DOKUZ EYLÜL ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

KÜME ÖRTÜSÜ PROBLEMİNİN GENETİK ALGORİTMA İLE ÇÖZÜMÜ

Müge OLUÇOĞLU

Aralık, 2023

İZMİR

KÜME ÖRTÜSÜ PROBLEMİNİN GENETİK ALGORİTMA İLE ÇÖZÜMÜ

Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi
Bilgisayar Bilimleri Anabilim Dalı Bilgisayar Bilimleri Programı

Müge OLUÇOĞLU

Aralık, 2023
İZMİR

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

MÜGE OLUÇOĞLU tarafından **PROF. DR. MURAT ERŞEN BERBERLER** yönetiminde hazırlanan **"KÜME ÖRTÜSÜ PROBLEMİNİN GENETİK ALGORİTMA İLE ÇÖZÜMÜ"** başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

Prof. Dr. Murat Erşen BERBERLER

Yönetici

Dr. Öğr. Üyesi Ayşe Övgü KINAY

Jüri Üyesi

Dr. Öğr Üyesi Kazım ERDOĞDU

Jüri Üyesi

Prof. Dr. Okan FISTIKOĞLU
Müdür
Fen Bilimleri Enstitüsü

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, eğitimim süresince bana her zaman destek veren, kıymetli bilgilerini her zaman benimle paylaşan, motivasyonumun düştüğü tüm zamanlarda beni hep toparlayıp çalışmamıza kaldığımız yerden büyük bir şevkle devam etmemi sağlayan, tüm sorularıma, takıldığım her konuya bitmeyen sabrıyla beni hep aydınlatan sevgili ve değerli danışman hocam Prof. Dr. Murat Erşen BERBERLER'e ve hayatımın her anında verdiğim tüm kararlarda beni destekleyen aileme ve bu tez süresince onları aksatsam da her anımda beni alttan alan yanımda olan tüm dostlarıma çok teşekkür ederim.

Müge OLUÇOĞLU

KÜME ÖRTÜSÜ PROBLEMİNİN GENETİK ALGORİTMA İLE ÇÖZÜMÜ

ÖZ

Küme Örtüsü Problemi (KÖP), çeşitli alanlarda uygulamaları olan bir kombinatoryal optimizasyon problemidir. Seçilen kümelerin toplam sayısını en aza indirirken tüm öğeleri kapsayacak şekilde belirli bir uzaydan alt küme seçmeyi amaçlar. Doğal evrim süreçlerinden esinlenen Genetik Algoritmalar (GA), karmaşık optimizasyon problemlerinin çözümünde umut vaat eden sonuçlar vermektedir. Bu çalışmada, KÖP'ün üstesinden gelmek için başlangıç popülasyonunda kümelerin frekanslarına göre sıralama yapan bir sezgisel algoritma kullanılarak genetik algoritma ile çözüm sunulmaktadır. Ayrıca çözüm kümesine iyi katkı yapacak alt kümelerin seçimlerine öncelik verilmesi için bir seçim formülü geliştirilmiştir. Önerilen algoritma, seçim, çaprazlama ve mutasyon gibi genetik operatörleri kullanarak bir çözüm popülasyonunu geliştirmeyi amaçlamaktadır. Bireylerin uygunluğu, minimum sayıda küme ile tüm elemanların örtülme yeteneklerine göre belirlenmektedir. Genetik algoritma, nesiller boyunca çözümleri yinelemeli olarak iyileştirir ve kademeli olarak optimum veya optimuma yakın çözümlere yaklaşmaktadır. KÖP'ün çeşitli örnek problemleri üzerinde deneyler yaparak önerilen algoritmanın sonuçları gösterilmiştir.

Anahtar kelimeler: Genetik algoritma, küme örtüsü problemi, yapay zeka yöntemleri, çizge teorisi

SOLUTION OF THE SET COVERING PROBLEM WITH GENETIC ALGORITHM

ABSTRACT

The Set Covering Problem (SCP) is a combinatorial optimization problem with applications in various fields. It aims to select subsets from a given space to cover all elements while minimizing the total number of selected sets. Genetic Algorithms (GA), inspired by natural evolutionary processes, show promising results in solving complex optimization problems. In this study, a genetic algorithm to overcome SCP using a heuristic that sorts sets by their frequency in the initial population is proposed. Additionally, a formula for selecting subsets that will make a positive contribution to the solution set has been established. The proposed algorithm aims to evolve a population of solutions using genetic operators such as selection, crossover and mutation. The fitness of individuals is determined by their ability to cover all elements with a minimum number of sets. The genetic algorithm iteratively improves solutions over generations and gradually approaches optimum or near-optimal solutions. The results of the proposed algorithm are demonstrated by conducting experiments on various example problems of the SCP.

Keywords: Genetic algorithm, set covering problem, artificial intelligence, graph theory

İÇİNDEKİLER

Sayfa

SINAV SONUÇ FORMU ii

TEŞEKKÜR iii

ÖZ iv

ABSTRACT v

ŞEKİLLER LİSTESİ viii

TABLOLAR LİSTESİ ix

KISALTMALAR. x

SEMBOLLER LİSTESİ xi

BÖLÜM BİR – GİRİŞ 1

BÖLÜM İKİ – KARMAŞIKLIK SINIFLARI 4

2.1 Asimptotik Notasyonlar 5

2.1.1 Büyük O Notasyonu (O) 5

2.1.2 Omega Notasyonu (Ω) …. 5

2.1.3 Teta Notasyonu (Θ) ……. 5

2.2 P ve NP Sınıflar ……..……6

BÖLÜM ÜÇ – PROBLEM TANIMI 7

3.1 Problemin Matematiksel Modeli 9

3.2 Literatür Taraması …… 9

BÖLÜM DÖRT – PROBLEMİN ÇÖZÜM YÖNTEMLERİ 14

4.1 Kesin Çözüm Yöntemleri 14

4.1.1 Enümerasyon (Sayımlama) 14

4.2 Matematiksel Model Yazılımı - GAMS 15

4.3 Sezgisel Yöntemler 20

4.3.1 İteratif Açgözlü Yöntemi İle Çözüm 21

4.3.2 Statik Açgözlü Yöntemi İle Çözüm 23

4.4 Meta-Sezgisel Yöntemler 24

BÖLÜM BEŞ – GENETİK ALGORİTMA 27

5.1 Genotip Kodlama. 30

5.1.1 İkili Kodlama 30

5.1.2 Tam Sayılı Kodlama 30

5.1.3 Gerçel Sayılı Kodlama 30

5.1.4 Permütasyon Kodlama 30

5.2 Popülasyon 32

5.3 Seçim Yöntemleri 35

5.3.1 Turnuva Seçim Yöntemi 36

5.3.2 Rulet Tekerleği Seçim Yöntemi 37

5.4 Elitizm 38

5.5 Çaprazlama 38

5.5.1 Tek Noktalı Çaprazlama Yöntemi 39

5.5.2 Çok Noktalı Çaprazlama Yöntemi 39

5.5.3 Oranlı Tek Düze Çaprazlama 40

5.6 Mutasyon 41

5.6.1 Gen Çıkarma Yöntemi 42

5.6.2 Yer Değiştirme Yöntemi ……... 42

5.6.3 Tersine Çevirme Yöntemi ……42

BÖLÜM ALTI – HESAPLAMA DENEMELERİ 44

BÖLÜM YEDİ – SONUÇLAR VE DEĞERLENDİRME 44

KAYNAKLAR 53

ŞEKİLLER LİSTESİ

Sayfa

Şekil 3.1. Örnek küme örtüsü problemi 8
Şekil 4.1. GAMS 0-1 matris örneği 16
Şekil 4.2. GAMS sonuç matrisi ve çözüm tablosu 17
Şekil 5.1. Genetik algoritma akış şeması 29
Şekil 5.2. İkili kodlama örneği 30
Şekil 5.3. Tam sayılı kodlama örneği 31
Şekil 5.4. Gerçel sayılı kodlama 31
Şekil 5.5. Permütasyon kodlama 31
Şekil 5.6. Turlama yöntemi çalışma prensibi 33
Şekil 5.7. Turlama yöntemi çalışma prensibi örnek problem 33
Şekil 5.8. Turnuva seçim yöntemi 37
Şekil 5.9. Rulet tekerleği seçim yöntemi 38
Şekil 5.10. Tek noktalı çaprazlama yöntemi örneği 39
Şekil 5.11. Çok noktalı çaprazlama yöntemi örneği 40
Şekil 5.12. Ebeveyn çaprazlama örneği 41
Şekil 5.13. Gen çıkarma mutasyon yöntemi örneği 42
Şekil 5.14. Yer değiştirme mutasyon yöntemi örneği 42
Şekil 5.15. Tersine çevirme mutasyon yöntemi örneği 43

TABLOLAR LİSTESİ

Sayfa

Tablo 3.1. Literatür taraması sonuçları 12
Tablo 3.1. devamı..... 13
Tablo 4.1. Birinci grup GAMS hesaplama denemeleri sonuçları 18
Tablo 4.2. İkinci grup GAMS hesaplama denemeleri sonuçları..... 18
Tablo 4.2. devamı..... 19
Tablo 4.3. Üçüncü grup GAMS hesaplama denemeleri sonuçları..... 19
Tablo 4.3. devamı..... 20
Tablo 6.1. Eşitlik 6.1 ile üretilen problemlerin çözümü 45
Tablo 6.1. devamı..... 46
Tablo 6.2. Eşitlik 6.2 ile üretilen problemlerin çözümleri 46
Tablo 6.2. devamı..... 47
Tablo 6.3. Eşitlik 6.3 ile üretilen problemlerin çözümleri 47
Tablo 6.3. devamı..... 48

SEMBOLLER LİSTESİ

m	: Eleman sayısı
n	: Alt küme sayısı
a_{ij}	: i. elemanın j. alt kümede yer alması
x_j	: j. alt kümenin çözüm kümesinde olması durumu
c_j	: Alt kümelerin maliyeti

KISALTMALAR

KÖP	: Küme Örtüsü Problemi
NP	: Deterministik olmayan Polinomiyel Zaman Sınıfı
DTM	: Deterministik Turing Makinesi
NDTM	: Non Deterministik Turing Makinesi
P	: Polinomiyel Zaman Sınıfı
GAMS	: The General Algebraic Modeling System
GA	: Genetik Algoritma

BÖLÜM BİR
GİRİŞ

Çizge teorisi, bilgisayar bilimleri ve özellikle matematikte uzun bir geçmişe sahiptir. İlk kez 18. yüzyılda Euler tarafından geliştirilmiştir. Euler'in ardından daha da gelişen çizge teorisi, nesnelerin (düğümlerin ya da tepelerin) ve aralarındaki ilişkilerin (ayrıtların veya kenarların) grafiksel temsilini sağlar. Çizge teorisi, gerçek dünya problemlerini analiz etmek ve modellemek için kullanılan güçlü bir araç haline gelmiştir. Bu alanda yapılan araştırmalar sayesinde, çizgelerin yapısını ve özelliklerini incelemek için çeşitli yaklaşımlar ve yöntemler geliştirmiştir.

Optimizasyon ise matematiksel modelleme ve analiz yöntemlerinin kullanıldığı bir disiplin olup bir dizi kısıt altında bir (bazen daha çok) amaç fonksiyonunun en iyi değerini bulma problemleriyle ilgilidir. Bu problem genellikle kaynakların sınırlı olduğu durumlarda, performansı artırmak, maliyeti en aza indirmek ve dolayısıyla verimliliği arttırmak gibi hedefleri karşılamayı amaçlar. Optimizasyonun temel amacı, belirli bir hedefe ulaşmak için en uygun kararları ve eylemleri belirlemektir. Optimizasyon problemleri genellikle matematiksel programlama veya yöneylem araştırması yöntemleriyle formüle edilir. Bu yöntemler, matematiksel modelleme ile birlikte çeşitli algoritmalar ve hesaplama yöntemleri kullanır.

Genetik algoritmalar, doğada gözlemlenen evrimsel sürece benzer şekilde çalışan arama ve optimizasyon yöntemleridir. Karmaşık çok boyutlu bir arama uzayında en iyinin hayatta kalması ilkesine göre bütünsel en iyi çözümü arar. Problemleri çözmek için bilgisayar ortamındaki evrim sürecini taklit ederler. Çözüm için tek bir yapı geliştirmek yerine bu tür yapılardan oluşan bir küme oluştururlar. Problemin birçok olası çözümünü temsil eden bu küme, genetik algoritma terminolojisinde popülasyon olarak adlandırılır. Popülasyonlar, vektörler, kromozomlar veya bireyler olarak adlandırılan sayı dizilerinden oluşur. Bir bireydeki her elemente gen denir. Popülasyondaki bireyler, evrim sürecinde genetik algoritma operatörleri tarafından belirlenmektedir.

Çizge teorisindeki temel problemlerden biri küme örtüsü problemidir. Küme örtüsü problemi (KÖP), verilen bir uzayda yer alan tüm düğümleri kapatan minimum sayıda alt kümenin seçilerek örtülmesi problemidir. Bu problem, pratik uygulamalarda yaygın olarak karşılaşılan bir optimizasyon problemini temsil eder. Bilgisayar biliminde işletme araştırmalarında, telekomünikasyon ve ulaşım gibi çeşitli alanlarda kullanımları vardır. Optimizasyon versiyonu NP-Zor karmaşıklık sınıfına ait bir problemdir. Bu yüzden tam bir çözümü hesaplamak için polinom zamanda verimli bir algoritma yoktur. Bu nedenle genellikle yaklaşık çözüm için sezgisel yöntemler ve meta-sezgisel yöntemler geliştirilmektedir. Bu tez çalışmasında meta-sezgisel yöntemlerden genetik algoritma seçilerek çözüm elde edilmiştir. Genetik algoritmanın çeşitli parametreleri kullanılarak problemdeki tüm elemanların en az küme ile örtülmesi amaçlanmıştır.

Bu tez çalışmasının ilerleyen kısımlarında bu bölüm ve en sonda kaynaklar dizini de dahil olmak üzere toplamda 7 bölüm vardır.

İkinci bölümde, asimptotik notasyonlar ve P ve NP sınıf kavramları açıklanmıştır.

Üçüncü bölümde, küme örtüsü probleminin tanımı yapılarak matematiksel modeli verilmiş ve problemin günlük hayattaki kullanım alanları açıklanmıştır.

Dördüncü bölümde, küme örtüsü problemi için var olan çözüm yöntemleri açıklanmıştır. Mevcut çözüm yöntemleri arasındaki farklar ve kıyaslamalar yapılmıştır.

Beşinci bölümde, bu tez çalışmasında kullanılan genetik algoritma yöntemlerinden bahsedilmiştir. Ek olarak, tezde kullanılan genetik algoritma parametreleri açıklanmıştır.

Altıncı bölümde ise küme örtüsü problemine ait örnek problemlerin çözüm sonuçları sunulmuştur. Ayrıca, kullanılan yöntemler kıyaslanmıştır.

Tezin son kısmı ise, kaynaklar dizininden meydana gelmiştir. Bu tez çalışmasında yararlanılan akademik çalışmalar listelenmiştir.

BÖLÜM İKİ
KARMAŞIKLIK SINIFLARI

Problem, algoritmalar ve hesaplama yöntemleri kullanılarak ele alınan, çözüm bulmayı gerektiren bir soru veya görevdir. Problemler karmaşıklık açısından farklılık göstermektedirler. Bu farklılıklar her problem için farklı çözüm türüne sahip olabileceklerini de göstermektedir. Tipik bir problemin 3 bileşeni vardır. Bunlar; problemin girdisi, problemin kısıtları ve problemin çıktısıdır. Problem girdisi; algoritmaya verilecek olarak verileri temsil etmektedir. Problemin kısıtları; kuralları tanımlayarak çözümün uyması gereken sınırlarını belirtir. Problemin çıktısı ise, algoritma sonucunda üretilen sonuç veya çözümdür. Bölüm 3'te küme örtüsü problemine ait girdi, kısıt ve istenilen çıktı şartları daha detaylı olarak ele alınmıştır.

