KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İSTATİSTİK VE BİLGİSAYAR BİLİMLERİ ANABİLİM DALI

DÖRT İŞLEM KOMBİNASYON PROBLEMİNİN TİP-2 AĞAÇ YAPISI İLE ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

Elçin AĞAYEV

MAYIS 2017
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İSTATİSTİK VE BİLGİSAYAR BİLİMLERİ ANABİLİM DALI

DÖRT İŞLEM KOMBİNASYON PROBLEMİNİN TİP-2 AĞAÇ YAPISI İLE ÇÖZÜMÜ

Elçin AĞAYEV

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"YÜKSEK LİSANS (İSTATİSTİK)"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 08.05.2017
Tezin Savunma Tarihi : 29.05.2017

Tez Danışmanı : Yrd. Doç. Dr. Orhan KESEMEN

Trabzon 2017

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü
İstatistik ve Bilgisayar Bilimleri Anabilim Dalında
Elçin AĞAYEV tarafından hazırlanan

DÖRT İŞLEM KOMBİNASYON PROBLEMİNİN TİP-2 AĞAÇ YAPISI İLE ÇÖZÜMÜ

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 09 / 05 / 2017 gün ve 1701 sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda
YÜKSEK LİSANS TEZİ
olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Yrd. Doç. Dr. Halil İbrahim ŞAHİN …...……

Üye : Yrd. Doç. Dr. Orhan KESEMEN …...……

Üye : Yrd. Doç. Dr. Fatma Zehra DOĞRU ……...……

Prof. Dr. Sadettin KORKMAZ
Enstitü Müdürü

ÖNSÖZ

"Dört İşlem Kombinasyon Probleminin Tip-2 Ağaç Yapısı İle Çözümü" isimli bu tez Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstatistik ve Bilgisayar Bilimleri Anabilim Dalı, Yüksek Lisans Programı'nda hazırlanmıştır.

Tez çalışma süresinde değerli yardım ve katkılarıyla beni yönlendiren benden yardımlarını, desteğini, sabrını ve bilgisini esirgemeyen değerli danışman hocam Yrd. Doç. Dr. Orhan KESEMEN'e, eğitimimde emeği geçen tüm hocalarıma, tez sürecinde hiçbir yardımdan kaçınmayan hocalarım Eda ÖZKUL'a, Özge TEZEL'e ve Buğra Kaan TİRYAKİ'ye teşekkürü bir borç bilirim.

Son olarak, tüm hayatım boyunca maddi ve manevi her zaman beni destekleyen, her adımımda arkamda duran aileme sonsuz teşekkürlerimi sunarım.

Bu tezin, bundan sonraki çalışmalara katkı sağlamasını temenni ederim.

Elçin AĞAYEV

Trabzon 2017

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum “Dört İşlem Kombinasyon Probleminin Tip-2 Ağaç Yapısı İle Çözümü” başlıklı bu çalışmayı baştan sona kadar danışmanım Yrd. Doç. Dr. Orhan KESEMEN‘in sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 29/05/2017

Elçin AĞAYEV

İÇİNDEKİLER

Sayfa No

Yüksek Lisans

ÖZET

DÖRT İŞLEM KOMBİNASYON PROBLEMİNİN TİP-2 AĞAÇ YAPISI İLE ÇÖZÜMÜ

Elçin AĞAYEV

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
İstatistik ve Bilgisayar Bilimleri Anabilim Dalı
Danışman: Yrd. Doç. Dr. Orhan KESEMEN
2017, 51 Sayfa

Kombinasyon problemleri olasılık teorisinde önemli bir role sahiptir. Bir olayın olasılığı, istenen koşulları sağlayan alt kümeler ile tüm mümkün alt kümelerin oranı olarak hesaplanmaktadır. İstenen koşulları her zaman matematiksel olarak hesaplamak zordur. Günümüzde bu zorluğu aşmak için gelişen bilgisayar teknolojisi kullanılmaktadır.

Dört işlem kombinasyon problemi, daha çok televizyon programlarında oynanan bir yarışma programı olarak karşımıza çıkmaktadır. Bu problemde belli sayıda verilen sayılar arasında dört işlem uygulanarak hedeflenen sayıya ulaşılmaya çalışılır. Bu işlemi gerçekleştirirken verilen her sayının bir kez kullanımına izin verilirken operatörler istenildiği kadar kullanılabilir. Bu problemde, bulunacak olası tüm dört işlem kombinasyonları iki sayı ve bir operatörden oluşan işlem öbeklerini içermektedir. Bu öbekler üç farklı kümenin kesişimi temel alınarak oluşturulmaktadır. Dolayısıyla bu kesişimi tam anlamıyla modelleyebilmek için yeni bir yaklaşım olan "Tip-2 Ağaç" yapısı geliştirilmiştir.

Bu çalışmada, verilen sayı miktarı artıkça hedefe ulaşma oranının belirlenmesinin yanı sıra rastgele oluşturulan her hedefe kaç farklı şekilde ulaşıldığı da hesaplanmıştır. Tüm hesaplamalarda Visual Studio 2012 C# ortamında yazılan programlar kullanılmıştır.

Anahtar Kelimeler: Kombinasyon, Dört işlem, Tip-2 Ağaç yapısı, İşlem öbeği.

Master Thesis

SUMMARY

SOLVING FOUR OPERATIONS COMBINATION PROBLEM USING TYPE-2 TREE STRUCTURE

Elçin AĞAYEV

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Statistical and Computer Sciences Graduate Program
Supervisor: Assist. Prof. Dr. Orhan KESEMEN
2017, 51 Pages

Combination problems play an important role in probability theory. The probability of an event is calculated as the ratio of all possible sub-clusters and sub-clusters that provide the desired conditions. It is always difficult to calculate the desired conditions mathematically. Today, developing computer technology is used to overcome this difficulty.

