulur, bu çözüm kümesi popülasyon olarak, çözümlerin kodları da kromozom olarak adlandırılır.

Adım 2. Popülasyondaki her bir kromozomun uygunluk fonksiyonuna göre ne kadar "iyi" olduğu bulunur. Uygunluk fonksiyonuna göre iyi çözüm sonuçları veren kromozomlar, yeni popülasyona alınmak üzere seçilirler.

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

Adım 4. Tüm kromozomların uygunlukları tekrar hesaplanır.

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

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

3.1 Genetik Algoritmaların Avantajları ve Sınırlılıkları

Genetik algoritlar,

- Paralelleştirilebilir
- Çözüm uzayı geniştir
- Global optimum değerine ulaşmak kolaydır
- Çok değişkenli fonksiyona sahip problemler için de uygulanabilir
- Sadece fonksiyon hesaplamalarını kullanır
- Farklı problemler için kolaylıkla uyarlanabilir
- Gürültülü işlevleri iyi yönetir
- Büyük ve anlaşılması zor problem çeşitlerini kolaylıkla çözüme ulaştırabilir
- Çok modlu problemler için iyi bir çözüm kümesi oluşturur.
- Amaç fonksiyonunun değerlendirilmesinde karşılaşılan zorluklara karşı oldukça dayanıklıdır.
- Çözüm uzayı hakkında özel bir bilgiye ya da türev değerine ihtiyaç duymaz
- Çözüm uzayında oluşan süreksizlikler genel optimizasyon performansı üzerinde çok az etkiye sahiptir
- Local optimum değerlerine takılma konusunda dirençlidir
- Büyük boyutlu optimizasyon problemleri için verimli çalışır
- Optimizasyon problemlerinin birçok çeşidi için çalışabilir

maddeleri ile avantajlı yanları,

- Uygunluk fonksiyonun belirlenmesi kolay değildir

- Problemin genetik gösteriminin tanımlanması kolay değildir
- Zamansız yakınsama gerçekleşir
- Popülasyon büyüklüğü, mutasyon oranı, çapraz geçiş oranı, seçim yöntemi ve gücü gibi çeşitli parametrelerin seçimi kolay değildir
- Türev kullanılamaz
- Probleme özgü bilgiler kolayca dahil edilemez
- Local optimumların tanımlanması kolay değildir
- Etkili bir sonlandırıcıya sahip değildir
- Düzgün tek tip fonksiyonlar için kullanışlı değildir
- Yerel bir arama tekniği ile birleştirilmesi gerekir
- Tam global optimumu bulmada sorun yaşatabilir
- Çok sayıda uygunluk (fitness) fonksiyon değerlendirmelerini gerektirir
- Yapılandırmak basit değildir

maddeleri ile de genetik algoritmaların sınırlılıkları verilebilir. (Sivanandam ve Deepa, 2007)

3.2 Kodlama

Bir problemin çözümü için genetik algoritmanın ilk adımı kodlamanın gerçekleştirilmesidir. Çözümlerin kodlanması, probleme özgü verilerin genetik algoritmanın kullanabileceği yapıya çevrilmesi işlemidir. Genetik algoritma ile çözülen problemler tiplerinden dolayı farklı kodlamalara sahip olabilirler. (Gen ve Cheng, 2000) Henüz tüm problemlere uyan genel bir genetik kodlama mevcut değildir. Hatta birbirine çok benzeyen problemlerin kodlamaları birbirinden çok farklı

olabilmektedir. (İşlier, 2001) Genetik algoritmalarda ikili kodlama (binary) biçimi oldukça yaygın bir yöntemdir. İkili kodlama biçimi genetik algoritmayı, incelenecek problemlerden daha çok bağımsız yapar. Bunun nedeni, ikili kodlama biçiminin kullanıldığı problemlerde, standart mutasyon ve çaprazlama yöntemlerinin sıkça kullanılabilmesidir. Bu yöntemlerin dışında gerçek değer kodlama biçimi, sinir ağları kodlama biçimi ve Lisp ifadeleri kodlama biçimleri mevcuttur. Bu yöntemlerin her biri için özel çaprazlama ve mutasyon operatörleri gerekmektedir. (Nabiyev, 2012)

3.2.1 İkili düzende kodlama

Genetik algoritma için en yaygın kodlama biçimidir. Problemin olası çözümleri 0 ve 1 değerleri olarak kodlanır. Klasik çaprazlama ve mutasyon işlemleri için kullanıma uygundur. Alt küme seçilim problemlerinde var-yok şeklinde kodlanacağı için kullanımı yaygındır. Fonksiyonel arama problemlerinde ise değerin 2 tabanındaki karşılığı olarak kullanılır.

3.2.2 Permütasyon (sıralı) kodlama

Sıralı (permütasyon) kodlama genellikle bileşi fonksiyon problemlerinde kullanılan bir tekniktir. Bu tarz problemlerde kromozomun uygunluk değeri hem genlerin değerlerine hem de sıralarına bağlıdır. Gezgin satıcı, şebeke tasarımı ve araç rotalama problemleri bu kodlamaya uygun problemlerdir.

3.2.3 Değer kodlaması

Karmaşık değerlerin, özellikle gerçek sayıların kullanıldığı problemler için uygun olan kodlama yöntemidir. İkili kodlamanın zorlandığı değerler içeren problemler için tercih edilmektedir. Değer kodlamasında kromozomlar, değerlerden oluşan bir dizi şeklinde oluşturulur. Bazı özel problemler için oldukça kullanışlı olmasına karşın

çaprazlama ve mutasyon işlemleri için özel operatörler tanımlanmasına ihtiyaç duyar. Bu kodlama genellikle sinir ağlarının ağırlıklarının bulunmasında kullanılır.

3.3 Uygunluk Fonksiyonu

Kromozomların ne kadar iyi olduğunu bulan fonksiyona uygunluk fonksiyonu denir. Bu fonksiyon işletilerek kromozomların uygunluklarının bulunmasına ise değerlendirme (evaluation) adı verilir. Uygunluk fonksiyonu kromozomların şifresini çözer (decoding) ve sonra hesaplama yaparak bu kromozomların uygunluğunu bulur. (Şen, 2004) Uygunluk değeri kromozomun hayatta kalma yeteneği ile doğru orantılıdır. Hangi kromozomların bir sonraki nesile taşınacağı ve hangi kromozomların yok olacağı uygunluk değerlerine göre belirlenir. Uygunluk değeri daha yüksek olan kromozomların sonraki nesillere aktarılma olasılığı daha yüksektir. (Herrera ve Verdegay, 1996) Maksimizasyon problemleri için kromozoma ait uygunluk değeri kromozomun amaç fonksiyonu değerine eşit olmaktadır. Fakat minimizasyon problemlerinde amaç en küçük değere ulaşmak olduğundan uygunluk değeri amaç fonksiyonu ile ters orantılı olarak hesaplanır. Ancak bu işlem negatif değerler içeren minimizasyon problemleri için tek başına yeterli olmamaktadır.

3.4 Başlangıç Populasyonunun Oluşturulması

Genetik Algoritmaları diğer sezgisel arama tekniklerinden ayıran özelliği, çözümü noktadan noktaya değil, noktaların oluşturduğu bir çözüm kümesi içinde aramasıdır. Genetik algoritmaların ilk adımı başlangıç popülasyonunun oluşturulmasıdır ve genellikle başlangıç popülasyonu rastgele oluşturulur. Rastgele oluşan çözümler bazı en iyileme problemlerinde algoritma sonlandığında optimumdan uzak çözümlere yönelmeye sebep olabilir. Bu problemlerde başlangıç popülasyonu çeşitli sezgisel yaklaşımlarla üretilebilir. (Taşkın ve Emel, 2009)

3.5 Çoğalma operatörü

Çoğalma işlemi, kromozomların uygunluk fonksiyonu değerlerine göre kopyalandığı bir süreçtir. Bir sonraki nesile aktarma işleminde uygunluk fonksiyonu değerleri yüksek olan kromozomların katkısı daha fazla olacaktır. (Bielli ve diğer., 2002) Çoğalma işlemi; kromozomların seçimi, seçilmiş kromozomların eşleme havuzuna alınması, havuzda çiftler halinde gruplara ayrılması adımlarını içerir. Kromozomlara ait uygunluk değerlerinin hesaplanmasından sonra yeni nesli üretecek olan kromozomların hangileri olacağına bu operatör ile karar verilir. Literatürde çeşitli çoğalma operatörleri mevcuttur. En çok kullanılanları rulet tekerleği, turnuva, sabit durum ve sıralama yöntemleridir.

3.5.1 Rulet tekerleği

Bu yöntemde kromozomlar uygunluk değerleriyle orantılı bir olasılık değerine göre seçilir. Popülasyon büyüklüğü genel olarak sabit tutulduğundan, popülasyonda yer alan tüm kromozomların birikimli bir olasılık değeri olacaktır. Bu değeri bulmak için kromozomların uygunluk değerleri toplanır. Her bir kromozomun uygunluk değeri bu toplama bölünerek [0, 1] arasında kromozomların seçilme olasılıkları bulunur. Bulunan olasılık değerleri 100 ile çarpılarak [0, 100] arasında değerler elde edilir. Bu değerler kullanılarak tablo oluşturulur. Çaprazlama işlemine girecek olan çift belirlenirken rastgele iki sayı üretilerek bu sayıların hangi kromozoma ait olan aralığa düştüğü bulunur. Belirlenen iki kromozom üzerinde çaprazlama işlemi yapılır. (Melanie, 1999)

3.5.2 Turnuva Yöntemi

Popülasyondan rastgele seçilen iki kromozom turnuvaya katılır. Bu iki kromozom uygunluk değerlerine göre kıyaslanır ve daha iyi değere sahip olan kromozom turnuvayı kazanır. Her kromozom iki kez turnuvaya girer. Popülasyondaki en iyi kromozom iki

turnuvayı da kazanır. İki turnuvayı da kaybeden kötü değerli kromozomlar sonraki nesile aktarılma şansını kaybederler. Bu işlem sistematik olarak yapılarak en iyi iki dizi bir sonraki nesile aktarılır. Bu şekilde popülasyonun ortalama uygunluk değeri sonraki nesillere geçildikçe artar. (Chambers, 1998)

3.5.3 Sıralama Yöntemi

Bütün kromozomlar uygunluk değerlerine göre sıralanır. Her kromozoma 1'den başlanarak popülasyon boyutuna kadar bir sıra atanır. Tüm kromozomlar için bir uyum fonksiyonu belirlenir. Sıralama sonucu en iyi kromozomlar sonraki nesile kopyalanırken, en kötü kromozomlar yok olmuş olur.