Farklı problemler için farklı algoritmalar kullanılmaktadır. Algoritmalar arasındaki verimliliğin ölçülmesi hesaplama karmaşıklığı veya algoritma karmaşıklığı olarak ifade edilmektedir. Problemin girdi boyutu büyüdükçe genelde algoritmanın zaman ve bellek ihtiyaçları da büyümektedir. Yürütme zamanı, problemin girdi boyutuna bağlı bir fonksiyon olarak algoritmanın yürütülmesi için geçen süreyi göstermektedir. Genellikle $O(f(n))$ olarak büyük O notasyonu ile gösterilmektedir. "n" girdi boyutu ile ilişkilidir. "$f(n)$", algoritmanın büyüme oranını temsil eden bir fonksiyondur. Algoritmanın yürütme süresi için bir üst sınır vermektedir. Alan karmaşıklığı ise, yine girdi boyutuna bağlı bir fonksiyon olarak algoritmanın yürütülmesi için gereken bellek alanı miktarını göstermektedir. Genellikle, zaman karmaşıklığı gibi, $O(g(n))$ olarak büyük O notasyonu ile gösterilmektedir. "$g(n)$","n" girdi boyutu arttıkça algoritmanın ihtiyaç duyacağı bellek alanını göstermektedir. Algoritmanın ihtiyaç duyacağı bellek alanı için bir üst sınır vermektedir (McConnell, 2001).

Algoritmaların analizleri sırasında amaç daha düşük zaman ve daha az bellek karmaşıklığına sahip algoritmalar tasarlamaktır. Daha düşük karmaşıklıklar, algoritmanın daha verimli olduğu ve daha büyük girdileri makul miktarda zaman ve bellekle işleyebileceği anlamına gelmektedir. (Papadimitriou ve Steiglitz, 1998).

2.1 Asimptotik Notasyonlar

Asimptotik notasyolar, algoritmaların analizinde, algoritmaların verimliliğini(karmaşıklığını) karşılaştırmak için kullanılan matematiksel araçlardır. Girdi boyutu arttıkça algoritmaların zaman ve bellek karmaşıklıklarının nasıl davrandığını anlamaya yol açmaktadır. Üç yaygın gösterim biçimi vardır (Cormen vd, 2022).

2.1.1 Büyük O Notasyonu (O)

Algoritmalar bağlamında en kötü durum zaman veya bellek karmaşıklığını ifade etmektedir. Bir fonksiyonun büyüme oranının üst sınırını temsil etmektedir. Bu sınır $n > n_o$ için k sabiti ile yapılmaktadır. Eğer, $\exists\ n_0, k > 0$ varsa ve bağıntı $k * f(n)$'in altında veya eşit olarak kalıyor ise $g(n) = O(f(n))$ biçiminde gösterilmektedir.

2.1.2 Omega Notasyonu (Ω)

Algoritmalar için en iyi durum zaman veya bellek karmaşıklığını ifade etmektedir. Bir fonksiyonun büyüme oranının alt sınırını temsil etmektedir. Bu sınır $n > n_o$ için k sabiti ile yapılmaktadır. Eğer, $\exists\ n_0, k > 0$ varsa ve bağıntı $k * f(n)$'in üstünde veya eşit olarak kalıyor ise $g(n) = \Omega(f(n))$ biçiminde gösterilmektedir.

2.1.3 Teta Notasyonu (Θ)

Algoritmalar için ortalama durum zaman veya bellek karmaşıklığını ifade etmektedir. Bir fonksiyonun büyüme oranının hem alttan hem de üstten sınırını temsil etmektedir. Bu sınır $n > n_o$ için k_1, k_2 sabitleri ile yapılmaktadır. Eğer, $\exists\ n_0, k_1, k_2 > 0$ varsa ve bağıntı $k_1 * f(n)$ ile $k_2 * f(n)$ fonksiyonları arasında kalıyor ise $g(n) = \Theta(f(n))$ biçiminde gösterilmektedir.

2.2 P ve NP Sınıflar

Deterministik Turing Makinesi (*DTM*), 1930'larda Alan Turing tarafından ortaya atılmış teorik bir hesaplama modelidir. Bu model, okuma-yazma başlığından, sonsuz uzun bir banttan ve kontrol biriminden oluşmaktadır. M programı için DTM bant üzerinde yazılmış verileri ele alarak sonlu adım sonra durursa M programı karar problemini çözer denilmektedir. Karar problemi sadece "evet" ya da "hayır" şeklinde yanıt veren problemlerdir. DTM, gerçek bir bilgisayarın basitleştirilmiş bir versiyonudur. Deterministik Olmayan Turing Makinesi (*NDTM*), aynı anda birden fazla olası yolu keşfedebilen teorik bir hesaplama modelini ifade etmektedir (Johnson, 1982).

Bir algoritmanın yürütme zamanı, n girdi değerine bağlı olarak gerekli tüm hesaplama adımlarının sayısını veren bir fonksiyondur. Bir polinom sınırlı algoritma, yürütme süresi herhangi bir üst sınır polinomu $P(n)$ ile sınırlanan algoritmadır. Bu fonksiyondan daha hızlı büyüyemez. Yani, girdi boyutu "n" ise, "k" bir sabit ise, algoritma $O(n^k)$ süresince zaman almaktadır. Bu tür algoritmalar tarafından çözülebilecek herhangi bir problem, polinomiyel sınıfa (P Sınıf) ait olarak sınıflandırılmaktadır. Sıralama, arama ve temel çizge algoritmaları P sınıfının bir parçasıdır.

NP sınıfı, Deterministik Olmayan Turing Makinesi tarafından polinom zamanda çözülebilen karar problemlerini oluşturmaktadır. Bir problem için herhangi bir çözümü verildiğinde problemin "n" girdi boyutuna göre bu çözümün polinomiyel zamanda doğrulanabildiği problem sınıfıdır.

BÖLÜM ÜÇ
PROBLEM TANIMI

Optimizasyon, matematiksel modelleme ve analiz yöntemlerini kullanarak, belirli hedeflerin belirli kısıtlar altında en iyi şekilde karşılanmasını amaçlayan bir çalışma alanıdır. Optimizasyon problemleri, bir amaç fonksiyonunun belirli kısıtlar altında maksimize veya minimize edilmesi ile ilgilenir. Kısıtlar, karar değişkenlerinin üzerine uygulanan koşullardır. Optimizasyon, çeşitli disiplinlerde geniş bir uygulama alanına sahiptir. Bu alanlara, lojistik, üretim planlaması, finans, ulaşım ve enerji yönetimi örnek olarak verilebilir. Bu problemlerin çözüm süreçlerinde matematiksel programlama yöntemleri, doğrusal ve türevsel programlama, sezgisel algoritmalar ve meta-sezgisel yöntemler gibi yöntemler kullanılarak en iyi çözüme ulaşmaya çalışılmaktadır.

0-1 tam sayılı programlama problemleri, bir optimizasyon problemi sınıfıdır. Bu problemler, karar değişkenlerinin sadece 0 ve 1 değerlerini alabileceği ve bazı kısıtlar altında bir amaç fonksiyonunu veya maliyeti minimize etmek ya da faydayı maksimize etmek için optimal bir çözümün bulunmasını hedeflemektedir. 0-1 tam sayılı programlama problemleri, bir dizi karar değişkeni ve bunlara uygulanması gereken kısıtlardan oluşur. Karar değişkenleri, genellikle bir seçim veya atama durumunu temsil etmektedir. Her bir karar değişkeni, sadece 0 veya 1 değerlerini alabilir. Bu durum, bir nesnenin seçilip seçilmediğini veya bir işin yapılıp yapılmadığını temsil etmektedir.

Bu problemler, doğrusal programlama, karar ağaçları, karışık tam sayılı programlama gibi çeşitli matematiksel yöntemlerle çözülebilir. 0-1 tam sayılı programlama problemleri, optimizasyon alanında yaygın olan ve uygulanan problemlerdir. Genellikle karmaşık yapılı ve büyük boyutlar içeren problemlerdir.

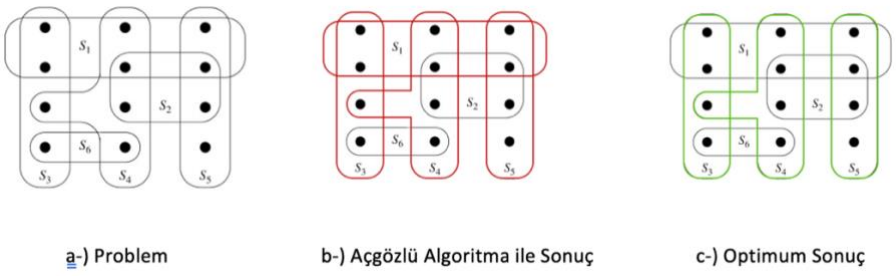
Küme örtüsü problemi, 0-1 tam sayılı programlama modelinin özel bir durumudur. Tüm kısıtlamaların eşitsizliği (>=), tüm sağ taraf değerlerinin 1 olduğu 0-1 tam sayılı bir programlama modelidir. Küme örtüsü probleminin optimizasyon versiyonu NP-Zor

olduğundan, bu problemin polinom zamanda optimum garantili çözümü yoktur. Problem, NP-Zor karmaşıklık sınıfındadır. Küme örtüsü probleminin amacı, en az sayıda alt kümeyi seçerek tüm elemanları örtmektir.

Küme örtüsü probleminde, alt kümeleri seçerken bazı kısıtlamalara uyulması gerekmektedir. Örneğin, her elemanın en az bir alt kümede yer alması kısıtı vardır. Bu tür kısıtlamalar, problemi daha da karmaşık hale getirmekte ve çözümünün bulunmasını zorlaştırmaktadır.

Küme örtüsü probleminin çözümü, özellikle büyük boyutlu problemler için pratikte zor olabilmektedir. Problemin NP-Zor olması, kesin çözümlerin bulunmasını güçleştirir ve genellikle büyük boyutlu problemler için kesin bir çözüm bulmak çok zaman almaktadır. Bu nedenle, küme örtüsü problemi için çoğunlukla sezgisel algoritmalar ve meta-sezgisel algoritmalar kullanılmaktadır. Bu algoritmalar, optimuma yakın bir çözüm üretmeye çalışırken, hesaplama süresini minimize etmek için tasarlanmaktadır.

Küme örtüsü probleminin günlük hayatta birçok alanda pratik uygulaması vardır. Bu kısımda bu pratik alanlarından örnekler verilecektir. Örneğin, mantıksal operatörlerin en küçüklenmesinde, bir şehirdeki güvenlik kameralarının yerleştirilmesinde veya ağ trafiğini en aza indirmek için gerekli yönlendirme tablolarını seçme gibi senaryolarda, tesis yerleşimi planlamasında, bir dizi hizmet noktasının yerleştirilmesi ve hizmet edilecek bölgelerin kapsanmasında benzer şekilde, telekomünikasyon ağları veya veri merkezlerinin yerleşimi gibi alanlarda küme örtüsü problemi çözülmektedir.



Şekil 3.1. Örnek küme örtüsü problemi

Şekil 3.1.a'da örnek bir problem verilmiştir. Bu problemde 12 adet eleman ve 6 adet altküme vardır. Amaç, en az sayıda alt küme seçerek 12 elemanın tamamını kapsamaktır. Şekil 3.1.b'de problem sezgisel yöntemle çözülmektedir. Kırmızı ile işaretli kümeler çözüm kümesine dahil olan kümelerdir. Sezgisel yöntem bir problem çözme tekniğidir. Sezgisel algoritmalar, en iyi çözümü aramaktan ödün vererek zaman maliyetini azaltmakta ve böylece yaklaşık çözümler bulmayı amaçlayan algoritmalardır. Ancak problemin optimum çözümü Şekil 3.1.c'de yeşil işaretli kümeler ile verilmiştir.

3.1 Problemin Matematiksel Modeli

$$Min\ Z = \sum_{j=1}^{n} c_j * x_j \tag{3.1}$$

$$\sum_{j=1}^{n} a_{ij} * x_j \geq 1 \qquad \text{i} = \overline{1,m} \tag{3.2}$$

$$x_j \in \{0,1\} \qquad \text{j} = \overline{1,n} \tag{3.3}$$

Eşitlik (3.1) 'de görüldüğü üzere küme örtüsü problemi bir minimizasyon problemidir. Amaç fonksiyonu z değişkeni ile tanımlanır. Problem veri seti m adet elemandan ve n adet toplam alt kümeden oluşmaktadır. c_j, pozitif değerli $1 * n$ boyutlu bir vektördür. Tüm alt kümelerin maliyetlerini temsil eder. Bu tez çalışmasında, tüm problemler tek maliyetli (unicost) problemler olup alt kümelerin maliyeti 1 olarak alınmıştır. Eşitlik (3.2) 'de verilen a_{ij}, 0 ve 1 değerlerinden oluşan $m * n$ boyutlu bir matristir. Eğer i. eleman, j. alt küme içinde yer alıyorsa, a_{ij} değer 1'dir. Aksi durumda değeri 0 olmaktadır. Eşitlik (3.3) 'de verilen x, $1 * n$ boyutlu bir vektördür. Eğer j. alt küme problem çözümüne dahil edilirse x_j değeri 1'dir. Aksi takdirde değeri 0'dır.

3.2 Literatür Taraması

Bala ve Padberg (1970), küme örtüsü problemine ilk kez değinmişlerdir. Problemin çözümü için matematiksel araştırma yapmışlardır. 1980'lerde, problemin çözümüne

yönelik teoremlerle sezgisel bir algoritma geliştirilmiştir (Wolsey, 1981). Yine aynı yıllarda, Hey (1981) sınırlı sayıda problem kümesinin çözümü için, ağaç arama yöntemi olan durum uzayı gevşetmesi yöntemini kullanarak bazı kısıtları atlayarak, alt sınırları hesaplamaya çalışmıştır. Küme örtüsü probleminin matematiksel modeli ilk olarak 1980'lerin sonlarında tanımlanmıştır (Beasley, 1987). Çalışmada problem boyutunu küçültmeye yönelik alt gradyant hesaplamalarıyla sezgisel algoritmalar geliştirilmesine rağmen istenilen başarı sağlanamamıştır. Yine bir sonraki çalışmasında Beasley, Lagrange Gevşetmesi ile sezgisel bir algoritma geliştirmiş ve diğer sezgisel algoritmalarla karşılaştırmıştır. Önceki çalışmasına göre süreçte daha iyi sonuçlar elde etmiştir (Beasley, 1990b).

Huang vd. (1994)'te ilk kez KÖP'ü genetik algoritma ile çözmüşlerdir. Çözüm yöntemlerinde genetik algoritmaya ilaveten Hamming uzaklıktaki bitişik komşu bireylere uygun çözüm sağlamadıklarında yeni bir ceza yöntemi uygulamışlardır. O zaman için bilinen en iyi sonuçlardan daha iyi sonuçlar bulmuşlardır. Günümüzde hâlâ optimal olarak kabul edilen değerlere, 1990'ların ortasında ulaşılmıştır (Beasley ve Chu, 1996). Çözüm yöntemi olarak genetik algoritma kullanılmıştır. Genetik algoritmada kullanılan yöntemde eşit maliyetli kümeler kapsadıkları eleman sayısına göre sıralanmaktadır. Uygunluk oranları için ceza yöntemi uygulanmıştır.