The problem of the four operation combinations is often confronted as a competition program played on television programs. It is tried to reach the target number by applying four operations between given numbers in this problem. When performing this operation, operators can be used as many times as required, provided that each number issued is allowed to be used once. In this problem, all possible combinations of four operations include two numbers and a set of operations consisting of one operator. These clusters are based on the intersection of three different clusters. Therefore, a new approach "Type-2 Tree" has been improved for fully model this intersection.

In this study, it has been calculated that how many different way the random number is reached for each randomly generated target, as well as determining the rate of reaching the target as the count of given numbers increases. All calculations use programs that written in the Visual Studio 2012 C # environment.

Key Words: Combination, Four operations, Type-2 Tree structure, Operation block.

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

P(n)	: n elemanlı bir kümenin permütasyonu
P(n, k)	: n elemanlı bir kümenin k elemanı alt küme permütasyonları
C(n, k)	: n elemanlı bir kümenin k elamanlı alt küme kombinasyonları
D(n)	: n elemanlı bir sayılar kümesinden dört işlemle üretilebilecek tüm olası sayıların sayısı
D(n, k)	: n elemanlı bir sayılar kümesinden k tane sayı kullanılarak elde edilen değer sayısı
"+"	: Toplama operatörü
"-"	: Çıkarma operatörü
"*"	: Çarpma operatörü
"/"	: Bölme operatörü
{ }	: Küme ayraçları
( )	: İşlem öncelik ayraçları
Σ	: Dizi toplam operatörü
$\prod$	: Dizi çarpım operatörü

1. GENEL BİLGİLER

1.1. Giriş

Günümüzde insanoğlunun yoğun iş temposuna ayak uydurması beraberinde psikolojik birçok sorunun yanında getirmektedir. Bu stresli yaşama ayak uydurmaya çalışan insanoğlu iş yaşamı dışında da farklı etkinliklere zaman ayırarak bu stresini azaltma yoluna gitmektedir. İnsanın boş zamanlarını değerlendirmek için seçtiği birincil alan oyunlar olmaktadır. İnsanların çocukluklarından beridir oyunlara karşı bir ilgisi olmuş ve bunu ileriki yaşamında da sürdürmeye devam etmektedir. Özellikle bilgisayarların ve cep telefonların gelişmesiyle birlikte oyunlar bir sektör haline gelmiştir. Bu kadar yoğun bir sektörde oyunlar çeşitlenmekte ve karmaşıklaşmaktadır. Oynanan oyunları genel itibariyle iki sınıfa ayırabiliriz. Bu oyunlar stratejik (akıl) oyunları olduğu gibi beceri (grafiksel aksiyon) oyunları olarak karşımıza çıkmaktadır. Beceri oyunları genelde çocukların ilgisini çekerken strateji oyunları büyüklerin ilgini çekmektedir. Strateji oyunlarının çoğunun çözümü kombinasyonlarla çözülebilmektedir.

Bu çalışmada televizyonlarda (TRT) yıllardır oynanan bir kelime-bir işlem oyunun bir işlem kısmını bilgisayar ortamında çözümü yapılmıştır. Bu oyun dört işlem kombinasyon problemi olarak tanımlanabilir. Bu problemin çözümü için tip-2 graf tekniği geliştirilmiştir.

Dört işlem kombinasyon problemi, İngilizce Countdown olarak bilinen bir işlem oyunu verilen sayılara dört işlem uygulayarak hedef sayıyı elde etmeyi amaçlayan bir zeka oyunudur (Colton, 2014). Bu oyun, ilk olarak, 1972 yılında Fransız televizyonlarında "des chiffres et des lettres" adıyla yayınlanmaya başlamıştır. Daha sonra, 1982 yılında İngiliz televizyonunda "countdown" adıyla yayınlanmıştır. İngiliz ve Fransız sürümleri bir birlerinden azda olsa farklıdırlar. İngiliz sürümünde adaylara sonuç 100-999 arasında verilerek 30 saniyede çözümü istendiği halde, Fransız sürümünde sonuç 101-999 arasında ve 45 saniye verilerek çözülmesi isteniyor. Her iki versiyonda sonuç rakam rasgele verilir. Ancak giriş sayılarında bazı farklılıklar vardır. Fransız sürümünde adaylara giriş rakamları rastgele verildiği halde İngiliz sürümünde adaylar giriş rakamlarını karışık kümeden kendileri seçiyorlar (Hutton, 2002; Mogos ve Florea, 2008). Türkiye'de ise "bir kelime bir işlem" olarak bilinen yarışma programıdır. Oyun 6 sayıyla oynanıyor. Bu sayılardan beş

tanesi 1 ile 9 arasında rastgele seçilirken, bir tanesi {25, 50, 75, 100} kümesinden rastgele seçilmektedir. Yarışmacılar bu 6 sayıyı kullanarak (+ - * /) operatörleri yardımıyla rasgele seçilen hedef bir (101 le 999 arasında) sayıya ulaşmaya çalışmaktadırlar. Oyunun kuralları, seçilen 6 sayı yalnız bir kere kullanılabilir. Ancak her iki sayı arasında yapılan işlem sonucu çıkan her yeni sayı diğer işlemde kullanılabilir. Örneğin, 6 sayı {3,6,1,4,5,25} olarak seçilsin ve hedef sayı 403 olsun. Bu durumda yapılan işlemler adım adım Tablo 1'de verilmiştir.

Tablo 1. Örnek dört işlem çözümü

	İşlemler	**Kalanlar**
Adım 1	6 * 5 = 30	{1,3,4,25,30}
Adım 2	30 + 1 = 31	{3,4,25,31}
Adım 3	3 * 4 = 12	{25,31,12}
Adım 4	25 - 12 = 13	{31,13}
Adım 5	13 * 31 = 403	{403}

Genelde sonucu elde etmek için birçok yol olmasına rağmen en kolay yol tercih edilir. En kolay yol ise en az sayıdan oluşmuş cevaptır.