3.6 Çaprazlama Operatörü

Mevcut gen havuzunun potansiyelini ölçmek üzere, bir önceki nesilden daha kaliteli bir nesil oluşturma amacıyla çaprazlama operatörü kullanılır. Çaprazlama genellikle, belirlenen bir çaprazlama oranına göre seçilen ebeveyn kromozomlara uygulanmaktadır. Çaprazlamanın iki amacı vardır. Birincisi çözüm uzayını aramak, ikincisi ise ebeveyn kromozomlarda yer alan bilgilerin sonraki nesillere aktarılmasını sağlamaktır. Çaprazlama (crossover) genetik algoritmanın en önemli araçlarından birisidir, popülasyonda çeşitliliği arttırarak en iyiye yaklaşmayı sağlar. En çok kullanılan çaprazlama yöntemleri ikilik kodlama düzeninde çaprazlama, permütasyon kodlamasında çaprazlama, sıralı kodlamada çaprazlamadır. (Affenzeller ve diğer., 2009) İkilik kodlama düzeninde çaprazlama türleri şunlardır:

3.6.1 Tek noktalı çaprazlama

Alt sınırı bir ve üst sınırı kromozom uzunluğu olan aralıkta rasgele bir tamsayı seçilir. Seçilen noktada kromozom parçaları yer değiştirirse bu olaya tek noktalı

çaprazlama denir.

3.6.2 İki noktalı çaprazlama

Alt sınırı bir ve üst sınırı kromozom uzunluğu olan aralıkta rasgele iki tamsayı seçilir. Seçilen noktalar arasında kalan kromozom parçaları yer değiştirirse bu olaya iki noktalı çaprazlama denir.

3.6.3 Tekdüze çaprazlama

Bu yöntemde ise rasgele olarak ebeveyn kromozomlarla aynı uzunlukta bir maske kromozom oluşturulur. Maskenin ilgili değeri 0 ise birinci, 1 ise ikinci kromozomun biti seçilir.

3.7 Mutasyon Operatörü

Genetik algoritmalarda sistem belli bir iterasyon sayısına geldikten sonra kromozomlar birbirine git gide benzemektedir. Kromozomlara ne kadar çaprazlama operatörü uygulansa da çözüm çeşitliliği sağlanamamaktadır. Bu durumda dizinin kendi içindeki genler rastgele yer değiştirilir. Bu hamle ile çözüm çeşitliliği korunmuş olur. Ancak mutasyon oranı iyi belirlenmelidir. Değişim fazla olursa çözümden uzaklaşılabilir. Bu sebeple mutasyonun uygulanma olasılığı $\%0,5$ ile $\%15$ arasında belirlenmelidir. Mutasyon kromozomun bir parçasının dışarıdan değiştirilmesi şeklinde tanımlanır. İkili kodlanmış kromozomlarda mutasyon rastgele belirlenen bir genin değiştirilmesiyle sağlanabilir.

BÖLÜM DÖRT
GENETİK ALGORİTMALAR İLE MAKSİMUM BAĞIMSIZ KÜME PROBLEMİNİN ÇÖZÜMÜ

Maksimum bağımsız küme probleminin çözümünde tek bir çözüm üzerinden ilerleyen yöntemler yerine bir grup çözüm (popülasyon) üzerinde arama yapan genetik algoritmalar kullanılmıştır. Bu yöntemin ilk adımı problemin, genetik algoritmada kullanılabilecek şekilde kodlanmasıdır.

4.1 Çözümün Kodlanması

MIS problemi için kullanılan genetik kodlama standart 0-1 ikili kodlama yöntemidir. n boyutlu bir problem için n-bitlik bir kromozom oluşturulur. Kromozom içinde çözümde yer alan tepe 1 ile, alınmayan tepe ise 0 ile temsil edilir. Bu temsillerin yer aldığı bitsel dizi hücreleri de gen olarak adlandırılır. (Beasley ve Chu, 1996)

4.2 Büyük Boyutlu Problemlerin Çözümünde Kullanılan Veri Yapıları

Çizgenin yapısını tutan ve aynı zamanda bağımsızlık kontrolü için kullanılan bitişiklik matrisi Şekil 4.1'deki gibi simetrik bir yapıya sahiptir. Bitişiklik matrisinin sadece üst ya da alt üçgenin kullanılmasının yeterli olması ve dönüşüm formülleri ile bu yapının tek bir dizi içinde tutulması yer karmaşıklığı konusunda daha büyük problemlerin çözülebilmesine imkan sağlamaktadır.

Bu sebeple kullanılan yer karmaşıklığını azaltmak amacıyla bitişiklik matrisi Şekil 4.2' deki gibi tek boyutlu bir dizide tutulmuştur. İlk matristen bu diziye dönüşüm için kullanılması gereken formüller üretilmiştir.

Bitişiklik matrisi olan A_{ij} matrisinden her değerin kendi i ve j değerlerini kullanarak

i\j	1	2	3	4	5	6
1		A1	A2	A4	A7	A11
2	A1		A3	A5	A8	A12
3	A2	A3		A6	A9	A13
4	A4	A5	A6		A10	A14
5	A7	A8	A9	A10		A15
6	A11	A12	A13	A14	A15	

Şekil 4.1 Bitişiklik Matrisi

K														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15

Şekil 4.2 Bitişiklik Dizisi

k dizisinde alması gereken yeri hesaplayan dönüşüm formülü (4.1) verilmiştir.

$$k = \frac{((i-1)*(i-2))}{2} + j \tag{4.1}$$

Dönüşüm formülü olan (5.1) bitişiklik matrisinin alt üçgen elemanlarının indis değerleri bulundukları yerlere yazılarak Şekil 4.3'deki matris elde edilmiştir.

i\j	1	2	3	4	5	6
1						
2	1					
3	2	3				
4	4	5	6			
5	7	8	9	10		
6	11	12	13	14	15	

Şekil 4.3 Bitişiklik Matrisi Alt Üçgen

Sonraki adımda matrisin bütün elemanlarından bulundukları hücrenin j değerleri çıkartılarak Şekil 4.4'deki matris elde edilir.

Bu adımda oluşan değerlere dikkat edildiğinde bu değerlerin bulundukları hücrelerin i değerlerinden 1 eksiğinin 2'li kombinasyonu olduğu görülür.

i\j	1	2	3	4	5	6
1						
2	0					
3	1	1				
4	3	3	3			
5	6	6	6	6		
6	10	10	10	10	10	

Şekil 4.4 Bitişiklik Matrisi Dönüşüm Adımı

0 = ((2-1) * (2-2))/2

1 = ((3-1) * (3-2))/2

3 = ((4-1) * (4-2))/2

6 = ((5-1) * (5-2))/2

10 = ((6-1) * (6-2))/2

Algoritmada matristen diziye dönüşüm kullanıldığı gibi gerektiğinde tutulan bitişiklik dizisinin k indisinden bitişiklik matrisi A_{ij}'ye ulaşabilmek için i ve j değerlerinin de hesaplanmasını sağlayan formüller de kullanılmıştır. Bu formüller (4.2) ve (4.3) ile verilir.

$$i = \lceil \frac{(-1 + \sqrt{(1 + 8k)}}{2} + 1 \rceil \tag{4.2}$$

$$j = k - \frac{((i-1) * (i-2))}{2} \tag{4.3}$$

4.3 Başlangıç Popülasyonunun Oluşturulması

Genetik algoritma kullanılırken genel yaklaşımın aksine öncül çözümler yani anaç bireyler tamamen rasgele oluşturulmamıştır. Çözümün daha iyi sonuçlara ulaşmasını kolaylaştırmak adına öncül çözümler sezgisel yaklaşımlar ile üretilmiştir. Bu yaklaşımlar problemin yapısını inceleyip mantıklı bir karar vererek bağımsız tepelerin çözüme alınma işlemidir. (Hifi, 1997) Sezgisel yaklaşımlardan gelen çözümler,

kaydırma tekniği kullanılarak iyileştirilmiş ve bu teknikten gelen çözümler ile başlangıç popülasyonunda çeşitlilik sağlanmıştır.

4.4 Kullanılan Çoğalma Yöntemleri

Genetik algoritmalar içinde ilk nesil oluşturulup, bu nesil kromozomlarına ait uygunluk fonksiyonu değerlerine göre sıralmaya sokulur. Yüksek değere sahip olan kromozomlar hiç bir değişikliğe uğramadan yeni nesile alınırlar. Bu işleme elitizm ismi verilir. Genellikle değişime uğramadan yeni nesile alınan kromozom sayısı popülasyon büyüklüğünün yarısı kadardır. Elitizm sayesinde iyi değerlere sahip kromozomlar korunmuş ve oluşacak yeni nesilin en az önceki nesil kadar güçlü olması sağlanmış olur. (Ahn ve Ramakrishna, 2003) Yeni popülasyonda boş kalan kromozomlar ise önceki nesile ait tüm kromozomlar üzerinde rulet tekeri tekniğinin kullanılmasıyla belirlenen ikili ebeveyn kromozomdan çaprazlama işlemi ile yeni kromozomlar üretilmesiyle oluşturulur.

4.5 Kullanılan Çaprazlama Yöntemi

Rulet tekeri tekniği ile belirlenen iki ebeveyn kromozomdan yeni bir kromozom üretilirken kullanılan teknik, her bir gen değerinin anneden mi babadan mı geleceğine karar verme işlemidir. Bu karar verme işleminde alışıla geldiği gibi iki değere sahip bir zar atılıp, gelen zara göre ilk kromozoma ait değer gelmesi durumunda ilk kromozomun ilgili geni, diğer durumda ise ikinci kromozomun ilgili değeri alınır. Ancak oluşacak kromozomun uygunluk değerinin yani başarısının daha yüksek olması için bu işlemde hileli bir zar kullanılır. Zardaki olasılık dağılımı anne ve baba kromozomların uygunluk değerleri ile doğru orantılıdır. Yeni oluşturulan kromozomda ilgili genin, uygunluk değeri yüksek kromozomdan alınma şansı daha fazladır.