Al-Sultan vd. (1996)'da KÖP'ü genetik algoritma ile çözmüşlerdir. Sonuçlarını (Beasley, 1990)'daki Lagrange yöntemi ile kıyaslamışlardır. 200 problem üzerinde yaptıkları çalışmalarından %26 oranında daha iyi sonuçlar elde etmişlerdir. (Grossman ve Wool, 1997) 9 farklı sezgisel algoritma denemiştir. En iyi sonuçları yalnızca rastgele açgözlü algoritmada elde etmişlerdir. Bu algoritma sözcüksel sıralı indeksler yerine rastgele sıralı indeksleme ile seçilmiştir. Marchiori ve Steenbeek (1998), iteratif bir sezgisel algoritma geliştirerek en iyi çözümün alt kümesini seçmek için belirli bir kural tasarlamışlardır. Caprara vd (1999), standart alt gradyan yöntemi ile birlikte Lagrange Gevşetmesini kullanarak gerçek hayat verilerinden demiryolu mürettebat problemleri üzerinde çalışmışlardır.

Küme Örtüsü problemini çözmek için sezgisel algoritmalar yetersiz kaldığından meta-sezgisel yöntemlerle çözme yöntemleri başlamıştır. Hadji vd. (2000) karınca kolonisi algoritması ile çözüm geliştirmişlerdir. Algoritmayı çeşitlendirmek için sezgisel bir yerel arama algoritması ile birleştirmişlerdir. Aickelin (2002), üç farklı aşama ile küme örtüsü problemini çözmüştür. Yönteminin ilk aşaması genetik algoritmadır. İkinci aşamada ise uygun değerli bireylerin iyi permütasyonları bulunmaktadır. Son aşamada ise tepe tırmanma algoritması kullanılarak optimize etmiştir. Duan (2007), bir çözümden bazı sütunları rastgele çıkartmış ve açgözlü strateji ile yerine yeni sütunlar eklemiştir. Ren vd. (2008), KÖP'ü çözmek için karınca kolonisi optimizasyon algoritmasını kullanmışlardır. Uygulamayı iki aşamaya bölmüşlerdir. İkinci aşamada dinamik bir sezgisel yöntem kullanmışlardır. Bu aşamada örtülmeyen kümeler için Lagrange dualite kullanılmıştır. İki aşamalı bu yöntem sonucunda problemlerin çözüm sonuçları optimum değerleri görmüştür.

Crawford vd. (2014a) küme örtüsü problemini çözmek için meta-sezgisel bir yaklaşım ele almışlardır. Ele alınan bu yöntem ikili ateş böceği algoritmasıdır. Ateş böceği algoritması, popülasyon tabanlı bir algoritmadır. Bu çalışmada algoritma 30 tekrar ile gerçekleştirilmiştir. Buna rağmen çözdürülen 65 problemin hepsinde optimumdan sapmalar görülmüştür. Aynı yıl aynı çalışma ekibi problemin çözümü için bu sefer de yapay arı kolonisi optimizasyon algoritmasını denemişlerdir. Yapay arı kolonisi optimizasyon algoritmasında, ateş böceği algoritmasından daha iyi sonuçlar elde etmelerini sağlamıştır. Yine 65 problem üzerinde çalıştırılan yapay arı kolonisi optimizasyon algoritmasında sapma miktarları daha düşük ve bazı problemler içinde bilinen optimum sonuçlar elde edilmiştir (Crawford vd, 2014b). Gao vd. (2015) yalnızca tek maliyetli(unicost) problemlerle çalışmışlardır. Bu problemler, problem setindeki her bir alt kümenin maliyetinin 1 olduğu problemlerdir. Çözüm setinde belirli bir süredir bulunmayan alt kümelerin seçilme olasılıklarını yükseltmeyi amaçlayan bir yöntem kullanmışlardır. Bu yöntem ile yerel optimumda sıkışıp kalmanın önüne geçilmiştir. Üç aşamalı bir yöntemdir. Test ettikleri problemlerde %70 başarı elde etmişlerdir.

Soto vd. (2017) KÖP çözümü için guguk kuşu algoritmasını kullanmıştır. Ama çözdürdükleri 65 tane problem içinden sadece birkaç tanesinde optimum sonuçları yakalamışlardır. Crowfard vd. (2018) kedi sürüsü optimizasyonu, ateş böceği, yapay arı kolonisi ve sıçrayan kurbağa algoritmaları gibi farklı meta-sezgisel algoritmalarla sorunu çözerek karşılaştırmışlardır. En iyi çözümü yapay arı algoritmasında elde etmişlerdir. 2020'de aynı araştırmacılar, öğrenmeye dayalı bir optimizasyon algoritması geliştirmişlerdir. Crowfard vd. (2020) bu kez bulgularını daha önce araştırmalarında uyguladıkları yöntemlerle karşılaştırmışlardır. Sonuç olarak yapay arı kolonisi algoritması en iyi sonuçları vermektedir ancak araştırmacılar oluşturdukları tekniğe güvenlerinin tam olduğunu ve basitçe yanlış parametreyi seçtiklerini belirtmişlerdir.

Razip ve Zakaria (2021) küme örtüsü problemini başlangıç popülasyonu bir sezgisel algoritmadan oluşacak şekilde genetik algoritma ile çözmüşlerdir. Elde ettikleri sonuçlar optimuma yaklaşmalarını sağlamış olsa da çeşitlilik açısından iyi sonuçlar elde edememişlerdir.

Tablo 3.1'de küme örtüsü problemi için yapılmış literatür taraması sonuçlarının özetleri görülmektedir. Tabloda son sütunda bulunun sonuçlar, kullanılan çalışma yöntem ve algoritma sonuçlarından elde edilen başarı yüzdesini göstermektedir. Derecelendirme 1 ile 4 arasında yapılmıştır. 1 kötü başarı yüzdesi olarak %0 - %25 arasında, 2 kötü-ortalama başarı yüzdesi olarak %26-%50 arasında, 3 ortalama-iyi başarı yüzdesi olarak %51-%75 arasında ve 4 ise iyi başarı yüzdesi olarak %76-%100 arasındaki sonuçları göstermektedir.

Tablo 3.1. Literatür taraması sonuçları

Yıl	**Yazar**	**Yöntem**	**Sonuçlar***
1990b	Beasley	Lagrange Gevşemeli Sezgisel Algoritma	1
1994	Huang vd	Genetik Algoritma	3
1996	Beasley ve Chu	Genetik Algoritma	4
1996	Al-Sultan vd	Genetik Algoritma	3

Tablo 3.1. devamı

Yıl	Yazar	Yöntem	Sonuçlar*
1997	Grossman ve Yün	9 Sezgisel Algoritma	2
1998	Marchiori ve Steenbeek	İteratif Sezgisel	2
1999	Caprara vd	Lagrange Gevşetmesi	1
2000	Hodji vd	Karınca Kolonisi Optimizasyonu	3
2002	Aickelin	3 aşamalı Meta-Sezgisel Algoritma	3
2007	Duan	Açgözlü Sezgisel Algoritma	2
2008	Ren vd	Karınca Kolonisi Optimizasyonu	3
2014a	Crawford vd	Ateşböceği Optimizasyonu Algoritması	2
2014b	Crawford vd	Yapay Arı Kolonisi	3
2015	Gao vd	Satır Ağırlıklı Yerel Arama Algoritması	3
2017	Soto vd	Guguk Kuşu Optimizasyon Algoritması	2
2018	Crawford, vd	Meta-Sezgisel Algoritmalar	2
2020	Crawford, vd	Öğrenme Tabanlı Optimizasyon Algoritması	2
2021	Razip ve Zakaria	Sezgisel Algoritma Tabanlı Genetik Algoritma	3

*1 kötü sonuçları, 2 kötü-ortalama sonuçları, 3 ortalama-iyi sonuçları ve 4 iyi sonuçları temsil etmektedir.

Bu tez çalışmasında, literatüre geçmiş problemlerle ve farklı kriterlerde oluşturulmuş problem setleri genetik algoritma ile çözülecektir. Genetik algoritmanın başlangıç popülasyonu oluşturulurken sezgisel yöntemler kullanılmaktadır. Çözüm vektörüne iyi katkı sağlayan alt kümelerin ayrıştırılması için bir seçim formülü kullanılmıştır. Sonrasında turlama yöntemi kullanılarak alt kümeler frekanslarına göre çözüm uzayına eklenecektir. Seçim formülü ve turlama tekniğine ait anlatım Bölüm 5'te detaylı olarak ele alınacaktır.

BÖLÜM DÖRT
PROBLEMİN ÇÖZÜM YÖNTEMLERİ

Küme örtüsü problemine birçok farklı yöntem ile çözüm yolları aranmıştır. Tezin bu bölümünde KÖP'e ait çözüm yöntemlerinden kesin çözüm yöntemleri, sezgisel yöntemler ve meta-sezgisel yöntemlerle çözüm örneklerine yer verilecektir.

4.1 Kesin Çözüm Yöntemleri

Problemlerin en iyi (optimum) çözüm değerlerini kesin olarak bulmaya yarayan yöntemlerdir. Kesin çözüm yöntemleri optimum çözümü bulmayı garanti ederler. Ancak kesin çözüm yöntemlerinde, problem boyutu ve karmaşıklığı arttıkça çözüm süreleri daha uzun sürebilmektedir. Bu süreler bazen pratikte kullanılamayacak kadar uzun olabilmektedir.

4.1.1 Enümerasyon (Sayımlama)

Enümerasyon yöntemi, bir problemin tüm olası çözümlerini sistemli bir şekilde listeleyen ve değerlendiren bir yöntemdir. Bu yöntem, küçük ve sınırlı çözüm uzayı için kullanılabilir, ancak genellikle büyük çözüm uzayı için pratikte uygulanması pek mümkün olmamaktadır.

Enümerasyon yöntemi, bir problemin kesin bulmak için olası çözümleri dikkate almak yani tek tek değerlendirmektedir. Bunun için, problemin kısıtları ve amaç fonksiyonu göz önünde bulundurmaktadır. Tüm olası çözümler listelenir. Ardından her bir çözüm, problemin kısıtları ve amaç fonksiyonu ile değerlendirilir. Her bir aday çözüm, problemin kısıtlarını tam olarak karşılayan bir çözümü temsil etmektedir. Enümerasyon yöntemi, tam çözüm sağlamayı garanti etmektedir (Zaozerskaya, 1998). Ancak, çözüm uzayının boyutu arttıkça, enümerasyon yöntemi kullanılarak tüm olası çözümlerin hesaplanması zaman açısından maliyetli hale gelmektedir. Büyük çözüm uzayı için hesaplama süresi ve bellek gereksinimleri nedeniyle pratikte uygulanması zor olan bir yöntemdir**.**

Küme örtüsü probleminin enümerasyon (*sayımla*) yöntemi ile çözümünü 9 eleman ve 9 alt kümeden oluşan bir problem ile açıklayalım. Ve alt kümelerin örttüğü elemanlar şu şekilde verilmiştir. Alt kümelerin isimlendirilmesi, *i* her bir indis değeri olacak olup, S_i şeklinde gösterilmiştir.

Elemanlar: {1,2,3,4,5,6,7,8,9} Alt kümeler: $\{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9\}$

S_1 : {1,4,6,8,9}

S_2 : {2}

S_3 : {3}

S_4 : {4}

S_5 : {5}

S_6 : {6}

S_7 : {7}

S_8 : {2,3}

S_9 : {5,7}

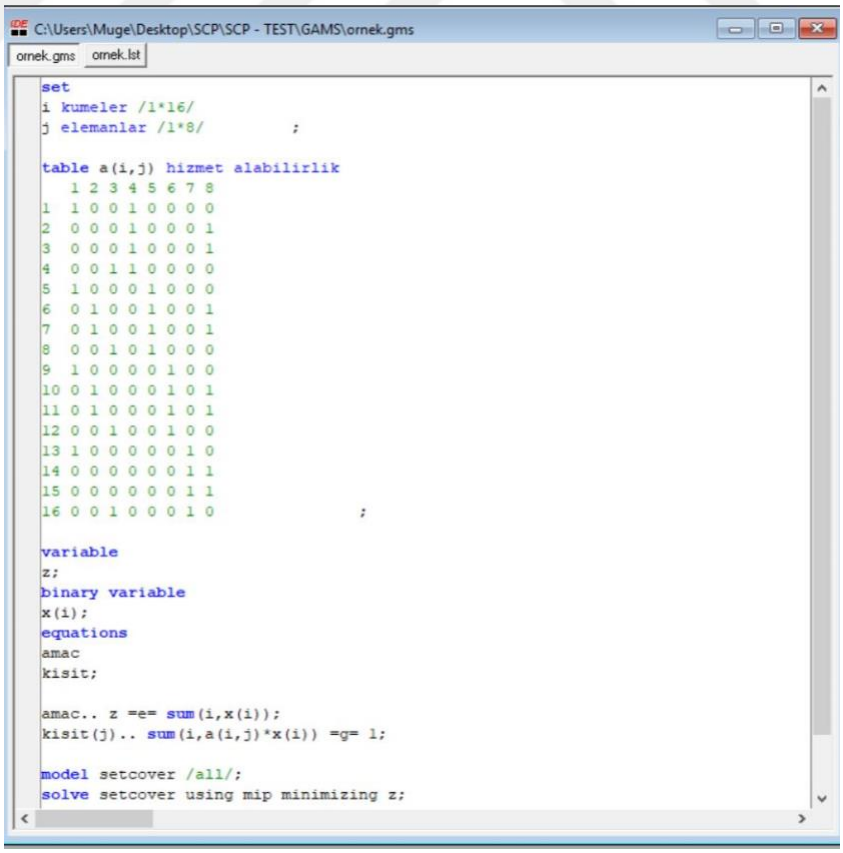
şeklinde verilmiştir. Enümerasyon yönteminde algoritma indis sırası düzeninde 1 numaralı alt kümeden başlayarak tüm alt kümeleri dolaşacaktır. Her yeni bir elemanı örttüğünde çözüm kümesine ilgili alt kümeyi ekleyecektir. 1,4,6,8,9 elemanlarını örttüğü için S_1, 2 elemanını örttüğü için S_2'yi, 3 elemanını örttüğü için S_3'ü, 5 elemanını örttüğü için S_5'i ve 7 numaralı elemanı örttüğü için S_7'yi çözüm kümesine dahil ederek problemi 5 alt küme ile "S_1, S_2, S_3, S_5, S_7" alt kümeleri ile çözmüş olacaktır.