Bu oyun oldukça yaygın olmasına rağmen ciddi bir bilimsel bir çalışma yapılmamıştır (Alliot, 2015). Bu konudaki ilk ciddi çalışma Defays (1990) tarafından yapılmıştır. Bu makalede oyunun çözümü için yapay zeka arama yöntemlerini kullandı. Ama Defayın bu yöntemi oyunu kesin çözememektedir. Hutton (2002) tarafından problemin çözümü, Haskel fonksiyonel programlama dili kullanılarak çözülmüştür. Birçok web sitesinde oyunun çözümü için farklı yaklaşımlar ve yazılımlar sergilense de, istenen sonucu vermemektedirler (Alliot, 2015). Alliot'un (2015) yaptığı çalışma ise doğru yaklaşımlar getirmesine rağmen sadece problemin çözümüne odaklanmıştır. Diğer kombinasyonları aramamaktadır.

1.2. Arama Yöntemleri

Arama algoritmaları ile problem çözme yapay zeka uygulamalarında en temel işlemelerden birisidir (Russell ve Norvig, 2010; Nilsson, 2014; Nabiyev, 2012). Birçok problemin çözümünde arama algoritmaları hızlı ve etkili çözümler sunmaktadır. Özellikle

çözüm yöntemi bilinmeyen problemlerin çözümünde tüm seçeneklerin değerlendirilmesi ile çözüme gidilmektedir. Arama yöntemleri kullandıkları tekniklere göre iki sınıfta incelenmektedir. Bunlar arama yaparken bir ön bilgiye ihtiyaç duyulup duyulmadığına göre ön bilgi kullanan ve kullanmayan diye iki ayrılır. Bunlara kısaca bilgili ve bilgisiz arama teknikleri denilmektedir.

Bilgili ve Bilgisiz arama yöntemleri için birçok algoritma geliştirilmiştir. Bunlardan en yaygın olanları aşağıda verilmiştir (Russell ve Norvig, 2010; Nabiyev, 2012):

Bilgisiz Arama (Körüne arama)

- Enine arama (Breadth-First)
- Derinine arama (Depth-First)
- Sabit maliyetli arama (Uniform cost search)
- Sınırlandırılmış derinine arama (Depth limited search)
- Özyinelemeli derinine arama (Iterative deepening)

Bilgili Arama Stratejileri (Sezgisel arama)

- En iyisini arama(Best-First)
- A*
- Tepe Tırmanma (Hill-Climbing)
- Işın arama (Beam)
- Dijkstra

Bir problem için hangi arama algoritmasının seçileceği o arama yönteminin özelliği ile ilgilidir. Bu özelliklerin en belirgin ölçütleri aşağıda maddeler halinde verilmiştir (Demir, 2016).

- Tamlık-Bütünlük (Completeness): Problemin bir çözümü olduğunda seçilecek olan strateji bu çözümü bulup bulmayacağının araştırılmasıdır.
- Zaman Karmaşıklığı (Time Complexity): Bir çözüm bulmak ne kadar zamana ihtiyaç olduğunun araştırılmasıdır.
- Bellek Karmaşıklığı (Space Complexity): Aramayı yapmak için ne kadarlık bir belleğe (memory) gerekli olacak? Özyinelemeli olan algoritmalarda bir önceki durum saklanacağından dolayı problemin çözümü için gerekli hafıza alanı özyinelemeli olmayan algoritmalara göre daha fazla olacaktır.

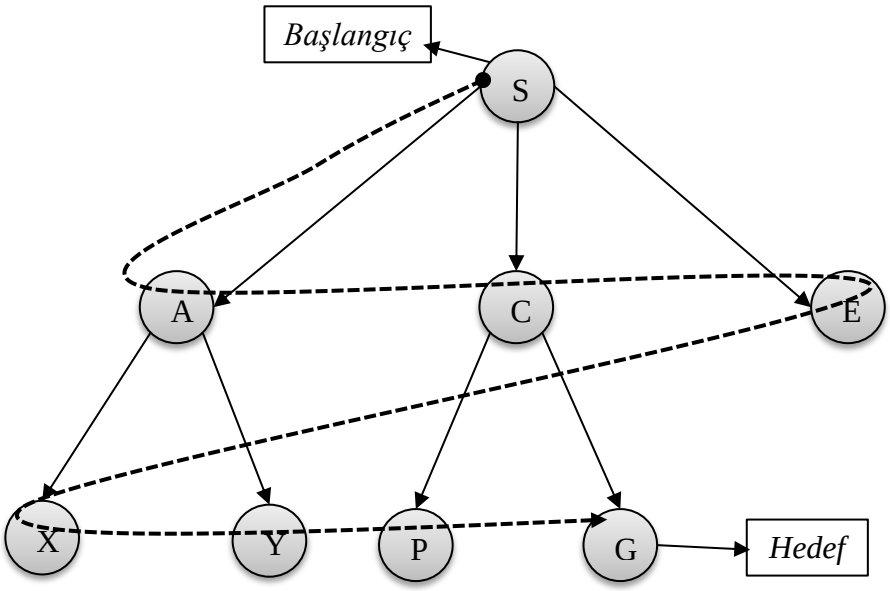
- Eniyileme (Optimality): Farklı çözümler mevcut iken seçilen stratejinin çözümü en iyi olan olup olmadığının araştırılmasıdır.

1.2.1. Kör (Bilgisiz) Arama Yöntemleri

Kör arama yöntemlerinde sonuca varmak için içinde bulunan durumun sonuç ile arasındaki adım sayısı veya mesafesinin bilinmemesidir (Ginsberg, 2012). Yani çözüme ne zaman varılacağı bilinmemektedir. Burada sadece bilinen şey içinde bulunan durumun hedef durum olup olmadığıdır. Kör arama yöntemlerinde çözüme ulaşmak için hiçbir bilgi verilmez. Aramanın her hangi adımında çözüme ne kadar yakın (veya uzak) olması hakkında veya çözümün bulunmasıyla ilgili fikir söylemek mümkün değildir.

1.2.1.1. Enine Öncelikli Arama

Enine aramada ağaç soldan sağa, yukarıdan aşağıya doğru taranıyor (Russell ve Norvig, 2010). Diğer bir deyişle, bir seviyedeki tüm düğümler genişletildikten (tarandıktan) sonra bir sonraki aşağı seviyeye geçilir. Yöntemin şematik gösterimi Şekil 1'de verilmiştir.