4.6 Kullanılan Mutasyon Yöntemi

Çözümlerin yerel maksimumlara takılmasını önlemek için mutasyon işlemi uygulanmıştır. Mutasyonun, hangi kromozomun, hangi genine uygulanacağı rastgele belirlenmiştir. Çözümlerde ikili kodlama kullanıldığı için mutasyon işleminde seçilen gen 1 ise 0, 0 ise 1 yapılmıştır. En iyi sonuçlar mutasyon ٪10 oranında kullanılarak elde edilmiştir. (De Falco ve diğer., 2002)

4.7 Kullanılan Parametre Değerleri

Problem üzerinde verimli şekilde çalışması amacıyla oluşturulan genetik algoritma, bir çok farklı parametreye sahiptir. Bu parametreler; popülasyon büyüklüğü, maksimum iterasyon sayısı, çaprazlama olasılığı, mutasyon olasığı ve durdurma kriterleridir. (Grefenstette, 1986) Bu parametreler belirlenirken literatürün yanı sıra farklı parametre değerleri üzerinde denemeler yapılmış farklı boyutta ve yoğunlukta problemler ile test edilmiştir. Popülasyon büyüklüğü, yeni nesil üretimlerinde sabit tutulmuş ve kötü değere sahip çözümlerin yok olması sağlamıştır.

BÖLÜM BEŞ
HESAPLAMA DENEMELERİ

Tepe sayıları 50, 100, 150, 200, 250, 300, 350, 400, 450 ve 500 olan ve her tepe sayısı için ayrıt yoğunlukları ٪5, ٪10, ٪20, ٪30, ٪40, ٪50, ٪60, ٪70, ٪80, ٪90 ve ٪95 olan 110 adet problem rastgele üretilmiştir. Problemleri üreten Pascal kodu EK-1 olarak ekler bölümünde verilmiştir. Üretilen problemler GAMS-CPLEX, yoğun çizgeler için tam çözüm algoritması, başlangıç popülasyonu rastgele oluşturulmuş genetik algoritma ve başlangıç popülasyonu sezgisel olarak oluşturulmuş genetik algoritma ile çözülmüş ve sonuçlar bu bölümde tablolar halinde raporlanmıştır. Ayrıt yoğunlukları ٪5, ٪10, ٪20, ٪30 olan düşük yoğunluklu problemlere ait optimum sonuçlara problemlerin yapısı gereği eldeki yöntemler ile ulaşmak mümkün olmamış, genetik algoritmanın test edilebilmesi için bahsedilen ayrıt yoğunluklarına karşılık gelen yeni bir problem seti üretilmiştir. Yeni problemlerin optimum değerleri problemlerin üretilme işleminde belirlenmiş ve genetik algortimanın bu değerlere ulaşma yetisi test edilmiş ve sonuçlar tablolar halinde raporlanmıştır.

5.1 GAMS-CPLEX Hesaplama denemeleri

Rastgele üretilmiş problemler GAMS-CPLEX çözücüsü ile çözülmüştür. Problemlere ait optimum değerler Tablo 5.3'de, çalışma zamanları ise Tablo 5.1 ve Tablo 5.2 ile verilmiştir. Metin dosyası formatında oluşturulan problemler, GAMS çözücüsüne ait dosya formatına çevrilmiştir. Bu işlem Pascal dilinde yazılan ve EK-3'de verilen kod ile yapılmıştır. Tablolardaki n değeri problemin boyutunu d ise problemin ayrıt yoğunluk değerini göstermektedir. Tablo 5.1 ve Tablo 5.2'de " > " ile gösterilen problemler makul süre içinde sonuca ulaşamayan problemleri göstermektedir. Bu problemler için optimum değer bulunamadığı için Tablo 5.3'de " - " ile işaretlenmişlerdir. Bahsedilen süre 13 saat çalışma zamanı olarak belirlenmiştir.

Tablo 5.1 GAMS Süreler 1 (sn)

n \ d	5	10	20	30	40	50
50	0,01	0,02	0,05	0,17	0,05	0,03
100	0,12	0,69	1,58	2,81	3,05	2,39
150	2,33	46,72	72,52	50,70	52,05	21,17
200	76,59	1125,37	999,97	442,95	253,11	152,41
250	6132,94	>	>	>	3009,44	1246,92
300	>	>	>	>	>	>
350	>	>	>	>	>	>
400	>	>	>	>	>	>
450	>	>	>	>	>	>
500	>	>	>	>	>	>

Tablo 5.2 GAMS Süreler 2 (sn)

n \ d	60	70	80	90	95
50	0,25	0,06	0,03	0,02	0,05
100	2,25	2,42	2,59	1,06	0,39
150	23,22	26,44	26,39	11,47	3,14
200	233,50	318,55	99,530	56,81	18,81
250	773,72	987,39	622,13	150,91	58,44
300	3759,56	1977,97	696,39	183,25	200,28
350	>	3679,44	1771,89	435,72	444,67
400	>	>	4662,31	2177,01	493,23
450	>	>	>	4131,50	942,38
500	>	>	>	9870,61	1974,53

Tablo 5.3 GAMS Optimum Değerler

n \ d	5	10	20	30	40	50	60	70	80	90	95
50	25	20	15	11	10	8	7	5	4	4	3
100	42	29	21	14	11	9	7	6	5	4	3
150	51	35	23	17	12	10	9	7	5	4	4
200	60	41	25	18	14	11	9	7	6	4	4
250	68	-	-	-	14	12	9	7	6	4	4
300	-	-	-	-	-	-	10	8	7	5	4
350	-	-	-	-	-	-	-	8	7	5	4
400	-	-	-	-	-	-	-	-	6	5	4
450	-	-	-	-	-	-	-	-	-	5	4
500	-	-	-	-	-	-	-	-	-	5	4

Tablo 5.4 Optimum Çözüm Üreten Algoritma Sonuçlar

n \ d	5	10	20	30	40	50	60	70	80	90	95
50	25	20	15	11	10	8	7	5	4	4	3
100	-	-	21	14	11	9	7	6	5	4	3
150	-	-	-	17	12	10	9	7	5	4	4
200	-	-	-	-	14	11	9	7	6	4	4
250	-	-	-	-	14	12	9	7	6	4	4
300	-	-	-	-	15	12	10	8	7	5	4
350	-	-	-	-	16	13	10	8	7	5	4
400	-	-	-	-	16	13	10	8	6	5	4
450	-	-	-	-	16	14	11	9	6	5	4
500	-	-	-	-	17	13	10	9	7	5	4

5.2 Yoğun Çizgeler İçin Tam Çözüm Algoritması Hesaplama Denemeleri

Sayımlama yöntemi ve GAMS-CPLEX bazı büyük boyutlu problemlerde süre anlamında verimli olmayabiliyorlar. Genetik algoritmanın test edilebilmesi için problemlere ait optimum değerlerin hesaplanabilmesi ve daha sonra da bu değerlerin genetik algoritmanın buldukları ile karşılaştırılması gerekmektedir. Bu sebeple problemlere ait optimum değerlerin bulunması için tam çözüm yapan bir algoritmaya ihtiyaç vardır. Sayımlama işlemini akıllıca yapan ve yoğun çizgeler için kabul edilebilir süreler içinde problemleri çözüme ulaştıran algoritma C dilinde kodlanmış ve tez için oluşturulmuş olan problemler üzerinde çalıştırılmıştır. Yoğun çizgeler için optimum değer bulan algoritmaya ait C kodu EK-4'de verilmiştir. Programa ait çalışma süreleri Tablo 5.5 ve Tablo 5.6'de, problemlere ait optimum değerler ise Tablo 5.5'de listelenmiştir. Tablo 5.5 ve Tablo 5.6'de " > " işareti bulunan problemlerin çözüm süreleri 10000 sn'den fazla oldukları için sonuca ulaşmayan problemleri göstermektedirler. Yapılan hesaplama denemelerinde arıt yoğunluğu düşük olan bazı problemler için 2 hafta çalışma süresi beklenmesine rağmen algoritma durum denemerini bitirip, sonuca ulaşamamıştır. (Nasibov ve diğer., 2013)

Tablo 5.5 Optimum Çözüm Üreten Algoritma Süreler 1

n \ d	5	10	20	30	40	50
50	151,271	3,797	0,093	0,000	0,000	0,000
100	>	>	87,253	1,062	0,078	0,015
150	>	>	>	45,814	1,344	0,124
200	>	>	>	>	11,719	0,610
250	>	>	>	>	54,237	2,375
300	>	>	>	>	284,230	8,797
350	>	>	>	>	1014,426	27,782
400	>	>	>	>	3476,524	68,315
450	>	>	>	>	7392,594	165,366
500	>	>	>	>	>	267,617

Tablo 5.6 Optimum Çözüm Üreten Algoritma Süreler 2

n \ d	60	70	80	90	95
50	0,000	0,000	0,000	0,000	0,000
100	0,000	0,000	0,000	0,000	0,000
150	0,015	0,000	0,000	0,000	0,000
200	0,109	0,000	0,000	0,000	0,000
250	0,188	0,031	0,015	0,000	0,000
300	0,641	0,078	0,015	0,000	0,000
350	1,640	0,156	0,015	0,000	0,000
400	3,407	0,297	0,047	0,000	0,000
450	7,109	0,594	0,063	0,015	0,000
500	10,657	0,766	0,079	0,016	0,000

Tablo 5.7 Problemlere Ait Bulunan Optimum Sonuçlar

n \ d	5	10	20	30	40	50	60	70	80	90	95
50	25	20	15	11	10	8	7	5	4	4	3
100	42	29	21	14	11	9	7	6	5	4	3
150	51	35	23	17	12	10	9	7	5	4	4
200	60	41	25	18	14	11	9	7	6	4	4
250	68	-	-	-	14	12	9	7	6	4	4
300	-	-	-	-	15	12	10	8	7	5	4
350	-	-	-	-	16	13	10	8	7	5	4
400	-	-	-	-	16	13	10	8	6	5	4
450	-	-	-	-	16	14	11	9	6	5	4
500	-	-	-	-	17	13	10	9	7	5	4

5.3 Başlangıç Popülasyonu Sezgisel Yöntemle Oluşturulan GA için Hesaplama Denemeleri

Başlangıç popülasyonu, litretürdeki genel yaklaşımın dışında sezgisel çözüm yaklaşımları aracılığıyla oluşturulan genetik algoritma tez için rastgele üretilen problemler üzerinde çalıştırılmıştır. Kullandığı opretörler gereği rastgelelik içeren genetik algoritma, her problem için 100 kez çalıştırılmış ve sonuçta ulaşılan minimum, maximum değerler, sırasıyla Tablo 5.8 ve Tablo 5.9'de verilmiştir. 100 çalıştırılmada ulaşılan ortalama değerler Tablo 5.10 ve Tablo 5.11'de, ortalama çalışma zamanları da saniye cisnsinden Tablo 5.12 ve Tablo 5.13'de verilmiştir. Kullanılan genetik algoritmaya ait C kodu EK-6 olarak verilmiştir.