4.2 Matematiksel Model Yazılımı- GAMS

Küme örtüsü probleminin matematiksel modeli kullanılarak optimum garantili çözümler elde etmek için GAMS yazılımı kullanılmıştır. GAMS (General Algebraic Modeling System), matematiksel programlama ve optimizasyon için kullanılan üst düzey bir modelleme sistemidir. GAMS, bir derleyici ve entegre yüksek performanslı çözücülerden oluşmaktadır. GAMS çeşitli çözücü algoritmaları içermektedir. Bu çözücüler sayesinde doğrusal, doğrusal olmayan, karışık tamsayılı problemler

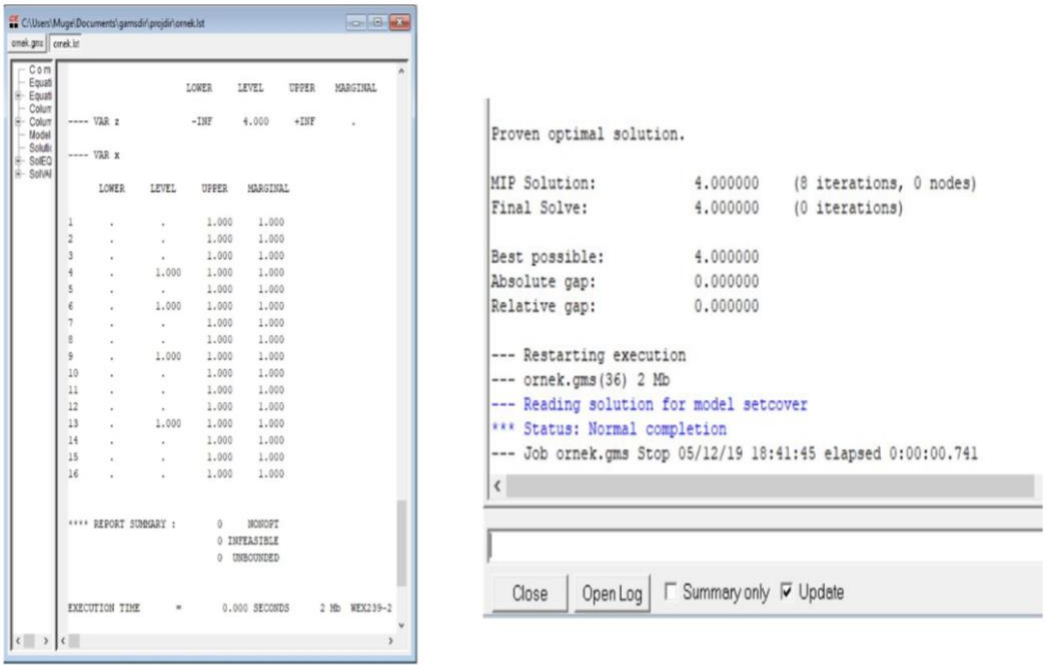
çözülebilmektedir. Çözüm yöntemi kolaylıkla değiştirilebilmektedir. Bu tezde geliştirilen ve kullanılan genetik algoritma ile kıyaslanması açısından GAMS'teki Cplex çözücüsü kullanılmıştır. Cplex çözücü, büyük, zor problemleri hızlı bir şekilde ve minimum kullanıcı müdahalesi ile çözmek için tasarlanmıştır. Cplex, yöntem olarak dal ve sınır algoritmasını kullanmaktadır. Küme örtüsü problemine ait örnek bir GAMS modeli Şekil 4.1'de verilmiştir.

Dal ve sınır algoritması, kombinatoryal optimizasyon problemlerinde kullanılan bir algoritma yöntemidir (Thomas vd, 2009). Bu yöntem, problemin çözüm uzayını ağaç yapısı şeklinde temsil etmektedir. Bu ağaç yapısı üzerinde dallandırma ve sınırlama işlemleri gerçekleştirilmektedir. Yöntemde, her bir aşamada, problemin en iyi çözümüne ulaşma potansiyeline sahip olan dallar seçilir ve diğer dallar kesilir. Bu sayede, gereksiz hesaplamalar önlenerek çözüm süresi ve hesaplama maliyeti azaltılmaktadır. Dal ve sınır yöntemi, genellikle optimizasyon problemlerinde verimli ve kesin çözümler sağlamak için kullanılan bir algoritma yöntemidir.



Şekil 4.1. GAMS 0-1 matris örneği

Elde edilen sonuçların ve sürelerin örnek bir gösterimi Şekil 4.2'de gösterilmiştir.



Şekil 4.2. GAMS sonuç matrisi ve çözüm tablosu

Test için farklı boyutlarda 81 problem seti üretilmiştir. Üretilen problemlerin karakteristik özellikleri Bölüm Altı'da detaylı olarak ele alınmıştır. Üretilen problem setlerindeki değerler sırasıyla problemdeki eleman sayısını (m) ve problemin alt küme sayısını (n) temsil etmektedir. Bu problemlerin çözümleri, GAMS yazılımının Cplex çözücüsü kullanılarak elde edilmiştir. GAMS yazılım lisansından kaynaklı olarak süre konusunda bir limit vardır. 16 dakikanın üstündeki çözümlerde mevcut en iyi değerini sonuç olarak sunmaktadır.

Tablo 4.1, Tablo 4.2 ve Tablo 4.3'te üç farklı tipteki problem setlerinin GAMS ile çözüm sonuçları ve dakika, saniye ve milisaniye cinsinden çözüm süreleri verilmektedir.

Tablo 4.1. Birinci grup GAMS hesaplama denemeleri sonuçları

No	Problem Size (m x n)	GAMS Çözüm	GAMS Süre
1	10-20(1)	3	0:00:00.283
2	10-20(2)	3	0:00:00.305
3	10-20(3)	3	0:00:00.224
4	10-20(4)	4	0:00:00.280
5	10-20(5)	4	0:00:00.267
6	10-20(6)	4	0:00:00.291
7	25-100(1)	3	0:00:01.461
8	25-100(2)	3	0:00:00.216
9	25-100(3)	3	0:00:00.307
10	25-100(4)	5	0:00:00.287
11	25-100(5)	4	0:00:00.350
12	25-100(6)	3	0:00:00.282
13	50-400(1)	3	0:00:00.869
14	50-400(2)	4	0:00:00.507
15	50-400(3)	4	0:00:00.429
16	50-400(4)	5	0:00:00.658
17	50-400(5)	4	0:00:01.247
18	50-400(6)	4	0:00:02.367
19	100-1600(1)	4	0:00:05.862
20	100-1600(2)	4	0:00:06.021
21	100-1600(3)	5	0:16:40.434
22	100-1600(4)	14	0:16:40.460
23	100-1600(5)	9	0:16:40.590
24	200-6400(1)	17	0:16:40.621
25	200-6400(2)	12	0:16:40.747
26	200-6400(3)	10	0:16:40.705
27	200-6400(4)	16	0:16:40.640

Tablo 4.2. İkinci grup GAMS hesaplama denemeleri sonuçları

No	Problem Size (m x n)	GAMS Çözüm	GAMS Süre
1	10-100(1)	2	0:00:00.718
2	10-100(2)	2	0:00:00.226
3	10-100(3)	2	0:00:00.224
4	10-100(4)	4	0:00:00.292
5	10-100(5)	4	0:00:00.374
6	10-100(6)	3	0:00:00.292
7	25-250(1)	3	0:00:00.294
8	25-250(2)	3	0:00:00.208
9	25-250(3)	3	0:00:00:212

Tablo 4.2. devamı

No	Problem Size (m x n)	GAMS Çözüm	GAMS Süre
10	25-250(4)	4	0:00:00.352
11	25-250(5)	5	0:00:00.302
12	25-250(6)	3	0:00:00.277
13	50-500(1)	3	0:00:00.614
14	50-500(2)	3	0:00:00.299
15	50-500(3)	4	0:00:00.587
16	50-500(4)	6	0:00:08.059
17	50-500(5)	7	0:00:00.816
18	50-500(6)	4	0:00:01.972
19	100-1000(1)	4	0:00:02.822
20	100-1000(2)	4	0:00:02.360
21	100-1000(3)	5	0:16:40.555
22	100-1000(4)	9	0:16:40.528
23	100-1000(5)	6	0:16:40.467
24	200-2000(1)	5	0:16:40.355
25	200-2000(2)	5	0:16:40.360
26	200-2000(3)	17	0:16:40.417
27	200-2000(4)	12	0:16:40.367

Tablo 4.3. Üçüncü grup GAMS hesaplama denemeleri sonuçları

No	Problem Size (m x n)	GAMS Çözüm	GAMS Süre
1	10-200(1)	2	0:00:00.224
2	10-200(2)	2	0:00:00.223
3	10-200(3)	2	0:00:00.231
4	10-200(4)	4	0:00:00.299
5	10-200(5)	4	0:00:00.300
6	10-200(6)	5	0:00:00.273
7	25-500(1)	3	0:00:00.215
8	25-500(2)	3	0:00:00.212
9	25-500(3)	3	0:00:00.213
10	25-500(4)	4	0:00:00.353
11	25-500(5)	4	0:00:00.849
12	25-500(6)	5	0:00:00.292
13	50-1000(1)	3	0:00:02.388
14	50-1000(2)	4	0:00:00.317
15	50-1000(3)	5	0:00:01.282
16	50-1000(4)	5	0:00:05.285
17	50-1000(5)	6	0:00:02.876
18	50-1000(6)	5	0:00:45.786

Tablo 4.3. devamı

No	Problem Size (m x n)	GAMS Çözüm	GAMS Süre
19	100-2000(1)	4	0:00:17.313
20	100-2000(2)	4	0:00:15.985
21	100-2000(3)	6	0:16:40.516
22	100-2000(4)	10	0:16:40.492
23	100-2000(5)	10	0:16:40.558
24	200-4000(1)	12	0:16:40.481
25	200-4000(2)	12	0:16:40.482
26	200-4000(3)	11	0:16:40.534
27	200-4000(4)	18	0:16:40.551

4.3 Sezgisel Yöntemler

Sezgisel yöntemler, kesin bir çözüm bulmanın hesaplama açısından maliyetli veya mümkün olmadığı durumlarda, karmaşık optimizasyon problemlerine yaklaşık çözümler bulmayı amaçlayan problem çözme yöntemleridir. Büyük veri kümelerini, yüksek boyutlu uzayları veya küme örtüsü problemi gibi kombinatoryal optimizasyon problemlerinin üstesinden gelmek için bilgisayar bilimlerinde çeşitli alanlarda yaygın olarak kullanılmaktadırlar. Kesin matematiksel formülasyonların yerine pratik kurallara veya sezgisel yöntemlere dayanmaktadırlar.

Sezgisel yöntemler, algoritmanın arama sürecini yönlendirerek problemin çözüm uzayını daha etkili bir şekilde keşfetmeyi sağlayan kısa yollar sunar. Bu algoritmalar optimal bir çözümü garanti etmemektedir, ancak genellikle makul bir süre içinde yaklaşık sonuçlar vermektedir. Sezgisel yöntem algoritmaları; esnektir, uyarlanabilirdir ve belirli problem uzaylarına uygulanabilmektedir, bu da bu yöntemleri karmaşık gerçek dünya problemlerini çözmede değerli araçlar haline getirir (Chvatal, 1979).

Sezgisel algoritmaların temel fikri, bilgisayarların matematiksel yöntemlerle çözümleyemediği karmaşık problemleri, biyolojik, fizyolojik ve sosyal sistemlerde gözlenen davranışları taklit ederek çözmektir. Bu algoritmalar; A* (A yıldız) araması, tepe tırmanma, en iyi öncelikli arama, açgözlü en iyi öncelikli arama ve geri izleme

algoritmalarıdır. Bu algoritmaların başarısı, uygulama alanlarına ve problem türüne göre değişebilmektedir. Çözümü araştırılan bir problem için doğru yöntemi seçmek bu yüzden çok önemlidir.

4.3.1 Dinamik Açgözlü Yöntemi ile Çözüm

Problemin çözüm süresini olabildiğince minimum sürelerde tutarak, optimuma yakın çözüm vermeyi amaçlamaktadır. Dinamik açgözlü yönteminde problemin çözüm süresinin daha az sürede çözülmesi için; problemde tanımlanan alt kümelerin, eleman sayıları büyükten küçüğe doğru sıralanarak oluşturulan yeni sıra düzeni ile problemin çözüm uzayına eklenecek kümeler seçilmektedir.

Açgözlü yöntem çözüm zamanını azaltmaya yönelik çalışan bir algoritmadır. Bu durum problemlerde sapmalara sebep olmaktadır. Aynı alt deseni içeren problemlerde, problem boyutu arttıkça sapma sayısı da doğru orantılı olarak artmaktadır. Durum 1 ve Durum 2'de eleman artışlarına bağlı sapma miktarları verilmiştir.

- Durum 1:

Elemanlar: {1,2,3,4,5} Alt Kümeler: $\{S_1, S_2, S_3, S_4\}$

S_1: {1,2,3}
S_2: {1,2,5}
S_3: {3,4}
S_4: {5}

Problemin bu boyutunda optimum çözüm S_2 ve S_3 alt kümelerinin birleşimidir. Fakat dinamik açgözlü yöntem ile çözüldüğünde çözüm kümesi S_1, S_2ve S_3'ten oluşmaktadır. Dinamik açgözlü yöntemde çözüm kümesinin optimumdan 1 saptığı görülmektedir.

- Durum 2:

Elemanlar: {1,2,3,4,5,6,7,8,9,10} Alt Kümeler: $\{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8\}$

S_1: {1,2,3}
S_2: {1,2,5}
S_3: {3,4}
S_4: {5}
S_5: {6,7,8}
S_6: {6,7,10}
S_7: {8,9}
S_8: {10}

İlk sıralama adımında;
S_1: {1,2,3}
S_2: {1,2,5}
S_5: {6,7,8}
S_6: {6,7,10}
S_3: {3,4}
S_7: {8,9}
S_4: {5}
S_8: {10}

Durum 2'de, Durum 1'den farklı olarak 5 eleman ve 4 altküme eklenmiştir. Problem incelendiğinde optimum çözüm kümesi S_2, S_3, S_6 ve S_7'den oluşmaktadır. Bu durumda optimum çözüm kümesinin sayısı 4'tür. Problem dinamik açgözlü yöntemi ile çözüldüğünde çözüm kümesinin S_1, S_2, S_3, S_5, S_6 ve S_7 olduğu görülmektedir. Sonuç olarak sapma sayısı 2 olmaktadır.

Her iki durumdaki örnekler incelendiğinde eleman sayısının artmasının sapma miktarının da artmasına sebep olduğu gözlenmiştir.

4.3.2 Statik Açgözlü Yöntemi ile Çözüm

Statik açgözlü yöntemde de genel açgözlü yöntemler gibi, problemin çözüm süresini azaltarak optimuma yakın sonuçların elde edilmesi amaçlanmaktadır. Bu yöntemde dinamik açgözlü yöntemden farklı olarak; problem setinde verilen kümeler sadece bir kez eleman sayısına göre sıralamasına göre sıralanmaktadır. Problemin çözüm uzayına eklenecek kümeler indis sıralamasına göre seçilmektedir.

Açgözlü yöntem çözüm zamanını azaltmaya yönelik çalışan bir algoritma olduğundan bu durum zor problemlerde sapmalara sebep olmaktadır. Aynı alt deseni içeren problemlerde, problem boyutu arttıkça sapma sayısı da doğru orantılı olarak artmaktadır. Durum 1 ve Durum 2'de eleman artışına bağlı sapma miktarları örneklerde görülmüştür.

- Durum 1:

Elemanlar: {1,2,3,4} Alt Kümeler: $\{S_1, S_2, S_3, S_4, S_5\}$

S_1: {1,2}
S_2: {1,4}
S_3: {2,3}
S_4: {3,4}
S_5: {4}

Problemin bu boyutunda optimum çözüm S_1 ve S_4 kümesinin birleşimidir. Fakat statik açgözlü yöntemi ile çözüldüğünde çözüm kümesi S_1, S_2 ve S_3'ten oluşmaktadır. Statik açgözlü yöntemde çözüm kümesinin optimumdan 1 saptığı görülmektedir.

- Durum 2:

Elemanlar: {1,2,3,4,5,6,7,8} Alt Kümeler: $\{S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$

S_1: {1,2,3}

S_2: {1,2,5}

S_3: {4,5}

S_4: {5,6}

S_5: {6,7,8}

S_6: {6,7}

S_7: {8}

Sıralama sonrası;

S_1: {1,2,3}

S_2: {1,2,5}

S_5: {6,7,8}

S_3: {4,5}

S_4: {5,6}

S_6: {6,7}

S_7: {8}

Durum 2'de Durum 1'den farklı olarak 4 eleman ve 3 küme eklenmiştir. Problem incelendiğinde optimum çözüm kümesi S_1, S_3 ve S_5'ten oluşmaktadır. Bu durumda optimum çözüm kümesinin sayısı 3'tür. Problem statik açgözlü yöntemi ile çözüldüğünde çözüm kümesinin S_1, S_2, S_3 ve S_5 olduğu görülmektedir. Sonuç olarak sapma sayısı 1 olmaktadır.