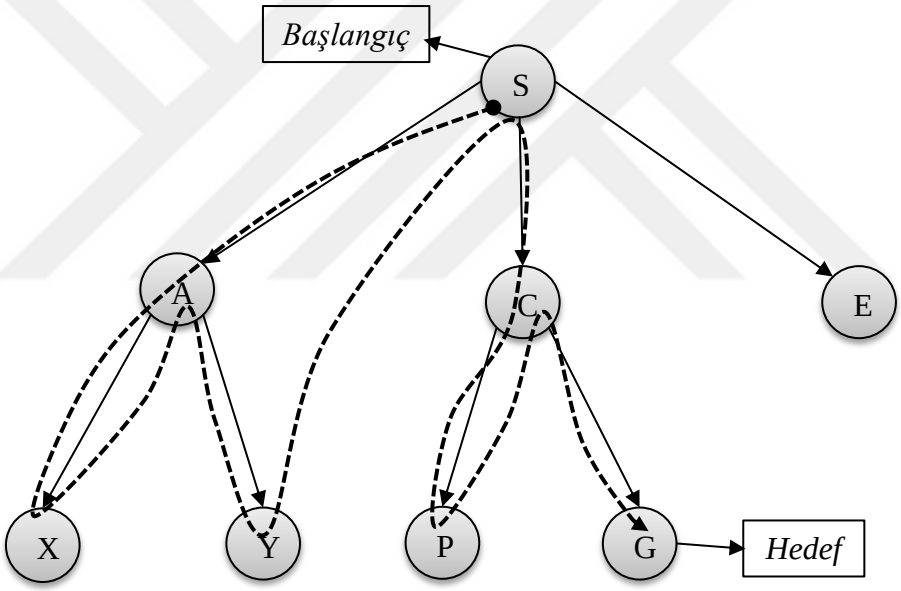


Şekil 1. Enine aramanın ağaç yapısındaki gösterimi ve işlem sırası.

Şekil 1'de görüldüğü gibi enine arama hep enine doğru ilerliyor. Burada başlangıç düğümümüz S, gitmek istediğimiz düğüm ise G'dir. Enine arama yöntemini kullanmak isteyen birisi S'den başlayıp A'ya, sonra sırasıyla C, E, X, Y, P şehirlerine giderek sonunda G ulaşır.

1.2.1.2. Derinine Öncelikli Arama

Derine aramada, arama ağacı yukarıdan aşağıya en sol düğümden başlayarak yaprak düğüme ulaşılana dek geliştiriliyor (Russell ve Norvig, 2010; Ginsberg, 2012). Eğer bir yolda çözüm bulunamazsa, arama sonraki en sol ve gidilmemiş (genişletilmemiş) devam ettirilir. Yöntemin şematik gösterimi Şekil 2'de verilmiştir.



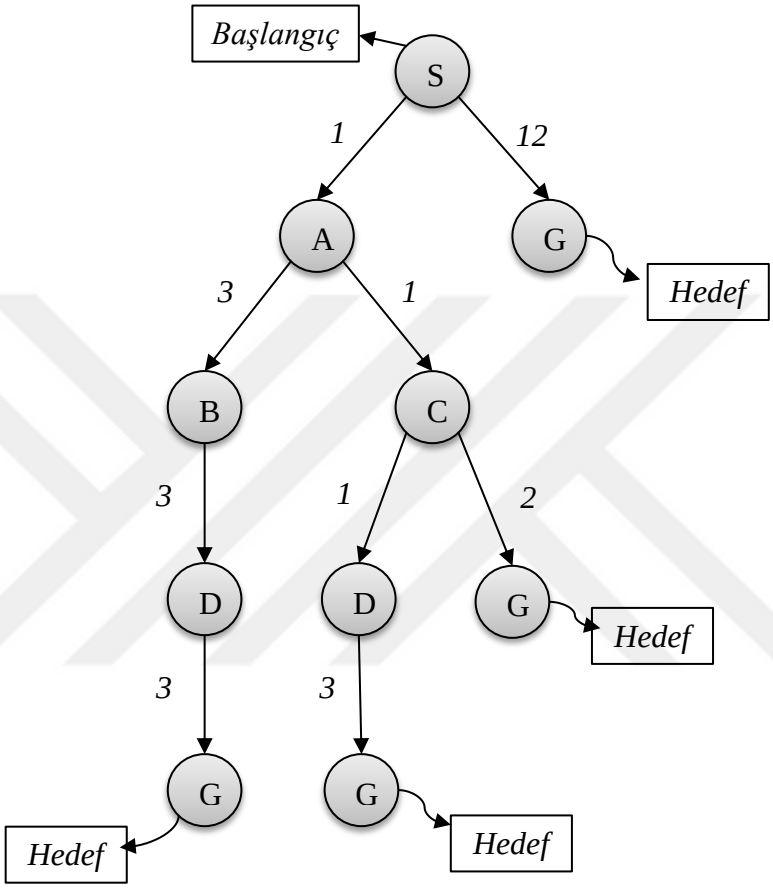
Şekil 2. Derinine aramanın ağaç yapısındaki gösterimi ve işlem sırası.

Derinine aramada amaç düğümleri derinine dolaşarak hedefe ulaşmaktır. Şekil 2'deki örnekten de görüldüğü gibi S'den G'ye gitmek isteyen kişi S'den başlayıp A'ya sonra sırasıyla X, A, Y, S, C, P, C'ye ve son olarak G'ye ulaşır.

1.2.1.3. Sabit Maliyetli Arama

Sabit maliyetli arama, arama algoritmaları arasında en iyi algoritmalardandır ve bünyesinde sezgiler içermemektedir. Bu yöntem herhangi bir genel grafı optimal maliyete

göre çözer. Bu arama yöntemi de derinine aramada olduğu gibi öncelikli kuyruk yapısını kullanır. Derinine aramada maksimum derinlik maksimum öncelikli ise, sabit maliyetli aramda minimum toplam maliyet maksimum önceliklidir. Bu arama yöntemi Şekil 3'teki örnek yardımıyla daha kolay anlatılmaya çalışılmıştır.