Tablo 5.8 Genetik Algoritmalar Minimum Değerler

n \ d	5	10	20	30	40	50	60	70	80	90	95
50	25	20	15	11	10	8	7	5	4	4	3
100	41	29	20	14	11	9	7	6	5	4	3
150	51	34	22	17	12	10	9	7	5	4	4
200	56	39	24	17	13	11	9	7	6	4	4
250	65	41	25	18	13	11	9	7	6	4	4
300	68	44	26	19	14	12	9	8	7	5	4
350	73	45	26	19	15	12	10	8	7	5	4
400	77	49	28	20	15	12	10	8	6	5	4
450	81	49	28	20	15	12	10	9	6	5	4
500	83	52	29	20	16	13	10	8	7	5	4

Tablo 5.9 Genetik Algoritmalar Maximum Değerler

n \ d	5	10	20	30	40	50	60	70	80	90	95
50	25	20	15	11	10	8	7	5	4	4	3
100	42	29	21	14	11	9	7	6	5	4	3
150	51	35	23	17	12	10	9	7	5	4	4
200	60	41	25	18	14	11	9	7	6	4	4
250	67	44	27	19	14	12	9	7	6	4	4
300	71	46	29	20	15	12	10	8	7	5	4
350	75	48	29	21	16	13	10	8	7	5	4
400	79	53	30	21	16	13	10	8	6	5	4
450	83	53	30	21	16	14	11	9	6	5	4
500	88	54	31	23	17	13	10	9	7	5	4

Tablo 5.10 Genetik Algoritmalar Ortalama Değerler 1

n \ d	5	10	20	30	40	50
50	25,00	20,00	15,00	11,00	10,00	8,00
100	41,64	29,00	20,75	14,00	11,00	9,00
150	51,00	34,04	22,93	17,00	12,00	10,00
200	59,16	40,59	24,23	17,85	13,84	11,00
250	65,24	41,94	25,94	18,93	13,99	11,93
300	68,80	45,00	27,72	19,16	14,71	12,00
350	74,01	46,73	27,72	19,38	15,59	12,92
400	78,06	51,23	29,28	20,16	15,73	12,70
450	82,44	50,45	29,69	20,62	15,98	13,42
500	86,44	52,95	29,91	21,47	16,10	13,00

Tablo 5.11 Genetik Algoritmalar Ortalama Değerler 2

n \ d	60	70	80	90	95
50	7,00	5,00	4,00	4,00	3,00
100	7,00	6,00	5,00	4,00	3,00
150	9,00	7,00	5,00	4,00	4,00
200	9,00	7,00	6,00	4,00	4,00
250	9,00	7,00	6,00	4,00	4,00
300	9,98	8,00	7,00	5,00	4,00
350	10,00	8,00	7,00	5,00	4,00
400	10,00	8,00	6,00	5,00	4,00
450	10,94	9,00	6,00	5,00	4,00
500	10,00	8,98	7,00	5,00	4,00

Sezgisel yöntemle oluşturulan genetik algoritmalar Tablo 5.10 ve Tablo 5.11'de görülen ortalama değerler ve problemlere ait optimum değerleri içeren Tablo 5.4 incelendiğinde problemleri optimum değerlere yakınlığı kabul edilebilir değerlerle çözdüğü görülmektedir. Bahsedilen ortalama değerler programın 100 kez çalıştırılması sonucu bulunan değerlere ait ortalamalardır.

Tablo 5.12 Genetik Algoritmalar Ortalama Süreler 1

n \ d	5	10	20	30	40	50
50	0,006	0,007	0,007	0,007	0,005	0,006
100	0,055	0,075	0,084	0,068	0,061	0,050
150	0,174	0,283	0,292	0,231	0,232	0,185
200	0,713	1,019	0,646	0,681	0,541	0,406
250	1,174	1,783	1,785	1,208	1,164	0,887
300	1,501	1,999	2,653	2,382	2,031	1,604
350	3,078	5,610	5,262	3,946	3,052	2,254
400	4,648	7,713	7,102	5,519	4,888	4,580
450	7,271	14,016	10,100	8,773	7,532	6,335
500	8,849	12,771	14,476	11,777	10,34	8,008

Tablo 5.13 Genetik Algoritmalar Ortalama Süreler 2

n \ d	60	70	80	90	95
50	0,006	0,006	0,006	0,004	0,004
100	0,058	0,056	0,050	0,043	0,047
150	0,152	0,144	0,175	0,161	0,103
200	0,375	0,340	0,371	0,364	0,243
250	0,819	0,781	0,737	0,748	0,513
300	1,340	1,300	0,969	1,170	0,951
350	2,573	2,074	1,554	1,708	1,572
400	3,433	2,959	3,460	2,829	2,620
450	4,853	4,038	5,050	4,035	4,037
500	10,558	5,811	5,099	6,011	5,692

5.4 Klasik Genetik Algoritmalar Hesaplama Denemeleri

Başlangıç popülasyonu rastgele oluşturulan genetik algoritma, tez için rastgele üretilen problemler üzerinde çalıştırılmıştır. Kullandığı operatörler gereği rastgelelik içeren genetik algoritma, her problem için 100 kez çalıştırılmış ve sonucunda ulaşılan minimum, maximum değerler, sırasıyla Tablo 5.14 ve Tablo 5.15'de verilmiştir. 100 çalıştırılmada ulaşılan ortalama değerler Tablo 5.16 ve Tablo 5.17'de, ortalama çalışma zamanları da saniye cisnsinden Tablo 5.18 ve Tablo 5.19'de verilmiştir. Kullanılan genetik algoritmaya ait C kodu EK-5 olarak verilmiştir.

Klasik yöntemle oluşturulan genetik algoritmalar Tablo 5.16 ve Tablo 5.17'de

Tablo 5.14 Klasik Genetik Algoritmalar Minimum Değerler

n \ d	5	10	20	30	40	50	60	70	80	90	95
50	24	19	13	10	8	7	6	4	3	3	2
100	38	25	16	12	9	9	7	5	4	3	3
150	43	28	17	13	11	8	6	5	4	3	3
200	57	35	21	15	12	9	7	7	6	4	3
250	54	34	20	15	11	9	7	6	5	4	3
300	57	35	22	16	12	9	7	7	6	4	3
350	59	37	22	15	12	10	8	6	5	4	3
400	61	39	23	16	13	10	9	7	6	4	3
450	66	40	23	17	13	10	8	7	5	4	3
500	68	41	25	17	14	10	9	7	6	4	3

Tablo 5.15 Klasik Genetik Algoritmalar Maksimum Değerler

n \ d	5	10	20	30	40	50	60	70	80	90	95
50	24	19	14	10	8	7	6	4	4	3	2
100	39	28	19	12	10	9	7	5	4	3	3
150	45	30	19	15	12	10	7	6	5	3	3
200	60	39	25	17	13	10	8	7	6	4	3
250	58	38	23	17	12	10	8	6	5	4	3
300	60	39	23	17	13	10	8	7	6	4	3
350	62	40	24	17	14	11	9	7	6	4	3
400	64	41	24	17	14	10	9	7	6	4	3
450	68	43	25	18	14	10	8	8	5	4	4
500	71	43	27	18	14	11	9	7	6	4	3

görülen ortalama değerler ve problemlere ait optimum değerleri içeren Tablo 5.7 incelendiğinde problemlere optimum değerlere bazı poblemlerde yakın değerler görülürken bazı problemlerde optimumdan uzaklaşmıştır. Bahsedilen ortalama değerler programın 100 kez çalıştırılması sonucu bulunan değerlere ait ortalamalardır. Genetik algoritmaların çalışma süreleri ise Tablo 5.18 ve Tablo 5.19'de görüldüğü şekilde kabul edilebilir büyüklüktedir.

Tablo 5.16 Klasik Genetik Algoritmalar Ortalama Değerler

n \ d	5	10	20	30	40	50
50	24,00	19,00	13,00	10,00	8,00	7,00
100	38,61	26,03	16,67	12,00	9,18	9,00
150	43,96	29,17	17,75	13,74	11,03	9,02
200	57,97	36,33	22,62	16,40	12,14	9,28
250	55,76	35,05	21,64	15,50	11,29	9,36
300	57,97	35,53	22,62	16,40	12,14	9,28
350	60,68	38,53	22,79	16,24	12,92	10,04
400	63,21	40,00	23,43	16,24	13,54	10,00
450	67,25	40,78	23,75	17,25	13,72	10,00
500	69,42	41,95	25,32	17,23	14,00	10,52

Tablo 5.17 Klasik Genetik Algoritmalar Ortalama Değerler

n \ d	60	70	80	90	95
50	6,00	4,00	3,00	3,00	2,00
100	7,00	5,00	4,00	3,00	3,00
150	6,09	5,83	4,40	3,00	3,00
200	7,92	7,00	6,00	4,00	3,00
250	7,19	6,00	5,00	4,00	3,00
300	7,92	7,00	6,00	4,00	3,00
350	8,13	6,98	5,61	4,00	3,00
400	9,00	7,00	6,00	4,00	3,00
450	8,00	7,42	5,00	4,00	3,82
500	9,00	7,00	6,00	4,00	3,00

5.5 Genetik Algoritmaların Bağıl Hata Tablolarına Göre Karşılaştırılması

Klasik genetik algoritmalar ve başlagıç popülasyonu sezgisel yöntemle oluşturulan genetik algoritmalar problemler üzerinde 100'er defa çalıştırılmış ve her problem için ulaşılan değerlerin ortalamaları hesaplanmıştır. Sezgisel yöntemle oluşturulan genetik algoritmaların problemlerde ulaştığı ortalama değerler ve problemlere ait optimum değerler kullanılarak Tablo 5.20 ve Tablo 5.21'deki bağıl hata tabloları oluşturlmuştur.