Her iki durumdaki örnekler incelendiğinde eleman sayısının artması ile sapma miktarının da artmasına sebep olduğu gözlenmiştir.

4.4 Meta-Sezgisel Yöntemler

Meta-sezgisel yöntemler, doğal süreçlerden esinlenerek geliştirildiği için, evrimsel ve kolektif davranışları taklit ederek, optimize edilmesi zor problemleri çözme potansiyeline sahiptirler. Meta-sezgisel yöntemler, karmaşık optimizasyon problemlerini çözmek amacıyla geliştirilen, yüksek seviyeli problem çözme

yöntemleridir. Bu yöntemler, çözüm uzaylarını keşfetme sürecine rehberlik ederek yaklaşık çözümler aramaktadırlar. Belirli bir sorunu hedefleyen spesifik sezgisel algoritmaların aksine, meta-sezgiseller çok çeşitli problem alanlarına uygulanabilen genel bir çerçeve sağlamaktadır. Farklı sezgisel yöntemleri birleştirip uyarlayarak veya yenilikçi arama yöntemleri sunarak çözüm uzayını verimli bir şekilde keşfetmeyi ve en iyi çözüme yaklaşmayı amaçlamaktadırlar.

Meta-sezgisel yöntemler genellikle evrimsel süreçlerden, sürü zekâsından veya benzetilmiş tavlama gibi doğal olaylardan veya sosyal davranışlardan ilham almaktadır. Bu yöntemlerde, genellikle bir popülasyon tabanlı iteratif bir yapı kullanır. Her iterasyonda, çözüm adaylarının bir popülasyonu değerlendirilmektedir. Bu değerlendirmeler sonucunda uygunluk veya amaç fonksiyonuna bağlı olarak çözümleri iyileştirmektedir. Daha iyi olan çözüm adayları bir sonraki popülasyon için seçilmektedir. Bu iterasyonlar, belirli bir sonlandırma kriterine veya maksimum iterasyon sayısına kadar devam etmektedir. Meta-sezgisel yöntemler, karmaşık arama alanlarında iyi çözümler bulmak için keşif ve kullanım stratejilerini akıllıca birleştirirler.

Meta-sezgisel yöntemler, genellikle sezgisel algoritmaların optimizasyon yöntemlerinin birleştirilmesiyle oluşturulmaktadır. Genetik algoritmalar, parçacık sürü optimizasyonu, karınca kolonisi optimizasyonu, tavlama benzetimi gibi farklı algoritmalar meta-sezgisel yöntemlere örnek olarak verilebilmektedir. Bu algoritmalar, genellikle probleme özgü parametrelerin otomatik olarak ayarlanması, birden fazla algoritmanın sonuçlarının birleştirilmesi veya yeni arama stratejilerinin geliştirilmesi gibi avantajlar sağlamaktadır.

Meta-sezgisel yöntemler, birçok uygulama alanında başarıyla kullanılmaktadır. Bu yöntemler, karmaşık optimizasyon problemlerini çözmek için etkili bir araç olarak kabul edilmektedir. Özellikle büyük veri setleri, yüksek boyutlu arama alanları veya kombinatoryal optimizasyon gibi zorlu problemlerde başarılı sonuçlar elde edilmektedir. Meta-sezgisel yöntemler, problem çözme sürecinde esneklik, uyarlanabilirlik ve çeşitli algoritmaların kombinasyonuyla avantajlar sunmaktadır. Bu nedenle, meta-sezgisel yöntemler, gerçek dünya problemlerinin etkili bir şekilde

çözümü için önemli bir araçtır.

Bu tez çalışmasında kullanılan meta-sezgisel yöntem genetik algoritma tekniğidir. Bir sonraki bölümde genetik algoritmadan ve bu tez çalışmasında kullanılan parametrelerden bahsedilecektir.

BÖLÜM BEŞ
GENETİK ALGORİTMA

Genetik algoritma, optimizasyon problemlerini çözmek için kullanılan bir meta-sezgisel arama yöntemidir. Genetik algoritmalar, doğal seleksiyon ve genetik süreçlerden ilham alarak, bu süreçleri simüle etmektedirler. Bu algoritma, biyolojik evrimin prensiplerini taklit ederek çözüm adaylarının genetik materyallerini manipüle etmeyi ve daha iyi çözümlerin ortaya çıkmasını amaçlamaktadır (Davis, 1994).

Genetik algoritma, popülasyon tabanlı bir yaklaşımı benimsemektedir. Temel amacı; bir popülasyon içerisindeki çeşitli çözüm adaylarının genetik materyallerinin kombinasyonları ile yeni ve daha iyi çözüm adayları üretmektir. Bir popülasyon içerisindeki bireyler, problemi temsil eden kromozomlar olarak ifade edilmektedir. İlk olarak, bir başlangıç popülasyonu oluşturulur, her bir birey bir çözüm adayını temsil eder. Bu bireylerin genetik materyali genellikle gen veya bit dizisi olarak ifade edilir. Ardından popülasyondaki bazı bireyler, çeşitli genetik operatörler aracılığıyla manipüle edilerek yeni çözüm adayları üretilmektedir.

Manipülasyon yöntemleri evrimsel süreci oluşturmaktadır. Bu süreçler, çaprazlama, mutasyon ve seçilim adı verilen genetik operatörler aracılığıyla gerçekleşmektedir. Çaprazlama operatörü, iki veya daha fazla bireyin genetik özelliklerinin belirli bir veya birkaç noktadan kesilip birleştirilmesiyle yeni çözüm adayları üretmektedir. Mutasyon ise rastgele olarak kromozomdaki bazı genlerin veya bitlerin değiştirilmesini sağlamaktadır. Bu da popülasyon içindeki çeşitliliği arttırmaktadır. Seçilim operatörü ise, her bir çözüm adayı bireyin uygunluk değeri değerlendirilir ve en uygun çözümler (yani bireyler) gelecek nesillere aktarılır. Bu şekilde, genetik algoritma, bir nesilden diğerine geçerek problem çözme sürecini evrimsel bir şekilde yönlendirerek daha iyi çözümleri bulmayı amaçlar.

Her bir evrimsel iterasyonda, çözüm adayları değerlendirilir, çaprazlama ve mutasyon operatörleri uygulanır ve yeni bir nesil oluşturulur. Bu işlem, belirli bir sonlandırma kriteri veya belirli bir sayıda iterasyon gerçekleşene kadar tekrar

etmektedir. Genetik algoritma, bu iterasyonlar boyunca çözüm adaylarını geliştirerek, problemi çözmek için en iyi çözüme yaklaşmayı amaçlamaktadır.

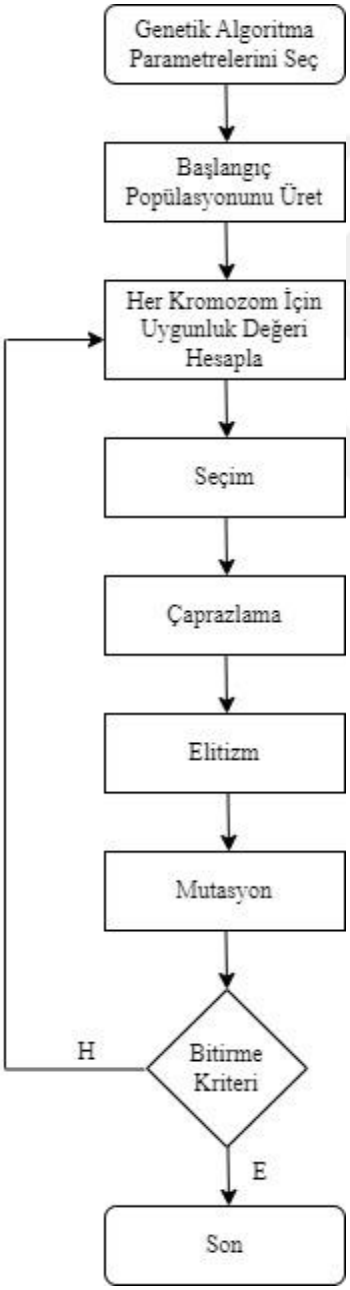
Genetik algoritma, geniş bir uygulama alanına sahip olup, kombinatoryal optimizasyon, veri madenciliği, yapay zeka ve mühendislik gibi birçok alanda başarıyla kullanılmaktadır. Özellikle büyük veri setleri, karmaşık sistemler, kombinatorik problemler gibi zorlu problemlerde başarılı bir şekilde sonuç elde etmektedir.

Genetik algoritmanın adımları şu şekildedir.

Adım 1. Olası çözümlerden oluşan bir başlangıç popülasyonu oluşturulur. Popülasyonda bulunacak birey sayısı için bir standart yoktur, problemin tipine veya büyüklüğüne göre değişebilmektedir.
Adım 2. Her bir kromozomun (bireysel) uygunluk değeri, amaç fonksiyonu kullanılarak hesaplanmaktadır. Bu adımda, en iyi birey seçilir ve yerel çözüm olarak tutulur. Seçim için rulet tekerleği seçimi, turnuva seçimi gibi seçim yöntemleri kullanılmaktadır.
Adım 3. Seçilen kromozomlar (bireyler) çaprazlanarak ve mutasyona uğratılarak yeni bir popülasyon oluşturulur.
Adım 4. Durdurma kriterleri karşılanana kadar Adım 2'den itibaren tüm adımlar tekrarlanır. Bitirme kriteri, belirli sayıda popülasyon veya yineleme sayısı olabilmektedir.
Adım 5. Çalışma sırasında bulunan en iyi kromozom (bireysel) sonuç olarak kabul edilmektedir.

Şekil 5.1'de genetik algoritmaya ait akış şeması verilmiştir. Genetik algoritmaya başlamadan önce uygun parametre değerleri seçilmelidir. Ardından başlangıç popülasyonu üretimi sağlanmalıdır. Bu tez çalışmasında başlangıç popülasyonu %25 oranında iyi katkı yapacak alt kümelerin seçim formülü ile belirlenmesi ve %25 oranında bu alt kümelerin frekanslarına göre turlama yöntemi kullanılması ve %50 oranında rastgele bireylerin seçilmesi ile oluşturulmaktadır. Sonrasında rulet tekerleği

yöntemi kullanılarak ebeveyn seçimi yapılmıştır. Seçilen ebeveynlerin bir sonraki nesle genlerini aktarabilmesi için orantılı tek düze çaprazlama yöntemi kullanılmıştır. Ardından elitizim uygulanarak popülasyondaki %50 oranında en iyi bireylerin seçilerek bir sonraki nesle aktarımı gerçekleşmiştir. En son adımda popülasyondaki bireylerin %1'ine ve rastgele seçilen bir genine tersine çevirme mutasyon operatörü uygulanmıştır. Tüm bu adımlar bir durdurma kriteri sağlanana kadar (bu tez çalışması için durdurma kriteri iterasyon sayısı olarak verilmiştir.) devam etmektedir.



Şekil 5.1. Genetik algoritma akış şeması

5.1 Genotip Kodlama

Genotip kodlama, çözülmekte olan problemin doğasını ve problemin özelliklerini en iyi şekilde yansıtan temsilidir. Genotip kodlama, bir bireyin(kromozomun) genetik özelliklerinin bir veri yapısı veya bir dizi sembol biçiminde temsil edilmesini ifade etmektedir. Probleme özgü bilgilerin genetik algoritma tarafından arama alanının verimli bir şekilde manipüle edilebilecek ve işlenebilecek bir formata dönüştürülmesidir. Genotip kodlama, popülasyonun çeşitliliğini, uygulanabilecek genetik operatörlerin türlerini ve optimale yakın çözümler bulmada algoritmanın genel performansını doğrudan etkilemektedir. Birçok kodlama yöntemi vardır.

5.1.1 İkili Kodlama

Genetik algoritmalarda kullanılan en yayın kodlama tekniğidir. Bu tekniğin temsilinde genotip bit dizilerinden oluşmaktadır. Çözüm uzayı, mantıksal karar değişkenlerinden yani değeri 0 veya 1 olan ikililerden oluşmaktadır. Bu gösterim biçiminde, her bir öğenin çözüm kümesine dahil olup olmadığı 0 ve 1 değerleri ile belirtilmektedir. Şekil 5.2'de ikili kodlama gösterimi verilmiştir. Bu tez çalışmasında, 0-1 ikili kodlama yöntemi kullanılmıştır. Kromozomdaki gen değeri 1 ise küme çözüme dahil edilir, 0 ise çözüme dahil edilmemektedir. Aynı şekilde kümelerdeki elemanlar da 0-1 şeklinde kodlanmıştır. Eleman kümede ise 1, kümede değilse 0 olarak alınır.

0	0	1	0	1	1	1	0	0	1

Şekil 5.2. İkili kodlama örneği

5.1.2 Tam Sayılı Kodlama

Çözüm uzayının her zaman ikili değerli olmadığı yani evet veya hayır olarak sınıflandırılmayan problem tiplerinde kullanılmaktadır. Tam sayılı kodlamada, kromozomun her bir geninin temsil ettiği değer belirli bir aralıktaki tamsayı

değerlerinden oluşmaktadır. Örneğin bir optimizasyon probleminde dört yönün - kuzey, güney, doğu, batı- kullanıp kodlanması gerekiyorsa bunların {1,2,3,4} olarak temsil edilmesi gerekmektedir. Şekil 5.3'te tam sayılı kodlama gösterimi verilmiştir.

1	2	3	4	3	2	4	1	2	1

Şekil 5.3. Tam sayılı kodlama örneği

5.1.3 Gerçel Sayılı Kodlama

Gerçel sayılı kodlamada, kromozomun genleri sürekli değerler alabilmektedir. Her bir gen, belirli bir aralıkta gerçel bir sayı ile temsil edilmektedir. Kromozom, gerçel değerlerin bir dizisinden oluşmaktadır. Şekil 5.4'te gerçel sayılı kodlama gösterimi verilmiştir.

0.5	0.2	0.6	0.8	0.7	0.4	0.3	0.2	0.1	0.9

Şekil 5.4. Gerçel sayılı kodlama

5.1.4 Permütasyon Kodlama

Permütasyon kodlama, bir dizi elemanın belirli bir sırada düzenlenmesini içeren problemler için kullanılır. Kromozomdaki her bir gen, sıralanması gereken bir öğeyi temsil etmektedir. Her bir genin değeri, permütasyondaki elemanın konumuna veya indeksine karşılık gelmektedir. Örneğin, N eleman varsa, kromozomdaki genler, elemanların konumlarını temsil eden 1'den N'ye kadar değerler alacaktır.

Permütasyon kodlamanın temel özelliği, her genin kromozom içinde benzersiz bir değere sahip olmasıdır. Böylece, her öğenin permütasyonda tam olarak bir kez görünmesi sağlanmış olmaktadır. Şekil 5.5'te permütasyon kodlama gösterimi verilmiştir.

1	5	9	8	7	4	2	3	6	0

Şekil 5.5. Permütasyon kodlama

5.2 Popülasyon

Bir popülasyon, bir nesildeki çözümlerin kümesidir. Nüfus sayısı belirlenirken dikkat edilmesi gereken hususlar vardır. Nüfus çeşitliliği korunmalıdır. Aksi takdirde, erken yakınsamaya yani belirli bir çözüm noktasına sıkışıp kalmasına yol açabilmektedir. Popülasyonun büyüklüğü, fazla olursa genetik algoritmanın yavaşlamasına neden olacaktır. Ayrıca popülasyon büyüklüğünün küçük seçilmesi de çaprazlama için yeterli olmayabilir. Bu nedenle bu tezde popülasyon büyüklüğü, problemin küme sayısına göre deneme yanılma yoluyla belirlenmiş ve bu yöntemin en iyi olduğuna karar verilmiştir.