Şekil 3. Sabit maliyetli aramanın ağaç yapısındaki gösterimi ve işlem sırası

Şekil 3'te S düğümünden başlanıldığını ve G düğümüne ulaşılmak istenildiği varsayılmaktadır. Algoritma ilk başta S düğümünün komşularının maliyetini çıkarır ve en az maliyetli komşuya gider. Şekil 3'te gösterildiği üzere S düğümünün komşusu iki düğüm vardır. A ve G düğümleri ilk aşamada kontrol edilmiş olunur. Yani ilk adımda A ve G düğümlerine bakılır ve bu düğümlerden kısa olan A düğümüne gidilir. A düğümünden yine komşu düğümler kontrol edilir. Böylelikle B ve C düğümlerini de kontrol etmiş olunur. Bu düğümlerden kısa mesafeli olan C düğümü seçilir. En son toplam maliyeti en az olan S, A, C, G → 4 birim olan yol seçilir. Bunu aşamalar şeklinde aşağıda anlatılmıştır ve öncelikli kuyruğun her elemanının [gittiği yol, toplam maliyet] şeklinde yazılmıştır.

Yineleme1: *{ [S → A , 1] , [S→G , 12] }*

Yineleme2: *{ [S→A→C , 2] , [S→A→B , 4] , [S→G , 12] }*

Yineleme3: *{ [S→A→C→D , 3] , [S→A→B , 4] , [S→A→C→G , 4] , [S→G , 12] }*

Yineleme4: *{ [S→A→B , 4] , [S→A→C→G , 4] , [S→A→C→D→G , 6] , [S→G , 12] }*

Yineleme5: *{[S→A→C→G, 4], [S→A→C→D→G, 6], [S→A→B→D, 7], [S→G, 12]}*

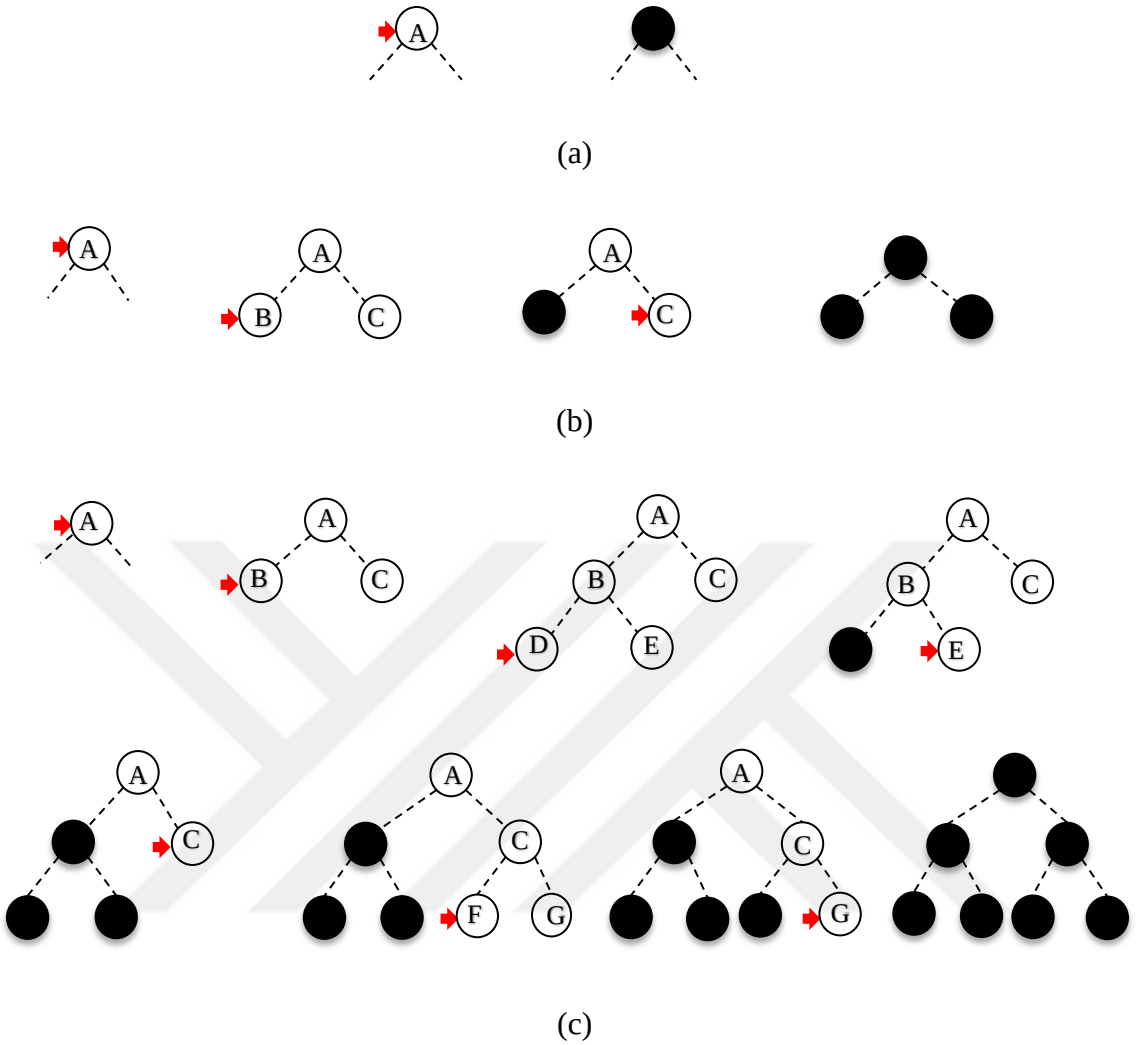
ve son olarak Yineleme6 sonucu S→A→C→G olarak veriyor.