Klasik genetik algoritmaların problemlerde ulaştığı ortalama değerler ve

Tablo 5.18 Klasik Genetik Algoritmalar Ortalama Süreler 1

n \ d	5	10	20	30	40	50
50	0,003	0,003	0,003	0,002	0,003	0,002
100	0,038	0,035	0,038	0,032	0,055	0,075
150	0,332	0,307	0,297	0,296	0,285	0,279
200	1,461	1,352	1,318	1,259	1,213	1,195
250	2,538	2,334	2,270	2,238	2,160	2,057
300	2,650	2,462	2,292	2,253	2,135	2,047
350	4,231	3,902	3,672	3,588	3,508	3,721
400	6,336	5,778	5,437	5,242	5,177	4,951
450	9,215	8,351	7,783	7,481	8,936	8,329
500	15,922	14,204	13,335	12,782	12,453	11,235

Tablo 5.19 Klasik Genetik Algoritmalar Ortalama Süreler 2

n \ d	60	70	80	90	95
50	0,001	0,001	0,001	0,001	0,000
100	0,069	0,058	0,052	0,053	0,063
150	0,252	0,249	0,227	0,215	0,212
200	1,117	1,028	0,924	0,824	0,867
250	2,027	1,866	1,660	1,541	1,499
300	2,001	1,854	1,672	1,552	1,496
350	2,951	2,712	2,542	2,380	2,317
400	4,412	4,124	4,015	3,951	3,542
450	8,422	7,344	6,250	6,063	5,922
500	10,562	9,375	8,751	7,860	7,719

problemlere ait optimum değerler kullanılarak Tablo 5.22 ve Tablo 5.23'deki bağıl hata tabloları oluşturlmuştur.

Oluşan bağıl hata değerleri incelendiğinde başlangıç popülasyonu sezgisel yöntemle oluşturulan genetik algoritmanın hata değerlerinin çok düşük olduğu, klasik genetik algoritmanın ise optimum değerlerden uzaklaştığı problemlerin çok sayıda olduğu görülmektedir.

Tablo 5.20 Sezgisel Yöntemle Başlatılan GA Bağıl Hata 1

n \ d	5	10	20	30	40	50
50	0,000	0,000	0,000	0,000	0,000	0,000
100	0,009	0,000	0,012	0,000	0,000	0,000
150	0,000	0,027	0,003	0,000	0,000	0,000
200	0,014	0,010	0,031	0,008	0,011	0,000
250	0,041	-	-	-	0,001	0,006
300	-	-	-	-	0,019	0,000
350	-	-	-	-	0,026	0,006
400	-	-	-	-	0,017	0,023
450	-	-	-	-	0,001	0,041
500	-	-	-	-	0,053	0,000

Tablo 5.21 Sezgisel Yöntemle Başlatılan GA Bağıl Hata 2

n \ d	60	70	80	90	95
50	0,000	0,000	0,000	0,000	0,000
100	0,000	0,000	0,000	0,000	0,000
150	0,000	0,000	0,000	0,000	0,000
200	0,000	0,000	0,000	0,000	0,000
250	0,000	0,000	0,000	0,000	0,000
300	0,002	0,000	0,000	0,000	0,000
350	0,000	0,000	0,000	0,000	0,000
400	0,000	0,000	0,000	0,000	0,000
450	0,005	0,000	0,000	0,000	0,000
500	0,000	0,002	0,000	0,000	0,000

5.6 Düşük Yoğunluklu Ek Problemler

Rastgele üretilmiş düşük yoğunluklu problemlerin (%5, %10, %20, %30) etkin sayımlama yapan algoritma ve GAMS ile tam çözümleri bulunamadığı için tezde önerilen genetik algoritmanın karşılaştırmalı başarım oranı hesaplanamamıştır. Bu handikabı ortadan kaldırmak için düşük yoğunluklu problemler rastgele üretilmeyip belli bir sistematiğe göre elde edilmiştir, şöyle ki; öncelikle tepe sayısına bağlı olarak ilgili yoğunluğa ait ayrıt sayısı elde edilmiştir. Ardından toplamda bu ayrıt sayısını elde edebilmek için, maksimum bağımsızlık sayısı bilinen ve kendi içinde ayrıt sayısının en fazla olduğu, birbirinden izole olarak yerleşecek tam çizgelerin tepe

Tablo 5.22 Klasik Genetik Algoritmalar Bağıl Hata 1

n \ d	5	10	20	30	40	50
50	0,040	0,050	0,133	0,091	0,200	0,125
100	0,081	0,102	0,206	0,143	0,165	0,000
150	0,138	0,167	0,228	0,192	0,081	0,098
200	0,034	0,114	0,095	0,089	0,133	0,156
250	0,180	-	-	-	0,194	0,220
300	-	-	-	-	0,191	0,227
350	-	-	-	-	0,193	0,228
400	-	-	-	-	0,154	0,231
450	-	-	-	-	0,143	0,286
500	-	-	-	-	0,176	0,191

Tablo 5.23 Klasik Genetik Algoritmalar Bağıl Hata 2

n \ d	60	70	80	90	95
50	0,143	0,200	0,250	0,250	0,333
100	0,000	0,167	0,200	0,250	0,000
150	0,323	0,167	0,120	0,250	0,250
200	0,120	0,000	0,000	0,000	0,250
250	0,201	0,143	0,167	0,000	0,250
300	0,208	0,125	0,143	0,200	0,250
350	0,187	0,128	0,199	0,200	0,250
400	0,100	0,125	0,000	0,200	0,250
450	0,273	0,176	0,167	0,200	0,045
500	0,100	0,222	0,143	0,200	0,250

sayısı hesaplanır. Probleme ait matris, hesaplanan tepe sayısını baz alan izole tam çizgelerle (örneğin k adet) doldurulur. Bu işlemden sonra eğer boşta tepe kaldıysa (örneğin m adet) kalan tepeleri içeren bir yol çizge ile örnek problem elde edilmiş olur. Oluşturulan örnek problemin tam çözümü ise tam çizgenin ve yol çizgenin maksimum bağımsızlık sayıları sırasıyla 1 ve $\lceil m/2 \rceil$ olduğundan k+$\lceil m/2 \rceil$ kadar olacaktır. Ayrıca ilgili örnek problemin en iyi çözümü probleme ait metin dosyasının son satırına yazdırılmıştır. Problemleri üreten programın Pascal dili ile yazılmış kaynak kodu EK-7 olarak verilmiştir.

Oluşturulan problemler üzerinde genetik algoritma 100 defa çalıştırılmış ve

Tablo 5.24 Düşük Yoğunluklu Ek Problemler GA Ortalama Süreler

n \ d	5	10	20	30
50	0,004	0,003	0,004	0,003
100	0,026	0,023	0,021	0,025
150	0,080	0,081	0,085	0,095
200	0,191	0,183	0,172	0,232
250	0,356	0,442	0,326	0,295
300	0,713	0,619	0,735	0,554
350	1,704	1,680	1,496	1,256
400	2,212	2,329	2,750	1,704
450	3,364	3,156	2,789	2,094
500	4,594	3,625	5,857	2,734

Tablo 5.25 Düşük Yoğunluklu Ek Problemler Optimum Değerler

n \ d	5	10	20	30
50	15	9	7	6
100	18	10	8	11
150	23	12	10	15
200	19	15	13	19
250	21	17	13	5
300	24	20	15	6
350	23	15	18	5
400	20	15	20	6
450	26	15	23	7
500	22	15	25	8

Tablo 5.24'daki ortalama çalışma süreleri elde edilmiştir.

Optimum değerleri Tablo 5.25'de verilen problemler genetik algoritmalar ile 100 defa çözdürülmüş ve genetik algoritmaların bu değerlere ulaşabildikleri görülmüştür. Algoritma içinde yer alan sezgisel yaklaşım, oluşturulan bu problemler üzerinde iyi bir başarı göstermiş ve genetik algoritma süreçleri sonrasında da optimum değerler hesaplanabilmiştir.

BÖLÜM ALTI
SONUÇLAR

Bu çalışmada NP-Zor sınıfı problemlerinden maksimum bağımsız küme probleminin çözümünde farklı yöntemler ve çözücüler test edilmiştir. Test işlemi Pascal dilinde yazılmış bir algoritma ile rastgele oluşturulmuş farklı yoğunluklara sahip problemler üzerinde yapılmıştır. Optimum garantili yaklaşımlardan sayımlama yöntemi kesin sonuç vermesine rağmen n tepeli bir çizge için mümkün olan bütün alt kümelerin çözüm olup olmadığını incelemektedir. Bu sebeple problem boyutu arttıkça incelenmesi gereken durum da programa ait çalışma zamanı da üssel olarak artacaktır. Sonraki adımda daha düşük çalışma zamanına sahip hazır yazılımlardan birisi olan ve problemin matematiksel modelini baz alarak optimum çözümü garanti eden GAMS-CPLEX çözücüsü kullanılmıştır. Bu çözücü problemi dal-sınır yöntemi ile çözdüğü için olası çözümler üzerinde daha hızlı deneme yaparak optimum sonuca ulaştırmaktadır. Problemlerin CPLEX ile çözülebilmesi için GAMS formatına çevrilmesi yine Pascal dilinde yazılan bir program ile yapılmıştır. GAMS-CPLEX çözücüsü çalışmanın ilk kısmında yer alan rastgele oluşturulmuş problemlerde sorun yaşanmadan çok kısa zamanda optimum sonuçlara ulaşmıştır. Ancak tezin asıl problemlerini çözerken farklı sorunlar ile karşılaşmıştır. Bu sorunlardan ilki denenecek ihtimalin fazla olduğu ayrıt yoğunluğu ⁒40 ve daha düsük problemlerde çözüm sürelerinin çok uzamasıdır. İkinci problem ise CPLEX'in yapısı gereği çözüm işlemini bilgisayarın RAM'i üzerinde yapması ve RAM yetmediği durumlara harddisk üzerinde işlem yürütmesiydi. Denenecek durumların sayısı arttıkça programın kullandığı çözüm ağacının hafızada kapladığı alan çok büyük boyutlara ulaşarak bilgisayarın çalışmasını kısıtlamakta ve problem çözüme ulaşmadan GAMS-CPLEX çözücüsünün bellek hatası vermesidir. Bu sebeplerden ötürü optimum çözümlere ulaşmak için C dilinde yazılan yoğun çizgeler için tam çözüm programı kullanılmıştır. Bu program MIS probleminin yapısı gereği, yoğun çizgelere ait problemlerde optimum değerleri kolaylıkla bulurken seyrek çizgelerde en küçük ikinci problem boyutu olan n=100 için bile optimum değerlere ulaşmada sorun yaşamıştır. Sonraki adımda meta-sezgisel yaklaşımla kısa sürelerde optimuma yakın