Genetik algoritmada başlangıç popülasyonunu oluşturmak için iki temel yöntem vardır. Bunlardan ilki popülasyonun rastgele üretilmesi, diğeri ise sezgisel algoritma ile oluşturulmasıdır. Çözümün optimum sonuçlara ulaşmasını sağlamak için sezgisel yaklaşımlar kullanılır.

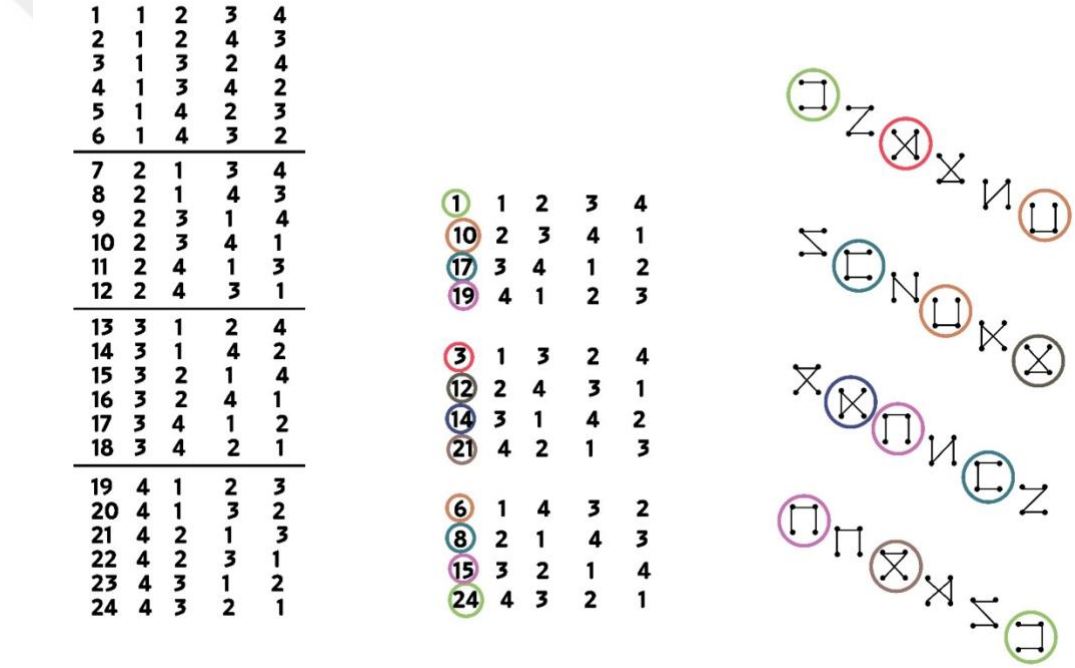
Rastgele seçim, her bir bireyin genetik materyalini oluşturmak için kullanılır. Genellikle, bu genetik materyal bir dizi gen veya bit olarak ifade edilir. Her gen, problem alanına uygun bir şekilde rastgele bir değerle veya aralıktan seçilen bir değerle atanmaktadır. Örneğin, bir optimizasyon problemi durumunda, her bir gen belirli bir parametreyi veya karar değişkenini temsil edebilir. Başlangıç popülasyonunun boyutu, problem özelliklerine bağlı olarak belirlenmektedir.

Çözümün optimum sonuca ulaşmasını engelleyen problemlerden biri de aynı eleman sayısına sahip kümelerden hangisinin önce çözüme alınacağıdır. Problemin doğası gereği çözüme bir küme eklendiğinde diğer kümelerin çözüme dahil edilmesini engelleyebilir. Eşitlik anında iyi çözüme ulaşmayı engellememek için çözüm üretme sıralarını manipüle eden bir yöntem geliştirilmiştir. Bu tekniğe turlama yöntemi denir (Gencer, 2017). Turlama yöntemi; indeks vektörünü dairesel bir kuyruğa dönüştürerek, her adımda farklı bir başlangıç noktasından başlayarak çözüm uzayını akıllı bir şekilde taramaktan ibarettir. Şekil 5.6'daki örnekte olduğu gibi [1,2,3,4,5] olarak elde edilen indeks vektörüne turlama yöntemi uygulandığında aşağıdaki diziler

elde edilmektedir. Şekil 5.7'de turlama yöntemine ait çalışma prensibi daha detaylı ele alınarak incelenmiştir. 4 elemandan oluşan bir örnek verilmiştir. Normal şartlar altında *4!*'den 24 farklı durum oluşması gerekirken turlama tekniği yöntemi sayesinde permütasyon uzayının sadece bir kısmı *n*(n-1)* kadarı seçilmiştir. Böylece sıralama sırasında karşılaşılan eşitlik durumlarında öncelik-sonralık tercihinden kaynaklanan olumsuz durumların sayısı azaltılmaktadır.

[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]

Şekil 5.6. Turlama yöntemi çalışma prensibi



Şekil 5.7. Turlama yöntemi çalışma prensibi örnek problem

Bu tez çalışmasında, başlangıç popülasyonu %50 oranında rastgele bireylerden seçilerek, %25 oranında turlama tekniğinden ve %25 oranında çözüm vektörüne iyi katkı sağlayan alt kümelerin seçiminden oluşturulmaktadır.

$$\boldsymbol{nr_i = fa_{i,0} + \sum_{j=1}^{fa_{i,0}} taban^{(mf_0 - mf_{fa_{i,j}})}} \qquad (\mathbf{5.1})$$

Eşitlik 5.1'de verilen seçim formülü; çözüm vektörüne dahil olacak alt kümelerin fazla eleman içermesi ve nadir eleman içermesi kriterlerini birleştirmek amacıyla ortaya çıkmıştır. Taban değeri deneysel çalışmalar sonucunda 10 olarak ele alınmıştır. $\boldsymbol{fa_{ij}}$, n sütun ve m satırdan oluşan problem veri setinin tutulduğu matristir. $\boldsymbol{mf}$ vektörü elemanların frekanslarının tutulduğu vektördür. $\boldsymbol{mf_0}$, tüm eleman değerine bölünür. Böylece elemanların ortalama frekans değerleri hesaplanmış olmaktadır. $\boldsymbol{nr_i}$ *i.* alt kümeye ait seçim değerlerini tutmaktadır.

İlgili alt kümeye ait bir eleman, tüm problem veri seti içinde nadir olarak görülüyorsa frekansı ortalama frekanstan oldukça düşük olacaktır. Dolayısıyla bu elemanın çözüm vektörüne katkısı oldukça fazla olacaktır. Diğer yandan, problem veri setinde sık görülen bir elemanın frekansı ortalama frekanstan oldukça büyük olacağı için üs negatif olarak gelecek ve taban değeri ne olursa olsun katkısı 0 ile 1 arasında olacaktır. Durum 1'de seçim formülünün çalışma prensibine ters olacak örnek bir problem verilmiştir.

- Durum 1:

Elemanlar: {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}
Alt Kümeler: $\{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}\}$

S_1: {1}
S_2: {2,3}
S_3: {4,5,6}
S_4: {7,8,9,10}
S_5: {11,12,13,14,15}
S_6: {1,2,4,7,11}
S_7: {1,3,5,8,12}
S_8: {6,9,10,13}
S_7: {7,14,15}
S_4: {2,3,6,15}

Durum 1'de 15 elemanlı 10 alt kümeden oluşan bir problem seti verilmiştir. Tüm elemanların problem setinde tekrar etme değerleri 2 ya da 3'tür. Seçim formülü uygulandıktan sonra $\boldsymbol{nr_i}$ değerleri tüm alt kümeler için sırasıyla,

nr_1: 1.251189
nr_2: 2.502378
nr_3: 8.274962
nr_4: 11.786849
nr_5: 15.298736
nr_6: 10.777339
nr_7: 13.038037
nr_8: 11.786849
nr_9: 6.014265
nr_{10}: 5.004755

şeklinde olmaktadır. Büyükten küçüğe sıralama yapıldıktan sonra yeni sıralama; nr_5, nr_7, nr_4, nr_8, nr_6, nr_3, nr_9, nr_{10}, nr_2 *ve* nr_1 şeklinde olmaktadır. Bu sıralama sonucunda çözüm kümesine S_5, S_7, S_4, S_8ve S_6 kümeleri alınmaktadır. Çözüm 5 alt küme olarak bulunmaktadır. Ancak problemin optimum çözümü 4 alt kümeden oluşmaktadır ve bu alt kümeler S_5, S_7, S_8ve S_6kümeleridir. Bunun gibi seçim formülünün çalışma prensibine ters düşen örnekler için genetik algoritma devreye girerek optimum sonuçların elde edilmesi amaçlanmıştır.

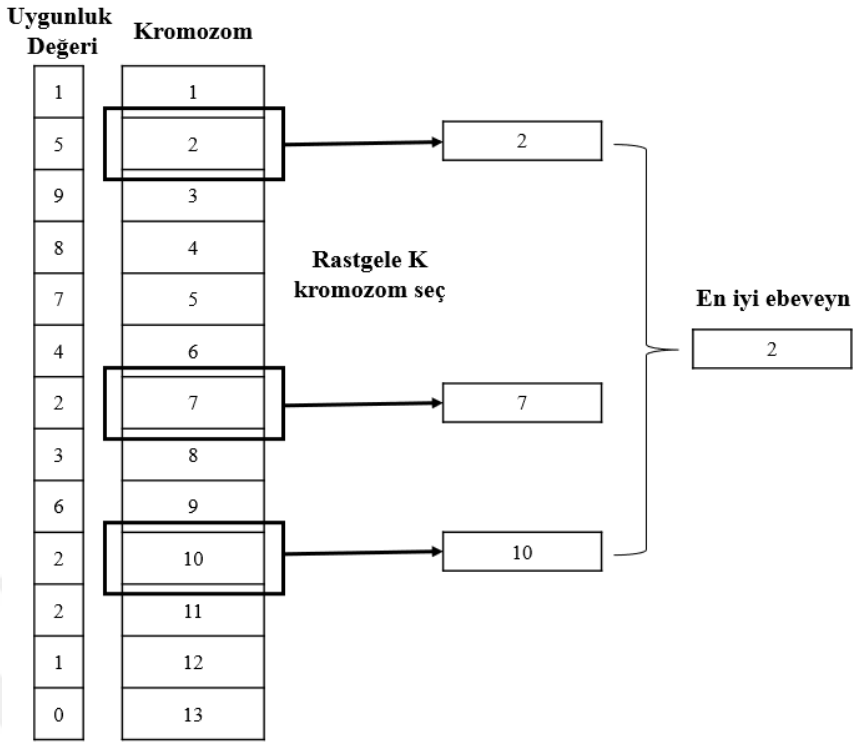
5.3 Seçim Yöntemleri

Ebeveyn seçimi, bir sonraki nesil için yavrular üretmek üzere çaprazlanacak bireylerin seçilmesi sürecidir. Seçilen her birey, uygunluğuyla doğru orantılı olarak çaprazlamada kullanılacak bir ebeveyndir. Uygunluğu yüksek olan bireylerin, özelliklerini bir sonraki nesle aktarma şansı daha yüksektir.

5.3.1 Turnuva Seçim Yöntemi

Turnuva seçim yönteminde, popülasyon içerisinden rastgele *K* adet birey seçilmektedir. Seçilen bu bireyler uygunluk fonksiyonlarına ya da amaç fonksiyonuna göre sıralanmaktadır. Sıralanan bu bireylerin içinden en uygun olan bireyler, bir sonraki nesli oluşturacak bireylerin ebeveynleri olarak çaprazlanma işlemi için seçilmektedir. Turnuva seçim yöntemi negatif uygunluklu değerler için de çalışabilmektedir (Demirel, 1999).

Turnuva seçim yöntemi, her turda yalnızca popülasyonun bir alt kümesinin değerlendirilmesini gerektirdiğinden, hesaplama açısından verimli bir yöntemdir. Popülasyondaki çeşitliliği korumaya yardımcı olur, iyi çözümlerin yayılarak bir sonraki nesillere aktarılmasını kolaylaştırır ve algoritmanın genel yakınsamasını ve performansını geliştirir. Küçük turnuva boyutu, zayıf bireylerin ara sıra kazanmasına izin vererek çeşitlilik sağlar ve erken yakınsamayı önlemektedir. Aynı zamanda, uygun bireylerin kazanan olarak seçilme olasılığı daha yüksek olduğundan, seçim baskısı artmaktadır. Şekil 5.8'de turnuva seçim yöntemine örnek verilmiştir. Bir popülasyon içinden rastgele *K* adet (Üç adet) kromozom seçilmiştir. Seçilen kromozomlar bir turnuvaya sokularak uygunluk değerleri kıyaslanmıştır. Bu üç bireyden çaprazlamaya girecek olan ilk ebeveyn birey 2 numaralı kromozom olarak seçilmiştir.

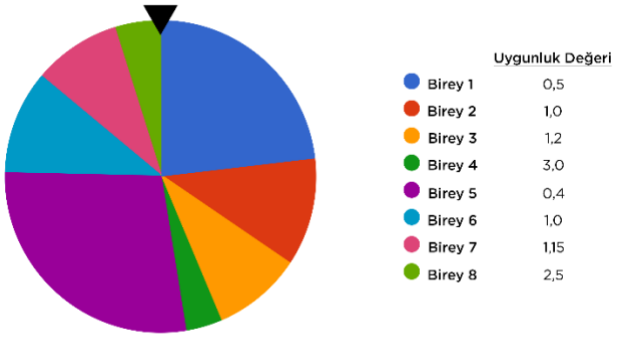


Şekil 5.8. Turnuva seçim yöntemi

5.3.2 Rulet Tekerleği Seçim Yöntemi

Bu tez çalışmasında ebeveyn seçimi için rulet tekerleği seçim yöntemi kullanılmıştır. Maksimum problemlerinde, uygunluk değeri yüksek olan rulet tekerleği üzerinden daha fazla pay almaktadır. Fakat küme örtüsü problemi minimum problemi olduğu için uygunluk değeri düşük olan rulet tekerleği üzerinden daha fazla pay alacaktır.

Bu seçim yönteminde bireylerin seçilme olasılıkları uygunluk fonksiyonları ile orantılıdır. Şekil 5.9'da görüldüğü gibi, popülasyondaki bireyler uygunluklarına göre orantılı olarak tekerleğe yerleştirilir. Tekerleğin etrafında sabit bir nokta seçilerek döndürülür. Sabit nokta ile çakışan birey, çaprazlama için seçilen ilk birey olur. Aynı şekilde tekerlek ikinci kez çevrilerek ikinci birey elde edilir.



Şekil 5.9. Rulet tekerleği seçim yöntemi

Eşitlik 5.2, rulet tekerleği seçim algoritmasındaki bireylerin yüzdelik değerleri hesaplanmaktadır. KÖP bir minimizasyon problemi olduğu için uygunluk değeri küçük olan bireylerin tekerlek üzerinde daha fazla pay alması gerekmektedir. Her bir bireyin, toplam örttüğü alt küme sayısı 100 ile çarpılıp toplam uygunluk değerine bölünmesi ile bireylerin yüzdelik değerleri hesaplanmaktadır.

$$\%\mathrm{i} = \frac{(100 * Uygunluk\ değeri_i)}{(\text{Toplam Uygunluk})} \tag{5.2}$$

5.4 Elitizm

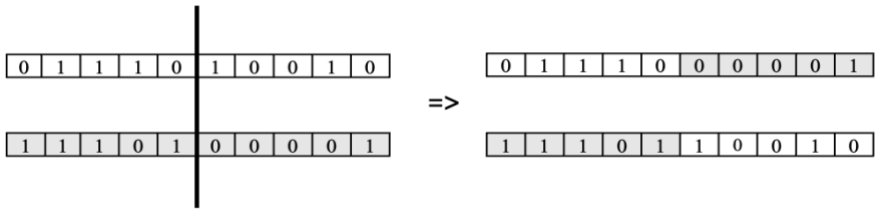
Elitizm, en iyi bireylerin mevcut nesilden sonraki nesle değişmeden geçmesine izin vermektir. Bu tezde, mevcut popülasyondaki bireyler uygunluk değerlerine göre sıralanmıştır. Listelenen bireylerin en iyi %50'si bir sonraki nesle değişmeden aktarılmıştır.