1.2.1.4. Sınırlandırılmış Derinine Arama

Bu arama yönteminde, derinine aramada olası sonsuz arama işlemini önlemek için aramanın belirli bir seviyeye kadar yapılması düşünülmektedir. Örneğin, Yol haritasında hiçbir çözüm 11'den fazla adım gerektirmeyecek. Çünkü burada yalnız 12 yerleşim vardır. Bu nedenle sınır olarak 11 kullanılabilir. Sonsuz döngülerin var olmadığını varsayılarak sorun sonlu derinlik seviyesinde çözülebilir.

1.2.1.5. Yinelemeli Derinine Arama

Satranç turnuvalarında oyunlar kesin zaman sınırı içinde oynanıyor. Satranç programı her hamle için ne kadar zaman kullanması gerektiğine önceden karar vermelidir. Pek çok satranç programları arama işlemini yinelemeli derinine arama ile gerçekleştiriyorlar. Başka bir deyişle, program önce 2. seviyede, sonra 3., sonra 4, … seviyede arama yapıyor. Bu, arama için ayrılan süre bitene kadar devam ediyor. Daha sonra program, bulunan hamleler içinden en iyisini çözüm gibi kabul ediyor (Şekil 4).



Şekil 4. Yinelemeli derinine arama yönteminin şekilsel gösterimi; (a) Yinelemeli derinine arama limit 0 oluğunda; (b) Yinelemeli derinine arama limit 1 olduğunda; (c) Yinelemeli derinine arama limit 2 olduğunda.

Bu arama yönteminde derinlik ayrılan zamana kadar devam ediyor ve zaman dolduğunda en sonucu geri gönderiyor.

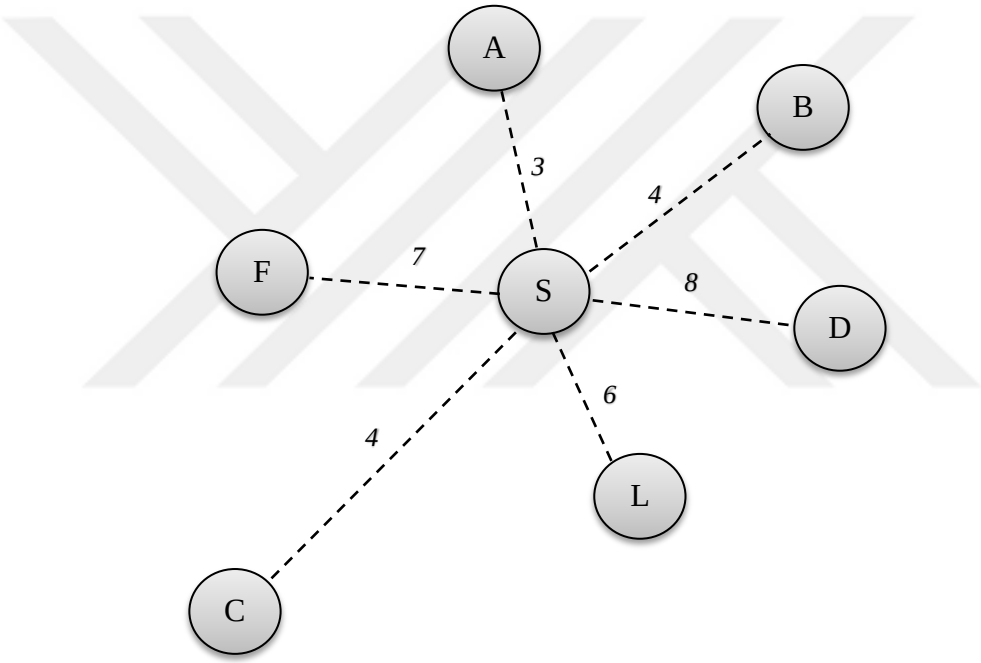
1.2.2. Sezgisel (Bilgili) Arama Yöntemleri

Sezgisel aramada ise bilgisiz aramanın tam aksine sonuca varmak için içinde bulunan durumun sonuç ile arasındaki adım sayısı veya mesafenin önceden bilinmesidir (Jones, 2015).

Kör arama yöntemleri basittir, fakat çoğu zaman pratik değildir. Kör arama yöntemleri bilgisiz yöntemlerdir. Yani, bu yöntemlerle arama, durum uzayı hakkında bilgi

olmadan gerçekleştirilir. Sezgisel arama yöntemleri ise önce en umut verici yolu incelemekle aramanın etkisini yükseltiyor. Sezgisel aramayı kullanmak için bize değerlendirme fonksiyonu (evaluation function) gerekmektedir. Değerlendirme fonksiyonu, hedef/amaç durumuna yakınlığı değerlendirmek için arama ağacında düğümleri inceler. Sezgisellik bir tahmindir, fakat aramayı gerçekleştirmek için yararlı bir yaklaşımdır.

Örneğin, arama uzayında bir karıncanın yiyecek depoları arasındaki gittiği mesafeleri sezgisel değerlendirme için kullanılabilir (Şekil 5). Üstünlük verilen yollar S'ye daha yakın olan yollardır.



Şekil 5. S'den karıncanın gittiği yolların uzunlukları.

1.2.2.1. En İyi Öncelikli Arama

Bazı hallerde amaca doğru herhangi bir yolun bulunması yeterli olsa da, bazı zamanlarda en iyi yolun bulunması gerekebilir. En hızlı, en düşük maliyetle ve en kolay yolla amaca ulaşılması için optimal arama yapılmalıdır. Bu durumda en iyi öncelikli arama ve A* algoritmalarını kullanılabilir.

En iyi öncelikli aramada temel düşünce değerlendirme fonksiyonu $f(n)$ kullanarak her bir düğüm için istenirliği (desirability) tahmin etmektir. Daha sonra en çok istenen düğümü genişletmeli ve düğümleri istenirliklerine göre azalan sırada sıralanmalıdır.

Yöntemin daha iyi anlaşılması için aşağıda bir örnek ele alınmıştır. Bu örnekte Kalbajar'den Bakü'ye gidilecek en kısa yol tahmin edilmesi istenmektedir. Problemin çözümüne geçemeden önce Bakü ile diğer şehirler arasındaki mesafenin belirlenir (Tablo 2).