sonuçlar veren genetik algoritmalar kullanılmıştır. Optimum bulma garantisi olmayan genetik algoritmaların test edilebilmesi için optimum değerleri bilinen problemlere ihtiyaç duyulmaktadır. Oluşturulan 110 adet problemden düşük yoğunlukta olanlar için optimum değer hesaplanamamıştır. Bu sebeple genetik algoritmaların seyrek problemler üzerinde test edilebilmesi için optimum değerleri önceden belirlenen problem grubu oluşturulmuştur. Seyrek çizgeler için sonraki çalışmalarda tam çözüm algritması önerilmesi düşünülmektedir. Çalışmanın asıl odaklandığı kısım ise klasik bir genetik algoritma ile sezgisel yöntemler ile oluşturulan bir genetik algortimanın çözüme ulaşma becerilerinin ve çalışma zamanlarının kıyaslanması olmuştur. Klasik genetik algoritma kontrol grubu olarak tutulmuş ve test grubu olarak belirlenen genetik algoritmaya ait başlangıç popülasyonu, literatürde yer alan sezgisel çözüm yaklaşımları ve bu yaklaşımlardan elde edilen sonuçlar kullanılarak farklı sonuçlar üretebilen kaydırma tekniği ile oluşturulmuştur. Kıyaslamanın net olarak yapılabilmesi için algoritmaların problemlerde aldıkları sonuçların bağıl hataları problemlere ait optimum sonuçlar kullanılarak hesaplanmış ve sonuçları kıyaslanmıştır. 110 ilk grup 40 ek problem seti olmak üzere 150 probleme ait sonuçlar ve algoritmaların çalışma süreleri tablolar ile raporlanmıştır. Her bir problem için genetik algoritmalar 100 defa çalıştırılmış ve buna göre değerlerin, minimum, maksimum ve ortalama tabloları oluşturulmuştur. Sezgisel yaklaşıma sahip genetik algoritmanın, klasik algoritmaya göre optimum sonuç bulma ve optimum değerlere yaklaşmada daha başarılı olduğu görülmüştür.

KAYNAKLAR

Affenzeller, M., Wagner, S., Winkler, S. ve Beham, A. (2009). *Genetic algorithms and genetic programming: modern concepts and practical applications*. Crc Press.

Ahn, C. W. ve Ramakrishna, R. S. (2003). Elitism-based compact genetic algorithms. *IEEE Transactions on Evolutionary Computation*, *7*(4), 367–385.

Alekseev, V. E. ve Lozin, V. V. (2004). Augmenting graphs for independent sets. *Discrete Applied Mathematics*, *145*(1), 3–10.

Arora, S., Lund, C., Motwani, R., Sudan, M. ve Szegedy, M. (1998). Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, *45*(3), 501–555.

Arora, S. ve Safra, S. (1998). Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM (JACM)*, *45*(1), 70–122.

Back, T. ve Khuri, S. (1994). An evolutionary heuristic for the maximum independent set problem. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, 531–535. IEEE.

Beasley, J. E. ve Chu, P. C. (1996). A genetic algorithm for the set covering problem. *European journal of operational research*, *94*(2), 392–404.

Bielli, M., Caramia, M. ve Carotenuto, P. (2002). Genetic algorithms in bus network optimization. *Transportation Research Part C: Emerging Technologies*, *10*(1), 19–34.

Boppana, R. ve Halldórsson, M. M. (1992). Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics*, *32*(2), 180–196.

Butenko, S. (2003). *Maximum independent set and related problems, with applications*. PhD thesis, University of Florida.

Chambers, L. D. (1998). *Practical handbook of genetic algorithms: complex coding systems*, *volume* 3. CRC press.

Çölkesen, R. (2007). *Bilgisayar programlama ve yazılım mühendisliğinde veri yapıları ve algoritmalar*. Papatya Yayıncılık.

Cura, T. (2008). *Modern sezgisel teknikler ve uygulamaları*. Papatya Yayıncılık Eğitim.

Danna, E., Rothberg, E. ve Le Pape, C. (2005). Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, *102*(1), 71–90.

De Falco, I., Della Cioppa, A. ve Tarantino, E. (2002). Mutation-based genetic algorithm: performance evaluation. *Applied Soft Computing*, *1*(4), 285–299.

Elmas, Ç. (2011). *Yapay zeka uygulamaları:(yapay sinir ağı, bulanık mantık, genetik algoritma)*. Seçkin Yayıncılık.

Gen, M. ve Cheng, R. (2000). *Genetic algorithms and engineering optimization*, *volume* 7. John Wiley & Sons.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning, 1989. *Reading: Addison-Wesley*.

Goldberg, D. E. ve Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, *1*, 69–93.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on systems, man, and cybernetics*, *16*(1), 122–128.

Griggs, J. R. (1983). Lower bounds on the independence number in terms of the degrees. *Journal of Combinatorial Theory, Series B*, *34*(1), 22–39.

Herrera, F. ve Verdegay, J. L. (1996). *Genetic algorithms and soft computing*. Physica-Verlag.

Hifi, M. (1997). A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems. *Journal of the Operational Research Society*, *48*(6), 612–622.

Holland, J. H. (1975). Adaptation in natural and artificial systems. *Ann Arbor, MI: The University of Michigan Press*.

İşlier, A. A. (2001). Üretim hücrelerinin bir genetik algoritma kullanılarak oluşturulması. *Anadolu Üniversitesi Bilim ve Teknoloji Dergisi*, *2*(1), 137–157.

Korshunov, A. (1974). Coefficient of internal stability of graphs. *Cybernetics and Systems Analysis*, *10*(1), 19–33.

Lê, N. C., Brause, C. ve Schiermeyer, I. (2017). On sequential heuristic methods for the maximum independent set problem. *Discussiones Mathematicae Graph Theory*, *37*(2), 415–426.

Loukakis, E. (1983). A new backtracking algorithm for generating the family of maximal independent sets of a graph. *Computers & Mathematics with Applications*, *9*(4), 583–589.

Loukakis, E. ve Tsouros, C. (1981). A depth first search algorithm to generate the family of maximal independent sets of a graph lexicographically. *Computing*, *27*(4), 349–366.

Mahadev, N. V. ve Reed, B. A. (1999). A note on vertex orders for stability number. *Journal of Graph Theory*, *30*(2), 113–120.

McConnell, J. J. (2001). Analysis of algorithms: An active learning approach, by jones and bartlett publishers. *Inc., Massachussetts, USA*.

Melanie, M. (1999). An introduction to genetic algorithms. *Cambridge, Massachusetts London, England, Fifth printing*, *3*, 62–75.

Nabiyev, V. V. (2012). *Yapay zeka: insan-bilgisayar etkileşimi*. Seçkin Yayıncılık.

Nasibov, E., Berberler, M. E. ve Atilgan, C. (2013). An efficient algorithm for exact solution of maximum independent set problem in dense graphs. *International Symposium on Computing in Science Engineering. Proceedings*.

Papadimitriou, C. H. ve Steiglitz, K. (1982). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Puchinger, J. ve Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, 41–53. Springer.

Sakamoto, A., Liu, X. ve Shimamoto, T. (1997). A genetic approach for maximum independent set problems. *Ieice Transactıons on Fundamentals of Electronics, Communications and Computer Sciences*, *80*(3), 551–556.

Şen, Z. (2004). Genetik algoritmalar ve eniyilieme yöntemleri (genetic algorithms and optimization methods). *Water Foundation, Istanbul (in Turkish)*.

Sivanandam, S. ve Deepa, S. (2007). *Introduction to genetic algorithms*. Springer Science & Business Media.

Taşkın, Ç. ve Emel, G. (2009). *Sayısal yöntemlerde genetik algoritmalar*. Alfa yayınları. Alfa Aktüel.

EKLER

EK 1: Metin Dosyası Şeklinde Problem Üreten Pascal Kodu

```
uses crt;

const
n=32;
p=50;

var
t       : text;
ns,ps,s : string;
a       : array[0..n,0..n] of byte;
i,j,k,m : longint;

begin
clrscr;
str(n,ns);
str(p,ps);
s:=ns+ps+'.txt';
assign(t,s);
rewrite(t);

m:=round((n*(n-1)*p)/(2*100));
i:=0;
j:=i+1;
while (k<n) do
begin
a[i,j]:=1;
a[j,i]:=1;
k:=k+1;
i:=i+1;
j:=j+1;
end;

randomize;
```

```
while (k<m) do
begin
repeat
i:=random(n)+1;
j:=random(n)+1;
until (i<>j) and (a[i,j]=0);
a[i,j]:=1;
a[j,i]:=1;
k:=k+1;
end;

writeln(t,n);
for i:=1 to n do
begin
for j:=1 to n do
write(t,a[i,j]:2);
writeln(t);
end;
close(t);

writeln('n:',n,'p:%',p,'m:',m);
writeln('Ok..');
repeat
until keypressed;
end.
```

EK 2: Sayımlama C Kodu

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

#define n 20
#define p 50


int main()
{
unsigned int i,j,z,l=0,uss,ii,jj;
unsigned int k[n+1]={0},max=0,sol[n+1]={0};
char a[n+1][n+1]={0};
FILE *f;
char fins[12],ns[8],ps[8];

itoa(n,ns,10);
itoa(p,ps,10);
strcpy(fins,"");
strcat(fins,ns);
strcat(fins,ps);
strcat(fins,".txt");
f=fopen(fins,"r");
fscanf(f,"%d",&i);
if (i!=n)
{
printf("Tepe Sayisi %d dosyadaki ile uyumsuz !",n);
return 1;
}
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
fscanf(f,"%d ",&a[i][j]);
fclose(f);
```