5.5 Çaprazlama

Genetik algoritmada çaprazlama, algoritmanın farklı bireylerden en iyi genleri çıkarmasına ve bunları potansiyel olarak üstün bireylerde birleştirmesine olanak tanır. Bu bireyler melezlenerek bir sonraki nesle aktarılır.

5.5.1 Tek Noktalı Çaprazlama Yöntemi

Tek noktalı çaprazlama yöntemi, yavru bireyler oluşturmak için ebeveyn bireyler arasındaki genetik özellikleri birleştirme operatörüdür. Genetik bilgi alışverişini teşvik eden ve popülasyonu çeşitlendiren üreme sürecinde önemli bir adımdır. Çaprazlama işlemi için seçilen iki ebeveyn bireyin kromozomunda rastgele bir nokta seçilmektedir. Bu noktaya geçiş noktası olarak adlandırılmakta ve ebeveynler arasında değiş tokuş yapılacak genlerin konumunu belirtmektedir. Çaprazlama işlemi bu noktadan itibaren yapılmaktadır. Tek noktalı çaprazlama yönteminin daha iyi anlaşılması Şekil 5.10 verilmiştir. Beyaz renk ebeveyn 1'i gri renk ise ebeveyn 2'yi temsil etmektedir. Rastgele seçilen beşinci genin konumu kesme noktası olarak seçilmiştir. Oluşan yeni bireylerden ilki kesme noktasına kadar olan özelliklerini ebeveyn 1'den, kesme noktasından sonrasındaki özelliklerini ise ebeveyn 2'den almaktadır. Aynı şekilde oluşan yeni ikinci birey ise kesme noktasına kadar olan özelliklerini ebeveyn 2'den alırken kesme noktası sonrasındaki özelliklerini ebeveyn 1'den almıştır.

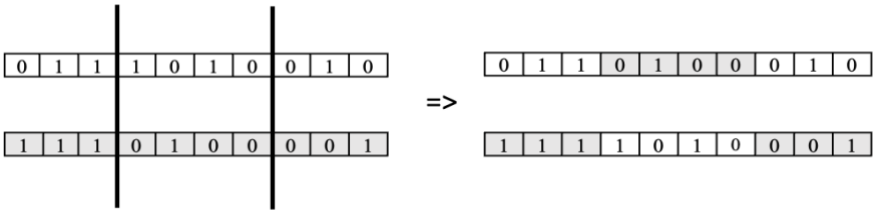


Şekil 5.10. Tek noktalı çaprazlama yöntemi örneği

5.5.2 Çok Noktalı Çaprazlama Yöntemi

Çok noktalı çaprazlama tekniğinde, genetik özellikler ebeveyn kromozomlarının farklı bölümlerinin rekombinasyonu ile gerçekleşmektedir. Böylece, popülasyonda çeşitliliği sağlayarak her iki ebeveynden özelliklerin bir kombinasyonunu miras alan yavru bireylerin yaratılmasını sağlamaktadır. Farklı çaprazlama yöntemleri, genetik özellikleri birleştirmenin farklı yollarını sunar ve belirli problem özellikleri için daha uygun olabilmektedir. Tek noktalı çaprazlama tekniğinden farklı olarak çok noktalı çaprazlama tekniğinde geçiş noktası birden fazla seçilmektedir. Seçilen geçiş

noktalarına bağlı olarak oluşan yeni bireylerde genler ebeveynlerden sırasıyla genetik alışverişin gerçekleşmesiyle alınmaktadır. Şekil 5.11'de çok noktalı çaprazlama örneği olarak verilen iki geçiş noktasına sahip (iki noktalı çaprazlama) bir örnek verilmektedir. Bu örnekte de yine beyaz renk ebeveyn 1'i, gri renk ise ebeveyn 2'yi temsil etmektedir. Rastgele olarak seçilen iki kesme noktasının konumu, üçüncü gen ve altıncı gen olarak seçilmiştir. Oluşan yeni bireylerden ilki ilk kesme noktasına (üçüncü gene) kadar olan özelliklerini ebeveyn 1'den, üçüncü noktadan ikinci kesme noktası olan altıncı gene kadar ise ebeveyn 2'den özelliklerini almaktadır. İkinci kesme noktasından sonra ise kalan özelliklerini yine ebeveyn 1'den alarak genleri tam sağlıklı bir birey oluşturulmuştur. Aynı şekilde oluşan yeni ikinci birey ise yine ilk kesme noktası olan üçüncü gene kadarki özelliklerini ebeveyn 2'den alırken iki kesme noktası arasında kalan dördüncü, beşinci ve altıncı genlerine ait özellikleri ebeveyn 1'den alarak tamamlamıştır. İkinci kesme noktasından sonra kalan özelliklerini ise yine ebeveyn 2'nin gen özelliklerini alarak tamamlamıştır.



Şekil 5.11. Çok noktalı çaprazlama yöntemi örneği

5.5.3 Orantılı Tekdüze Çaprazlama

Bu çaprazlama yönteminde her gen ayrı ayrı incelenmektedir. Seçme yöntemleriyle seçilen kromozomlar (bireyler) orantılı olarak çaprazlanmaktadır. Oran, kromozomun uygunluk değerine göre hesaplanmaktadır. Ortaya çıkan orana bağlı olarak, karşılık gelen genler arasında geçişler yapılır. Çaprazlama sonucunda bir birey oluşur. Uygunluk değerlerini kullanan formüller Eşitlik 5.3 ve Eşitlik 5.4'te gösterilmiştir.

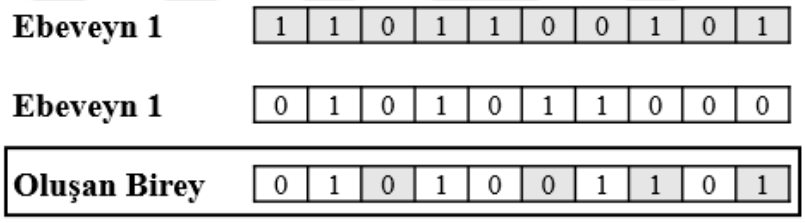
C_t çaprazlama toplamı, C_r çaprazlama oranı, $p1_f$ ebeveyn 1'in uygunluk değeri, $p2_f$ ebeveyn 2'nin uygunluk değeridir. İki ebeveynin uygunluk değerleri toplanarak

çaprazlama toplamı elde edilmektedir. Ardından her bir bireyin çaprazlama oranının hesaplanması için yüz ile çarpılıp çaprazlama toplamına bölünmüştür.

$$C_t = p1_f + p2_f \tag{5.3}$$

$$C_r = \frac{(p1_f * 100)}{(C_t)} \tag{5.4}$$

Formülden orana bağlı olarak, 0-1 olmak üzere iki sayı arasında rastgele bir seçim yapılır. 0 ise ebeveyn 1'den gen aktarılır, 1 ise gen ebeveyn 2'den oluşacak yeni bireye aktarılır. Örnek bir çaprazlama Şekil 5.12'de verilmiştir. Ebeveyn1'in çaprazlama oranı %40, ebeveyn 2'nin çaprazlama oranı %60'tır. Üçüncü gen, altıncı gen, sekizinci gen ve onuncu gen ebeveyn1'den gelirken oluşan yeni bireyin diğer genleri ise ebeveyn 2'den gelmektedir. Bu tez çalışmasında orantılı tekdüze çaprazlama yöntemi kullanılmıştır.



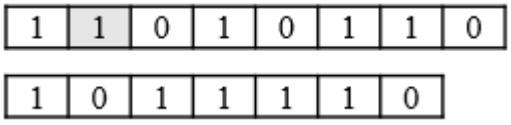
Şekil 5.12. Ebeveyn çaprazlama örneği

5.6 Mutasyon

Genetik algoritmada mutasyon, kromozomu popülasyonun bir neslinden diğerine geçirirken genetik çeşitliliği destekleme için kullanılan genetik operatördür. Çözüm, önceki çözümden tamamen farklı olabilir. Mutasyon, kromozomun kalitesini iyileştirebileceği gibi kötüleştirebilir de. Geni mutasyona uğratmanın birçok yöntemi vardır (Bäck vd, 1993).

5.6.1 Gen Çıkarma Yöntemi

Gen çıkarma yönteminde, kromozom üzerinde rastgele seçilen bir gen o bireyden atılmaktadır. Şekil 5.13'te verildiği üzere bireyin ikinci geni kromozom üzerinden atılarak sekiz geni bulunan kromozomun mutasyon sonucunda artık yedi geni bulunmaktadır. Bu mutasyon yöntemi küme örtüsü probleminin her bir elemanın örtülmesi koşulunu sağlamadığı için kullanılamamaktadır.



Şekil 5.13. Gen çıkarma mutasyon yöntemi örneği

5.6.2 Yer Değiştirme Yöntemi

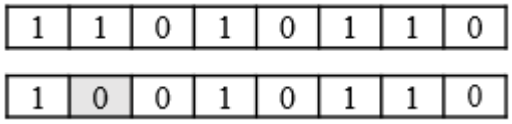
Yer değiştirme yöntemi, kromozom içindeki rastgele seçilen iki genin konumlarının değiştirilmesi yöntemidir. Örneğin Şekil 5.14'de de verildiği gibi kromozomdaki ikinci ve beşinci genlerin değerleri birbirleri ile yer değiştirilmiştir. Bu mutasyon yöntemi, permütasyon kodlamalı kromozomlar için ögelerin sırasını değiştirmektedir.

Şekil 5.14. Yer değiştirme mutasyon yöntemi örneği

5.6.3 Tersine Çevirme Yöntemi

Tersine çevirme mutasyon yöntemi ikili kodlanmış kromozomlarda kullanılmaktadır. Kromozom içinde rastgele seçilen bir genin değerinin tersine çevrilmesini içermektedir. Örneğin, seçilen genin değeri 0 ise, 1 olarak değiştirilir veya genin değeri 1 ise 0 olarak gen mutasyona uğratılmaktadır. Küme örtüsü probleminin koşullarına uygun olan mutasyon yöntemleri yer değiştirme ve tersine çevirme

yöntemleridir. Bu tez çalışmasında, tersine çevirme yöntemi kullanılmıştır. Mevcut genlerden birinin değeri mevcut değerinin tersine çevrilmiştir. Popülasyonun %1 oranındaki bireyine mutasyon uygulanmaktadır. Eğer mutasyon sonucu oluşan birey sağlıklı bir birey değilse yeni bir birey oluşturulmaktadır. Şekil 5.15'te tersine çevirme mutasyon yönteminin örneği verilmiştir. Kromozomdaki ikinci genin değeri 1 iken tersine çevrilerek genin değeri 0 olmuştur.



Şekil 5.15. Tersine çevirme mutasyon yöntemi örneği

BÖLÜM ALTI
HESAPLAMA DENEMELERİ

Test için farklı boyutlarda 81 problem seti üretilmiştir. Tüm üretilen problemlerde eleman değerleri 10, 25, 50, 100 ve 200 olarak ele alınmıştır. Alt kümeler ise 3 farklı grupta üretilmiştir. *n* alt küme sayısını *m* eleman sayısını vermektedir. Buna göre ilk grup için alt küme sayısı Eşitlik 6.1'de verilmiştir. *i* değerleri 1'den başlamaktadır. Eleman değeri bir sonraki değere geçince *i* değeri 1 artmaktadır. Beş farklı boyutta eleman değeri olduğu için *i* en fazla 5 olmaktadır. İkinci grup için alt küme sayısı Eşitlik 6.2'de verilmiştir. Üçüncü grup içinse alt küme sayısı Eşitlik 6.3'de verilmiştir. Üretilen problemlerdeki eleman-alt küme ilişkisi bu şekilde oluşturulmuştur. Problem isimlendirmesine göre 1-2-3 numaralı problemlerde alt kümelerdeki eleman sayısı rastgele olarak oluşturulmuştur. Ancak 4-5-6 numaralı problemlerde her kümede eşit sayıda eleman bulunmaktadır.

$$n = m * 2^{i} \quad (6.1)$$

$$n = m * 10 \quad (6.2)$$

$$n = m * 20 \quad (6.3)$$

Bu çalışmada önerilen genetik algoritmada, genotip ikili kodlama biçimindedir. Başlangıç popülasyonunun oluşturulmasında öncelikle çözüm vektörüne iyi katkı sağlayacak alt kümelere seçim formülü uygulanmıştır. Seçim formülündeki taban değeri 10 olarak alınmıştır. Farklı problem tiplerinde denenmiş ortalama en iyi sonuçların bu taban değerinde verildiği gözlenmiştir. Sonrasında bu kümelerin frekanslarına göre turlama tekniği oluşturulmuştur. %50 oranında kalan bireyler rastgele olacak şekilde seçilmiştir. Seçim yöntemi olarak rulet tekerleği yöntemi kullanılmıştır. KÖP, minimizasyon problemi olduğu için bireylerin uygunlukları hesaplanırken normalizasyon yapılmıştır. %50 oranındaki bireylere elitizm uygulanarak en iyi %50 birey bir sonraki nesle direkt olarak aktarılmıştır. Çaprazlamada orantılı tekdüze çaprazlama yöntemi uygulanmıştır. Ebeveynler çaprazlama oranlarına göre çocuk bireylere genlerini aktarmışlardır. Popülasyonun %1 oranındaki bireylerin bir genine tersine çevirme mutasyon yöntemi uygulanmıştır.

Genetik algoritma C programlama dilinde kodlanmıştır. Genetik algoritmada üretilen tüm problemlerin çözüm sonuçları bilinen optimum sonuçlarla karşılaştırılmıştır. Bilinen optimum değerler GAMS yazılımı ile çözdürülerek elde edilmiştir. Metotların çözüm süreleri dakika, saniye ve milisaniye olarak belirtilmiştir. Genetik algoritma büyük boyutlu problemlerde optimum değerden sapmalar göstermiştir. Ancak bu çözüme kısa sürede ulaşıldığı için sapmalar göz ardı edilmiştir.

Üretilen problem setlerinde ilk değer problemdeki eleman sayısını (m) ikinci değer ise problemin alt küme sayısını (n) temsil etmektedir. Tablo 6.1'de üretilen problemlere ait genetik algoritma ile elde edilen çözüm sonuçları ve süreleri gösterilmektedir. Genetik algoritmaya ait çözüm ve süreler ortalama değerleri belirtmektedir.

C dilinde programlanmış genetik algoritma, 16 GB RAM ve M1 işlemcili bir macOS işletim sistemine sahip bilgisayarla çözülmüştür. Bilinen değerler sütunu GAMS yazılımı ile çözdürülen problemlerin sonuçlarından elde edilmektedir.