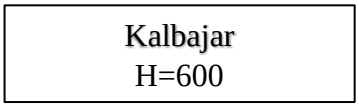
Tablo 2. Bakü ile diğer şehirler arasındaki yol mesafesi.

Kalbajar	600	Yevlakh	410
Shamkir	650	Agjabadi	380
Ganja	580	Shamakhi	240
Agdam	520	Shirvan	270
Khankendi	550	Salyan	300
Fuzuli	500	Sumgayit	120
Barda	450	Baku	0

Kalbajar ve Bakü arasındaki yol güzergahını belirlerken başlangıç olarak Klabajar seçilmiştir. Bu güzergah boyunca her şehirden gidilecek komşu şehirler oklarla gösterilmiştir (Şekil 6).

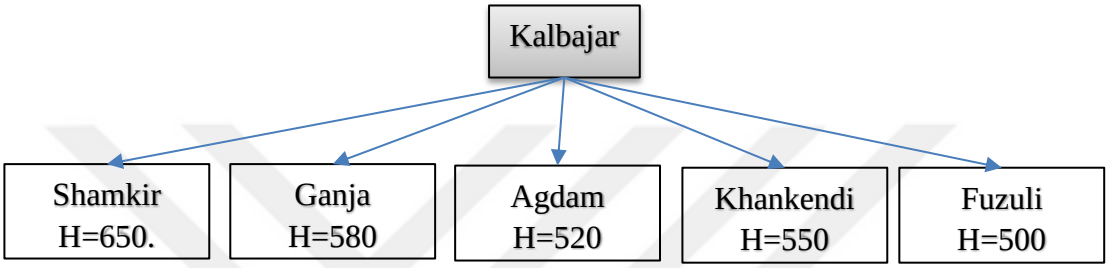
Şekil 6. Azerbaycan'da Kalbajar ile Bakü arasındaki yol güzergahının belirlenmesinde en iyi öncelikli arama yönteminin kullanılması.