```
clock_t start, end;
start = clock();

uss=pow(2,n);
for(i=1;i<uss;i++)
{
k[0]=0;
z=1;
for(j=1;j<=n;j++)
{
if(k[j]+z==2)
{
k[j]=0;
z=1;
}
else
{
k[j]+=z;
z=0;
}
k[0]+=k[j];
}
z=1;
for(ii=1;ii<n;ii++)
for(jj=ii+1;jj<=n;jj++)
if(a[ii][jj] && k[ii]+k[jj]>1) z=0;
if(z && k[0]>max)
{
max=k[0];
for(j=1;j<=n;j++)
sol[j]=k[j];
}
}

end = clock();
printf("\nTime:%f\n\n",(float)(end - start)/CLK_TCK);
```

```
printf("Opt:%d\n\nVektor:",max);
for(j=1;j<=n;j++)
if (sol[j]) printf("%d ",j);
printf("\n");
}
```

EK 3: Metin Dosyasında Verilen Problemi GAMS Formatına Çeviren Pascal Kodu

```
uses crt;

const
n=5;
p=60;

var
t          : text;
i,j,m      : longint;
a          : array[1..n,1..n] of byte;
ns,ps,ts   : string;

begin
clrscr;
str(n,ns);
str(p,ps);
ts:=ns+ps+'.txt';
assign(t,ts);
reset(t);
readln(t,i);
if i<>n then
begin
writeln('Boyut Hatasi!');
repeat
until keypressed;
halt;
end;
for i:=1 to n do
for j:=1 to n do
begin
read(t,a[i,j]);
if (i<j) and (a[i,j]=1) then m:=m+1;
end;
close(t);
```

```
ts:=ns+ps+'.gms';
assign(t,ts);
rewrite(t);
writeln(t,'set');
writeln(t,'i /1*',n,'/;');
writeln(t);
writeln(t,'variables');
writeln(t,'z');
writeln(t,'x(i)');
writeln(t,'binary variable x(i);');
writeln(t);
writeln(t,'equations');
writeln(t,'obj,');
for i:=1 to m-1 do
writeln(t,'c',i,',');
writeln(t,'c',m,';');
writeln(t);
writeln(t,'obj..z=e=sum(i,x(i));');
m:=0;
for i:=1 to n-1 do
for j:=i+1 to n do
if a[i,j]=1 then
begin
m:=m+1;
writeln(t,'c',m,'..x(''',i,''')+x(''',j,''')=l=1;');
end;
writeln(t);
writeln(t,'model mis /all/;');
writeln(t,'solve mis using mip maximizing z;');
writeln(t,'display z.L');
close(t);
writeln('ok..');
repeat
until keypressed;
end.
```

EK 4: Yoğun Çizgeler İçin Tam Çözüm Algoritması C Kodu

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <bitset>
#include <time.h>

#define n 50

using namespace std;

typedef bitset<n+1> SET;
SET graf[n+1], mis[n+1];

unsigned int solution[n+1], bestSolution[n+1];
unsigned int lastList[n+1];

int main() {

unsigned int i, j, k, last, score = 0;
unsigned int bound, control, itr = 0;
bool flag;

// DOSYA OKUMA
ifstream read("5095.txt");

if (!read) {
cout << "Dosya bulunamadi!";
getchar();
exit (EXIT_FAILURE);
return 0;
}

read >> k;
if (k != n) {
```

```
cout << "Boyut hatasi! n sabitini degistir!";
getchar();
exit (EXIT_FAILURE);
return 0;
}

for (i=1; i<=n; i++) {
graf[i].set(i, true);
for(j=1; j<=n; j++) {
read >> k;
if(k == 1)
graf[i].set(j, true);
}
}
read.close();



clock_t start, end;
start = clock();

// ANA Dongu
score=1;
control = n;
bestSolution[1]=1;
for (i=1; i<=control; i++) {

for (j=1; j<=n; j++) {
solution[j] = 0;
mis[j].reset();
}

solution[i] = 1;

mis[1] |= graf[i];

last = i;
lastList[1] = i;
```

```
k = 2;  lastList[2] = 0;

for (j = n; j >= i; j--)
if(!mis[1].test(j)) {
bound = j;
break;
}

if (bound == i) continue;

flag = true;
do {
if (!flag)
last = lastList[k];
mis[k].reset();
solution[lastList[k]] = 0;
k--;
if (k == 0) break;
flag = false;

for (j = last+1; j<=bound; j++) {

if(!mis[k].test(j)) {
mis[k+1] = mis[k] | graf[j];
k++;
solution[j] = 1;
lastList[k] = last = j;
flag = true;
}

// misin boyu n ise dur
}

if (score < k) {
for (j=1; j<=n; j++)
bestSolution[j] = solution[j];
score = k;
```

```
control = n - score;
}

itr++;


if (bound - lastList[2] + 1 < score)
break;

} while (last-1 != bound || k > 2);
}

end = clock();

printf("Skor = %d\n", score);
for (i=1; i<=n; i++)
if (bestSolution[i] == 1)
printf("%d ", i);
cout << endl;

printf("\n\nZaman: %.3f sn.\n\n", (float)
  (end - start)/CLOCKS_PER_SEC);

printf("iterasyon = %d\n", itr);

getchar();
return 0;
}
```

EK 5: Başlangıç Populasyonu Rastgele Üretilen GA C Kodu

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

#define n  100    //tepe, cozum vektoru eleman sayisi
#define p  50      //problem ayrit yogunlugu
#define m  n      // populasyon (cesitli cozum sayisi)
#define maxit n

unsigned int fit[m+1]={0},fi[m+1]={0};


void qsort(unsigned int ll,unsigned int rr)
{
unsigned int i,j,x,y;
i=ll;
j=rr;
x=fit[(int)((ll+rr)/2)];
do
{
while (fit[i]>x) i++;
while (x>fit[j]) j--;
if (i<=j)
{
y=fit[i]; fit[i]=fit[j]; fit[j]=y;
y=fi[i]; fi[i]=fi[j]; fi[j]=y;
i++; j--;
}
}while (!(i>j));
if (ll<j) qsort(ll,j);
if (i<rr) qsort(i,rr);
}
```

```
int main()
{

int toplam=0,ctop=0,zar=0,zar1=0,zar2=0,zar3=0,e1=0,e2=0,e3=0;
int iter=0,i,j,k,h,q,rus[m+1]={0},cap[3]={0},pk;
char mat[n+1][n+1]={0},pop[m+1][n+1]={0},pop2[m+1][n+1]={0};

//dosyadan problemi alma
FILE *f;
char fins[12],ns[8],ps[8];

// Text dosyadan okunuyor..
itoa(n,ns,10);
itoa(p,ps,10);
strcpy(fins,"");

strcat(fins,ns);
strcat(fins,ps);
strcat(fins,".txt");
f=fopen(fins,"r");

fscanf(f,"%d",&pk);
if(n!=pk)
{
printf("boyut hatasi!");
return 1;
}
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
fscanf(f,"%d",&mat[i][j]);
fclose(f);
// START
clock_t start, end;
start = clock();
//baslangic pop rastgele olusturulmasi (problem kisitlara uygun!)
for(i=1;i<=m;i++)
for(j=1;j<=n;j++)
```

```
{
pop[i][j]=2;
pop2[i][j]=2;
}

srand(time(NULL));

for(i=1;i<=m;i++)
for(j=1;j<=n;j++)
{
if(pop[i][j]==2)
{
h=rand()%4;
if  (h==0)
pop[i][j]=0;
else
pop[i][j]=1;
if(pop[i][j]==1)
{
for(k=1;k<=n;k++)
{
if(mat[j][k]==1)
{
pop[i][k]=0;
}
}
}
}
}

bas:
//fitness hesabi ve fitness degerine (hedefe) gore siralama
for(i=1;i<=m;i++)
{
fit[i]=0;
fi[i]=i;
for(j=1;j<=n;j++)
```

```
fit[i]+=pop[i][j];
if(fit[i]==0)
{
pop[i][1]=1;
fit[i]=1;
}
}
qsort(1,m); //max degeri uste alir

//elitism
for(i=1;i<=m/2;i++)
for(j=1;j<=n;j++)
pop2[i][j]=pop[fi[i]][j];

//caprazlama hazirligi es secimi (rulet)
toplam=0;
for(i=1;i<=m;i++)
toplam+=fit[i];

rus[0]=0;
for(i=1;i<=m;i++)
rus[i]=rus[i-1]+(fit[fi[i]]*100)/(toplam+1);
for(i=m/2+1;i<=m;i++)
{
e1=0;
zar=rand()%100;
do
{
e1++;
}while((rus[e1]<zar) && (e1<m));
do
{
e2=0;
zar2=rand()%100;
do
{
e2++;
```

```
}while((rus[e2]<zar2) && (e2<m));

}while(e1==e2);

//caprazlama (fitness a gore oncelikli)
ctop=fit[e1]+fit[e2];
cap[0]=0;
for(j=1;j<2;j++)
cap[j]=cap[j-1]+(fit[e1]*100)/ctop;
for(j=1;j<=n;j++)
{
if(pop2[i][j]==2)
{
zar3=rand()%100;
e3=0;
do
{
e3++;
}while((cap[e3]<zar3) && (e3<2));
if(e3==1)
pop2[i][j]=pop[fi[e1]][j];
else
pop2[i][j]=pop[fi[e2]][j];

if(pop2[i][j]==1)
{
for(k=1;k<=n;k++)
{
if(mat[j][k]==1)
{
pop2[i][k]=0;
}
}
}
}
}
}
```

```
//bitirme sarti
iter++;
if(iter<maxit)
{
for(i=1;i<=m;i++)
for(j=1;j<=n;j++)
{
pop[i][j]=pop2[i][j];
pop2[i][j]=2;
}

for(j=1;j<=n;j++)
printf("%5d",pop[1][j]);
printf("\n\n");
printf("%5d",fit[1]);
printf("\n\n");

goto bas;
}
end = clock();

fprintf("\ntime:%f mis:%d\n",(float)(end - start)/CLK_TCK,fit[1]);