Tablo 6.1. Eşitlik 6.1 ile üretilen problemlerin çözümü

No	Problem Size (m x n)	GA Çözüm	GA Süre	Bilinen Değerler	Mutlak Hata
1	10-20(1)	3	0.000422	3	--
2	10-20(2)	3	0.000415	3	--
3	10-20(3)	3	0.000428	3	--
4	10-20(4)	4	0.000431	4	--
5	10-20(5)	4	0.000420	4	--
6	10-20(6)	4	0.000572	4	--
7	25-100(1)	3	0.051136	3	--
8	25-100(2)	3	0.049685	3	--
9	25-100(3)	3	0.048736	3	--
10	25-100(4)	5	0.053484	5	--
11	25-100(5)	**5**	0.053663	4	1
12	25-100(6)	3	0.051055	3	--
13	50-400(1)	4	1.972.539	3	1
14	50-400(2)	4	1.963.098	4	--
15	50-400(3)	4	1.992.941	4	--
16	50-400(4)	**6**	2.193.527	5	1

Tablo 6.1. devamı

No	Problem Size (m x n)	GA Çözüm	GA Süre	Bilinen Değerler	Mutlak Hata
17	50-400(5)	**5**	2.230.817	4	1
18	50-400(6)	4	1.992.451	4	--
19	100-1600(1)	4	68.669.144	4	--
20	100-1600(2)	4	71.151.077	4	--
21	100-1600(3)	5	71.013.519	5	--
22	100-1600(4)	**19**	88.379662	14	5
23	100-1600(5)	**7**	63.648.354	6	1
24	200-6400(1)	**27**	2.158.122.803	17	10
25	200-6400(2)	**18**	1.953.111.816	12	6
26	200-6400(3)	**14**	1.640.886.963	10	4
27	200-6400(4)	**27**	2.954.638.672	16	11

Tablo 6.1'de sapan değerlerin mutlak hatası en büyük problem boyutu 6400'de 11 olarak saptanmıştır. Bu problem grubunda toplamda 10 problemde sapma tespit edilmiştir.

Tablo 6.2. Eşitlik 6.2 ile üretilen problemlerin çözümleri

No	Problem Size (m x n)	GA Çözüm	GA Süre	Bilinen Değer	Mutlak Hata
1	10-100(1)	2	0.043941	2	--
2	10-100(2)	2	0.042904	2	--
3	10-100(3)	2	0.044353	2	--
4	10-100(4)	4	0.051218	4	--
5	10-100(5)	4	0.050958	4	--
6	10-100(6)	3	0.044762	3	--
7	25-250(1)	3	0.254729	3	--
8	25-250(2)	3	0.260051	3	--
9	25-250(3)	3	0.257005	3	--
10	25-250(4)	4	0.268890	4	--
11	25-250(5)	5	0.264687	5	--
12	25-250(6)	3	0.245108	3	--
13	50-500(1)	3	11.435.682	3	--
14	50-500(2)	3	3.599.841	3	--
15	50-500(3)	4	3.893.944	4	--
16	50-500(4)	**7**	3.981.837	6	1
17	50-500(5)	7	4.789.147	7	--
18	50-500(6)	4	4.307.945	4	--
19	100-1000(1)	4	18.704.697	4	--
20	100-1000(2)	4	18.204.485	4	--

Tablo 6.2. devamı

No	Problem Size (m x n)	GA Çözüm	GA Süre	Bilinen Değer	Mutlak Hata
21	100-1000(3)	**6**	18.511.345	5	1
22	100-1000(4)	**12**	21.785.172	9	3
23	100-1000(5)	**7**	20.912.020	6	1
24	200-2000(1)	5	136.643.951	5	--
25	200-2000(2)	5	139.043.015	5	--
26	200-2000(3)	**27**	222.893.616	17	10
27	200-2000(4)	**17**	169.058.670	12	5

Tablo 6.2'de sapan değerlerin mutlak hatası en büyük problem boyutu 2000'de 10 olarak saptanmıştır. Bu problem grubunda toplamda 6 problemde sapma tespit edilmiştir. GAMS'in çözüm süreleri ile kıyaslandığında genetik algoritma daha iyi zamanlarda sonuçlara ulaştığı için sapma miktarları tolere edilmiştir.

Tablo 6.3. Eşitlik 6.3 ile üretilen problemlerin çözümleri

No	Problem Size (m x n)	GA Çözüm	GA Süre	Bilinen Değer	Mutlak Hata
1	10-200(1)	2	0.139357	2	--
2	10-200(2)	2	0.138006	2	--
3	10-200(3)	2	0.142122	2	--
4	10-200(4)	4	0.147109	4	--
5	10-200(5)	4	0.149253	4	--
6	10-200(6)	5	0.144315	5	--
7	25-500(1)	3	1.046.285	3	--
8	25-500(2)	3	1.011.659	3	--
9	25-500(3)	3	1.039.058	3	--
10	25-500(4)	4	1.915.324	4	--
11	25-500(5)	4	1.065.212	4	--
12	25-500(6)	4	1.835.237	4	--
13	50-1000(1)	3	5.529.721	3	--
14	50-1000(2)	4	5.302.804	4	--
15	50-1000(3)	5	54.525.024	5	--
16	50-1000(4)	**6**	5.938.272	5	1
17	50-1000(5)	**8**	22.149.052	6	2
18	50-1000(6)	**6**	31.759.827	5	1
19	100-2000(1)	4	75.650.192	4	--
20	100-2000(2)	4	74.739.883	4	--
21	100-2000(3)	**7**	74.948.433	6	1
22	100-2000(4)	**14**	117.839.523	10	4

Tablo 6.3. devamı

No	Problem Size (m x n)	GA Çözüm	GA Süre	Bilinen Değer	Mutlak Hata
23	100-2000(5)	**14**	109.579.887	10	4
24	200-4000(1)	**17**	459.628.510	12	5
25	200-4000(2)	**18**	559.645.264	12	6
26	200-4000(3)	**16**	447.994.598	11	5
27	200-4000(4)	**28**	772.808.838	18	10

Tablo 6.3'de sapan değerlerin mutlak hatası en büyük problem boyutu 4000'de 10 olarak saptanmıştır. Bu problem grubunda toplamda 5 problemde sapma tespit edilmiştir. GAMS'in çözüm süreleri ile kıyaslandığında genetik algoritma daha iyi zamanlarda sonuçlara ulaştığı için sapma miktarları tolere edilmiştir.

Farklı alt küme adedindeki ve farklı karakteristik özellikteki 81 adet problem, macOs işletim sistemli, 16 GB RAM'li, M1 işlemcili bir bilgisayarda genetik algoritma programı ile çözülmüştür. Problemlerin bilinen çözüm değerleri GAMS yazılımı ile hesaplanmıştır. Kullanılan GAMS yazılımının lisansından kaynaklı süre konusunda limit kısıtı yaşanmıştır. 16 dakikanın üzerindeki çözümlerde program mevcut en iyi değeri sonuç olarak göstermektedir. Problemin genetik algoritma ile çözümündeki mutlak hataları da tablolarda yer almaktadır. Karşılaşılan en büyük mutlak hata 200-6400(4) kodlu ve 6400 alt kümeli problemde olup optimum değerden 11 sapma gerçekleşmiştir. Ele alınan küme örtüsü probleminin ait olduğu karmaşıklık sınıfının karakteri gereği optimumdan sapmalar büyük boyutlu problemlerde daha sık görülmüş ve boyut arttıkça mutlak hataların da arttığı gözlenmiştir.

BÖLÜM YEDİ
SONUÇLAR VE DEĞERLENDİRME

Bu tez çalışmasında önerilen genetik algoritmanın yanında asıl katkı sağlayan, çözüm kümesine iyi katkı sağlayacak olan kümelerin seçim formülü ile sıklıklarının belirlenmesidir. Oluşan bu sıklıklarına göre, indis öncelikleri yüzünden dezavantaja düşen ve permütasyon uzayının sadece belirli bir kısmı ile çalışma avantajı sağlayan turlama yöntemi olmuştur.

Gelecek çalışmalarda, farklı operatörler ve farklı parametre değerleri ile problemler çözdürülerek bu tez çalışmasında kullanılan parametre ve yöntemler analiz edilerek kıyaslanacaktır. Mevcut bulunan en iyi değerlerden sapan problemler için sapma miktarlarının düşürülmesi amaçlanmaktadır.

Bu tez çalışmasının başlangıç popülasyonunun oluşturulduğu kısımda kullanılan seçim formülü ve frekanslarına göre sıralama yapmaya yardımcı olan turlama tekniği yöntemlerine ait kaba kodlar Algoritma 1'de ve Algoritma 2'de verilmiştir. Programın sonraki kısmında Bölüm Beş'te açıklanan operatörlerle genetik algoritma uygulanmıştır.

Algoritma 1: Seçim Formülü

```
Sabit m eleman, n alt küme sayı değerleri, iterasyon sayısı, taban değeri,dmax
OKU (Problem dosyası)
mf0 = mf[0]*1.0/m
for i =1 to n
        nr[i] = pow(taban, mf0-mf[fa[i][j])
Quicksort(nr,ni)
```

Algoritma 2: Turlama Yöntemi

```
Sabit dmax = n - 1
Tamsayı d, t, k
Karakter tc
Dizi dac[n + 1]

tekrar:
   tc = 1
   for d = 1 to dmax:
      for j = 0 to n:
         dac[j] = 0
         da[j] = 0
      t = 1 - d
      do while(da[0] < n):
         if tc == 1 ise
            t'yi d kadar artır
         if t > n ise
            t'yi n'e mod al
         if dac[t] == 0 ise:
            dac[t] = 1
            da[0] artır
            da[da[0]] = t
            tc = 1
         else:
   do while(dac[t] == 1)
               t++
               if t > n ise
                  t = 1
            tc = 0
      for i = 1 to n:
         for j = 1 to n:
```

```
            nv[j] = ni[da[j]]
            da[j]++
            if da[j] > n ise
                da[j] = 1
        end for
        for j = 0 to m:
            fm[j] = 0
        k = 0
        for j = 0 to n:
    fsv[j] = 0
while(fm[0] < m):
    k++
    kc = 0
    for j = 1 to fa[nv[k]][0]:
        if fm[fa[nv[k]][j]] == 0 ise:
            fm[fa[nv[k]][j]] = 1
            fm[0]++
            kc = 1
    if kc == 1 ise:
        fsv[0]++
        fsv[fsv[0]] = nv[k]
popi++
pop[popi][0] = fsv[0]
for j = 1 to fsv[0]:
    pop[popi][fsv[j]] = 1
    nf[fsv[j]]'yi n - fsv[0] ile artır
if fsv[0] < minv[0] ise:
    minv[0] = fsv[0]
    for j = 1 to fsv[0]:
        minv[j] = fsv[j]
```

```
Quicksort(nf,nfi)
if tek == 0 ise:
    for i = 1 to n:
        ni[i] = nfi[i]
    tek = 1
    goto tekrar
```

KAYNAKLAR

Aickelin, U. (2002). An indirect genetic algorithm for set covering problems. Journal of the Operational Research Society, 53, 1118-1126.

Al-Sultan, K. S., Hussain, M. F., & Nizami, J. S. (1996). A genetic algorithm for the set covering problem. Journal of the Operational Research Society, 47(5), 702-709.

Balas, E., &Padberg, M. W. (1972). On the Set-Covering Problem. Operations Research, 20(6), 1152–1161. http://www.jstor.org/stable/169305

Bäck, T., Hoffmeister, F., Belew, R. K., & Booker, L. B. (1993). Proceedings of the 5th International Conference on Genetic Algorithms.

Beasley, J. E. (1987). An algorithm for set covering problem. European Journal of Operational Research, 31(1), 85-93.

Beasley, J. E. (June 1990). OR-Library http://people.brunel.ac.uk/~mastjjb/jeb/info.html

Beasley, J. E. (1990). A lagrangian heuristic for set-covering problems. Naval Research Logistics (NRL), 37(1), 151-164.

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

Caprara, A., Fischetti, M., & Toth, P. (1999). A heuristic method for the set covering problem. Operations research, 47(5), 730-743.

Chvatal, V. (1979). A greedy heuristic for the set-covering problem. Mathematics of operations research, 4(3), 233-235.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms. MIT press.

Crawford, B., Soto, R., Suárez, M. O., Paredes, F., & Johnson, F. (2014, June). Binary firefly algorithm for the set covering problem. In 2014 9th Iberian Conference on Information Systems and Technologies (CISTI) (pp. 1-5). IEEE.

Crawford, B., Soto, R., Cuesta, R., & Paredes, F. (2014). Application of the artificial bee colony algorithm for solving the set covering problem. The Scientific World Journal, 2014.

Crawford, B., Soto, R., Monfroy, E., Astorga, G., García, J., & Cortes, E. (2018). A meta-optimization approach to solve the set covering problem. Ingeniería, 23(3), 274-288.

Crawford, B., Soto, R., Palma, W., Aballay, F., Lemus-Romani, J., Misra, S., ... & Rubio, J. M. (2020). A teaching-learning-based optimization algorithm for the weighted set-covering problem. Tehničkivjesnik, 27(5), 1678-1684.

Davis, L. (1994). Handbook of Genetic Algorithms, Van Nostrand Reinhold.

Demirel, L. (1999). Genetic algorithms in engineering optimization (Doctoral dissertation, Institute of Science and Technology).

Duan, C. (2007). A Heuristic Algorithm for Set Covering Problem. *Computer Science*.

Eremeev, A. V. (1999). A genetic algorithm with a non-binary representation for the set covering problem. In Operations Research Proceedings 1998: Selected Papers of the International Conference on Operations Research Zurich, August 31–September 3, 1998 (pp. 175-181). Springer Berlin Heidelberg.

Gencer, M. (2017). Genetik algoritma ile maksimum bağımsız küme probleminin çözümü [Yüksek Lisans Tezi]. Dokuz Eylül Üniversitesi

Grossman, T., & Wool, A. (1997). Computational experience with approximation algorithms for the set covering problem. European journal of operational research, 101(1), 81-92.

Gao, C., Yao, X., Weise, T., & Li, J. (2015). An efficient local search heuristic with row weighting for the unicost set covering problem. European Journal of Operational Research, 246(3), 750-761.

Hadji, R., Rahoual, M., Talbi, E. G., & Bachelet, V. (2000). Ant colonies for the set covering problem. In Abstract proceedings of ANTS (pp. 63-66).

Hey, A. M. (1981). Algorithms for the set covering problem. University of London (United Kingdom).

Huang, W. C., Kao, C. Y., & Horng, J. T. (1994, June). A genetic algorithm approach for set covering problems. In Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence (pp. 569-574). IEEE.

Johnson, D. S. (1982). The NP-completeness column: An ongoing gulde. Journal of Algorithms, 3(4), 381-395.

Marchiori, E., & Steenbeek, A. G. (1998). An iterated heuristic algorithm for the set covering problem.

McConnell, J. J. (2001). Analysis of Algorithms: An Active Learning Approach, by Jones and Bartlett Publishers. Inc., Massachussetts, USA.

Papadimitriou, C. H., & Steiglitz, K. (1998). Combinatorial optimization: algorithms and complexity. Courier Corporation.

Razip, H., & Zakaria, N. (2021). Genetic Algorithm with Approximation Algorithm Based Initial Population for the Set Covering Problem. In Proceedings of International Conference on Communication and Computational Technologies: ICCCT 2021 (pp. 59-78). Springer Singapore.

Ren, Z. G., Feng, Z. R., Ke, L. J., & Zhang, Z. J. (2010). New ideas for applying ant colony optimization to the set covering problem. Computers & Industrial Engineering, 58(4), 774-784.

Thomas H, C., Charles, E., Ronald L, R., & Clifford, S. (2009). Introduction to Algorithms Third Edition.

Soto, R., Crawford, B., Olivares, R., Barraza, J., Figueroa, I., Johnson, F., ... & Olguín, E. (2017). Solving the non-unicost set covering problem by using cuckoo search and black hole optimization. Natural Computing, 16, 213-229.

Wolsey, L. A. (1982). An analysis of the greedy algorithm for the submodular set covering problem. Combinatorica, 2(4), 385-393.

Yelbay, B., Birbil, Ş. İ., & Bülbül, K. (2014). The set covering problem revisited: an empirical study of the value of dual information. Journal of Industrial and Management Optimization.

Zaozerskaya, L. A. (1998). On L-class enumeration algorithm for set covering problem. In Proc. of (pp. 139-142).