}
```

EK 6: Sezgisel Yöntemle Oluşturulan GA C Kodu

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

#define n 100
#define p 5

#define m n
#define maxit n
#define dmax (n-1)

unsigned int pop1f[m+1]={0},pop1fi[m+1]={0},pop2f[m+1]={0};
unsigned int av[n+1]={0},avi[n+1]={0},avo[n+1]={0};

void qsortbk(unsigned int l,unsigned int r)
{
unsigned int i,j,x,y;
i=l;
j=r;
x=av[(int)((l+r)/2)];
do
{
while (av[i] > x) i++;
while (x > av[j]) j--;
if (i<=j)
{
y=av[i]; av[i]=av[j]; av[j]=y;
y=avi[i]; avi[i]=avi[j]; avi[j]=y;
i++; j--;
}
}while (!(i>j));
if (l<j) qsortbk(l,j);
if (i<r) qsortbk(i,r);
```

```
}

void qsortbk2(unsigned int l,unsigned int r)
{
unsigned int i,j,x,y;
i=l;
j=r;
x=pop1f[(int)((l+r)/2)];
do
{
while (pop1f[i] > x) i++;
while (x > pop1f[j]) j--;
if (i<=j)
{
y=pop1f[i]; pop1f[i]=pop1f[j]; pop1f[j]=y;
y=pop1fi[i]; pop1fi[i]=pop1fi[j]; pop1fi[j]=y;
i++; j--;
}
}while (!(i>j));
if (l<j) qsortbk2(l,j);
if (i<r) qsortbk2(i,r);
}

int main()
{
char tc=1,dac[n+1]={0},smis[n+1]={0},tmis[n+1]={0};
char chi[n+1]={0},nodes;
char pop1[m+1][n+1]={0},pop2[m+1][n+1]={0},pop[m+1][n+1]={0};
unsigned int da[n+1]={0},mis=0,misn,c,d,e,i,j,k,t,mv,e1,e2,it=1;
unsigned int vd[n+1]={0},eM[n+1][n+1]={0},popi[m+1]={0};
float vdf[n+1]={0},popf[m+1]={0},ed;
FILE *f;
char fins[12],ns[8],ps[8];

// Text dosyadan okunuyor..
itoa(n,ns,10);
itoa(p,ps,10);
```

```
strcpy(fins,"");

strcat(fins,ns);
strcat(fins,ps);
strcat(fins,".txt");
f=fopen(fins,"r");

fscanf(f,"%d",&i);
if (i!=n)
{
printf("Tepe Sayisi %d dosyadaki ile uyumsuz!",n);
return 1;
}
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{
fscanf(f,"%d",&k);
if(k==1)
{
vd[i]++;
eM[i][0]++;
eM[i][eM[i][0]]=j;
}
}
fclose(f);

// START
clock_t start, end;
start = clock();
e=0;
for(i=1;i<=n;i++)
{
if(vd[i]==0)      // İzole Tepe
vd[i]=n;      //  ?=1 V n
if(vd[i]>e)
e=vd[i];
}
```

```
c=0;
for(i=1;i<=n;i++)
{
vd[i]=2*e-2*vd[i]+1;
c+=vd[i];
}

for(i=1;i<=n;i++)
vdf[i]=vdf[i-1]+(vd[i]*100.0/c);

// Baslangic Populasyonu
srand(time(NULL));
t=1;
do
{
misn=0;
for(j=1;j<=n;j++)
tmis[j]=0;
mv=0;
nodes=0;
do
{
j=0;
e=rand()%101;
d=rand()%1000;
ed=e*1.0+(d*1.0/1000);
do
{
j++;
}while(ed>vdf[j] && j<n);
if(tmis[j] == 0)
{
mv++;
misn++;
tmis[j]=1;
if (mv==n) nodes=1;
for(k=1;k<=eM[j][0] && nodes==0;k++)
```

```
if(tmis[eM[j][k]]==0)
{
mv++;
tmis[eM[j][k]]=2;
if (mv==n) nodes=1;
}
}
}while(nodes==0);
for(j=1;j<=n;j++)
if (tmis[j]==1)
pop1[t][j]=1;
else
pop1[t][j]=0;
pop1f[t]=misn;
pop1fi[t]=t;
if(misn >= mis)
{
mis=misn;
//printf("BPR:%d\n",mis);
for(j=1;j<=n;j++)
{
smis[j]=tmis[j];
avo[j]+=2*(2-smis[j]);
}
}
t++;
}while(t<=m); // !!!

for(j=1;j<=n;j++)
{
av[j]=avo[j]*avo[j] + (n-vd[j]);
avi[j]=j;
}
qsortbk(1,n);

e=1;
for(d=1;d<=dmax;d++)
```

```
{
for(j=0;j<=n;j++)
{
dac[j]=0;
da[j]=0;
}
t=1-d;
do
{
if(tc==1)
t+=d;
if(t>n)
t=t%n;
if  (dac[t]==0)
{
dac[t]=1;
da[0]++;
da[da[0]]=t;
tc=1;
}
else
{
do
{
t++;
if(t>n)
t=1;
}while(dac[t]==1);
tc=0;
}
}while(da[0]<n);
for(i=1;i<=n;i++)
{
misn=0;
for(j=1;j<=n;j++)
tmis[j]=0;
mv=0;
```

```
nodes=0;
for(j=1;j<=n;j++)
{
if(tmis[avi[da[j]]]==0)
{
tmis[avi[da[j]]]=1;
misn++;
mv++;
if (mv==n) nodes=1;
for(k=1;k<=eM[avi[da[j]]][0] && nodes==0;k++)
if(tmis[eM[avi[da[j]]][k]]==0)
{
mv++;
tmis[eM[avi[da[j]]][k]]=2;
if (mv==n) nodes=1;
}
}
da[j]++;
if (da[j]>n)
da[j]=1;
}
if(nodes==1)
{
e++;
if(e>m)
e=1;
if(misn > pop1f[e])
{
pop1f[e]=misn;
for(j=1;j<=n;j++)
if (tmis[j]==1)
pop1[e][j]=1;
else
pop1[e][j]=0;
}
if(misn >= mis)
{
```

```
mis=misn;
//printf("dP : %d\n",mis);
for(j=1;j<=n;j++)
{
smis[j]=tmis[j];
avo[j]+=8*(2-smis[j]);
}
}
}
}
}

for(j=1;j<=n;j++)
{
av[j]=avo[j]*avo[j] + (n-vd[j]);
avi[j]=j;
}
qsortbk(1,n);

bas:
// Rulet Tekerlegi
for(i=1;i<=m;i++)
{
for(j=1;j<=n;j++)
pop[i][j]=pop1[i][j];
popi[i]=pop1f[i];
}
c=0;
for(i=1;i<=m;i++)
c+=popi[i];
popf[0]=0.0;
for(i=1;i<=m;i++)
popf[i]=popf[i-1]+(popi[i]*100.0/c);

// Siralama & Elitism
qsortbk2(1,m);
for(i=1;i<=(m/2);i++)
```

```
{
for(j=1;j<=n;j++)
pop2[i][j]=pop1[pop1fi[i]][j];
pop2f[i]=pop1f[i];
}

// Çaprazlama
for(i=(m/2)+1;i<=m;i++)
{
e=rand()%101;
d=rand()%1000;
ed=e*1.0+(d*1.0/1000);
e1=0;
do
{
e1++;
}while(ed>popf[e1] && e1<m);
do
{
e=rand()%101;
d=rand()%1000;
ed=e*1.0+(d*1.0/1000);
e2=0;
do
{
e2++;
}while(ed>popf[e2] && e2<m);
}while(e1==e2);
c=popi[e1]+popi[e2];
c=popi[e1]*100/c;
for(j=1;j<=n;j++)
{
e=rand()%101;
if(e<=c)
chi[j]=pop[e1][j];
else
chi[j]=pop[e2][j];
```

```
}
misn=0;
for(j=1;j<=n;j++)
tmis[j]=0;
mv=0;
nodes=0;
e=0;
j=0;
do
{
j++;
e++;
if(j>n)
j=1;
if((chi[avi[j]]==1&&tmis[avi[j]]==0)||(e>n*n&&tmis[avi[j]]==0))
{
mv++;
misn++;
tmis[avi[j]]=1;
if (mv==n) nodes=1;
for(k=1;k<=eM[avi[j]][0] && nodes==0;k++)
if(tmis[eM[avi[j]][k]]==0)
{
mv++;
tmis[eM[avi[j]][k]]=2;
if (mv==n) nodes=1;
}
}
}while(nodes==0);
for(j=1;j<=n;j++)
if (tmis[j]==1)
pop2[i][j]=1;
else
pop2[i][j]=0;
pop2f[i]=misn;
if(misn > mis)
{
```

```
mis=misn;
printf("CO:%d\n",mis);
for(j=1;j<=n;j++)
smis[j]=tmis[j];
}
}

//Iterasyon
it++;
if(it<=maxit)
{
for(i=1;i<=m;i++)
{
for(j=1;j<=n;j++)
pop1[i][j]=pop2[i][j];
pop1f[i]=pop2f[i];
pop1fi[i]=i;
}
goto bas;
}

end = clock();
printf("\nTime:%f\n\n",(float)(end - start)/CLK_TCK);

printf("MIS:%d\n\n",mis);
for(j=1;j<=n;j++)
if (smis[j]==1) printf("%d",j);
printf("\n");
return 0;

}
```

EK 7: Optimum Değerleri Bilinen Seyrek Problemler Üreten Pascal Kodu

```
uses crt,math;

const
n=500;
p=30;

var
t                       : text;
ns,ps,s                 : string;
a                       : array[0..n,0..n] of byte;
i,j,k,l,ne,m,me,o,c     : longint;

begin
clrscr;
str(n,ns);
str(p,ps);
s:=ns+ps+'.txt';
assign(t,s);
rewrite(t);

ne:=trunc((n*(n-1)*p)/(2*100));

m:=2;
repeat
inc(m);
me:=trunc(m*(m-1)/2);
o:=trunc(ne/me);
c:=o*m;
writeln(c);
until c<n;
writeln('---');
writeln(o);
writeln(m);
writeln(n-c);
```

```
for k:=1 to o do
begin
l:=(k-1)*m+1;
for i:=1 to l+m-1 do
for j:=i+1 to l+m-1 do
begin
a[i,j]:=1;
a[j,i]:=1;
end;
end;

for k:=m*o+1 to n-1 do
begin
a[k,k+1]:=1;
a[k+1,k]:=1;
end;



writeln(t,n);
for i:=1 to n do
begin
for j:=1 to n do
write(t,a[i,j]:2);
writeln(t);
end;
writeln(t,o+ceil((n-c)/2));
close(t);

writeln('n:',n,'p:%',p,'m:',m);
writeln('Ok..');
repeat
until keypressed;
end.
```