Başlangıç durumu,



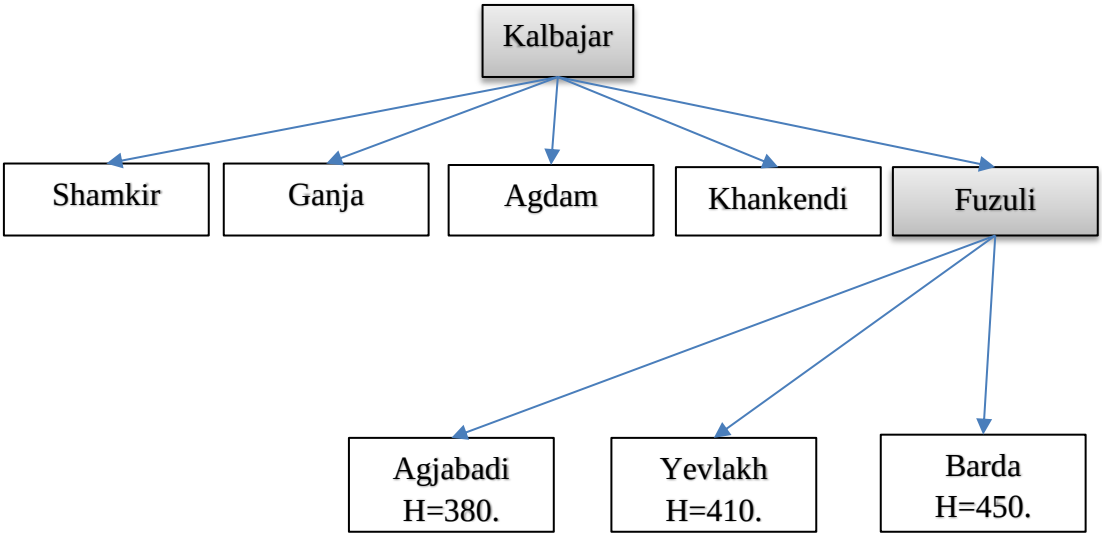
Şekil 7. Başlangıç durumun belirlenmesi

olarak verilmektedir. Kalbajar açıldığında (Şekil 8),



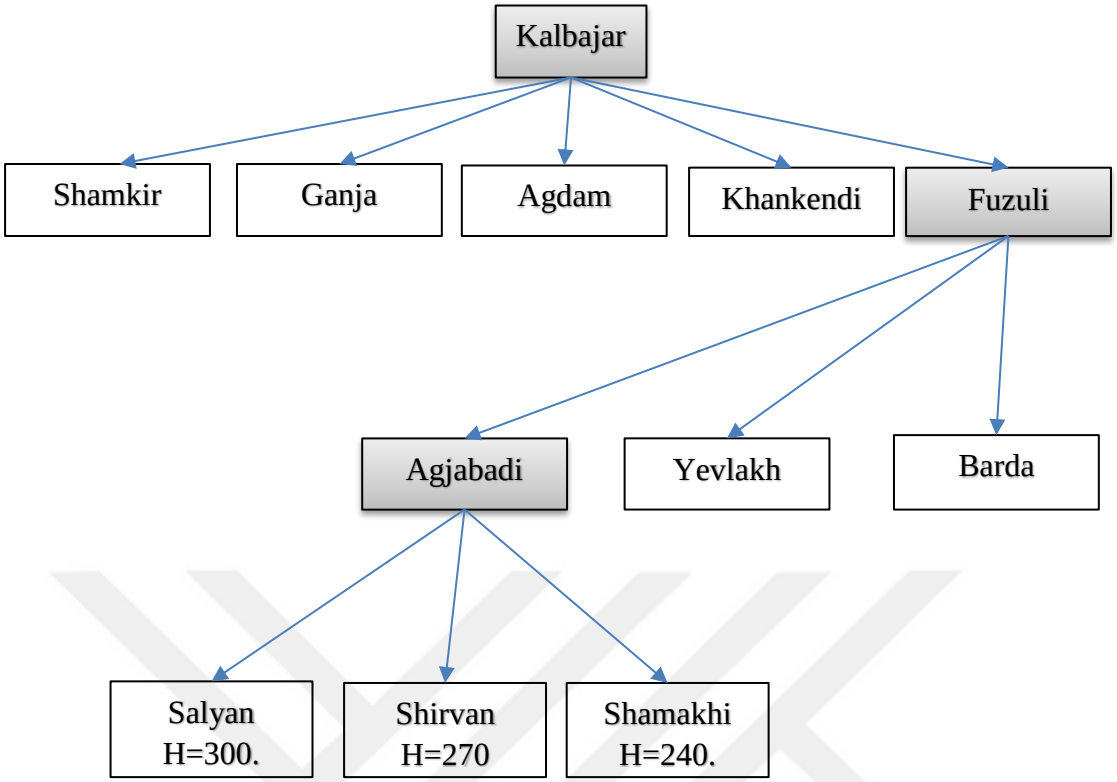
Şekil 8. Kalbajarın açılımı

sonuçları elde edilir. Bu sonuçlara göre en kısa yol (H=500) Fuzuli olduğundan bu düğüm açılarak bir sonraki düğümler elde edilir (Şekil 9).



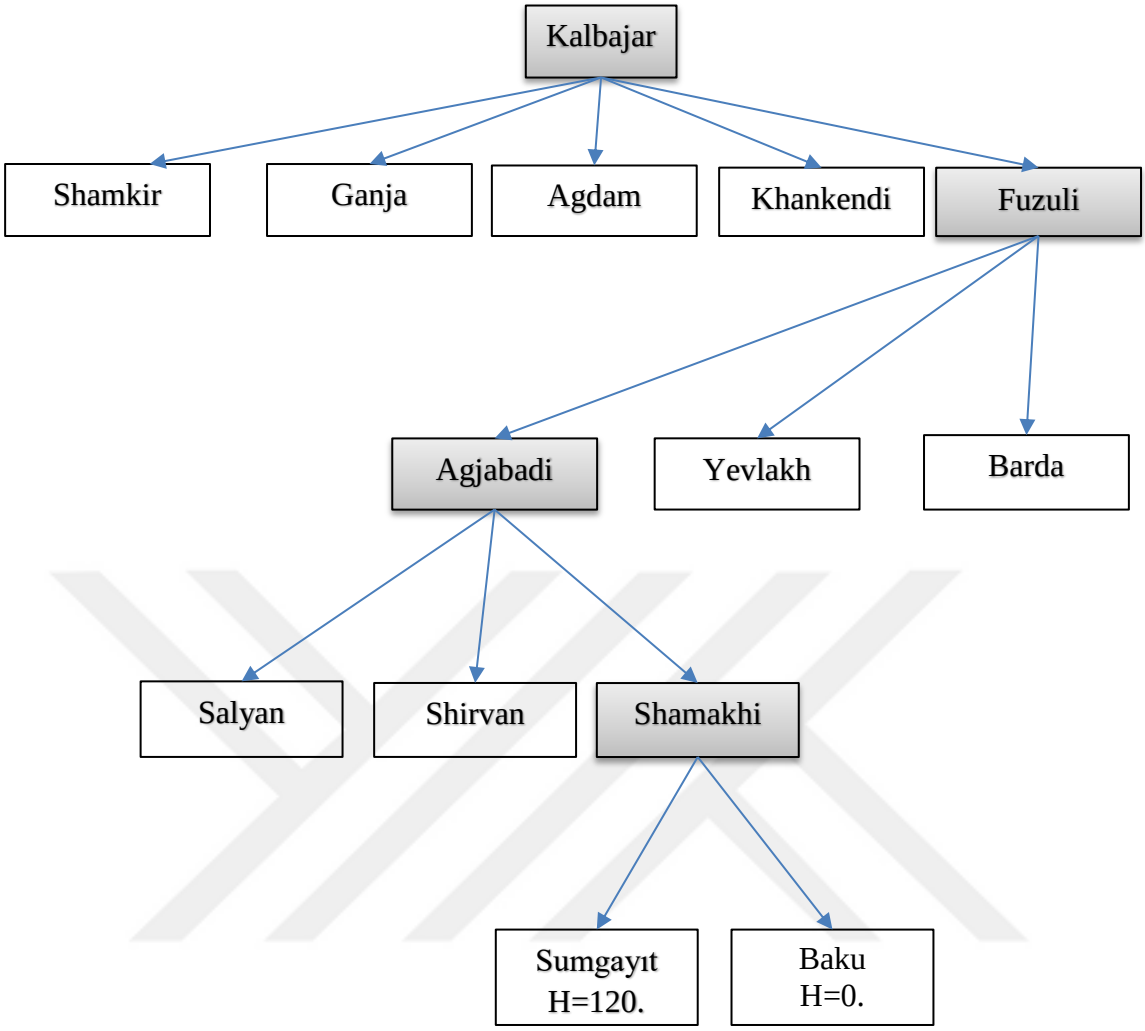
Şekil 9. Fuzulinin açılımı.

Fuzuli açıldığında en kısa yol (H=380) olarak Agjabadi olarak görülmektedir. Bir sonraki adımda Agjabadi açıldığında (Şekil 10) en kısa yol (H=240) olarak Shamakhi bulunur.



Şekil 10. Agjabadi düğümünün açılması.

Son olarak Shamakhi açıldığında Bakü'ye ulaşılmış olur (Şekil 11).



Şekil 11. Shamakhi düğümünün açılması

Bu örnekte görüldüğü gibi şehirler istenirliklerine göre azalan sırada sıralanarak sonuçlar elde edildi (Şekil 6).

1.2.2.2. A* Arama

Bu arama yöntemi aslında en iyi öncelikli arama yönteminin gelişmiş biçimdir. Temel düşüncesi, yüksek maliyetli yollara doğru genişleme yapmamaktır.

Değerlendirme fonksiyonu $f(n) = g(n) + h(n)$ biçiminde verilmektedir. Burada,

n – aramadaki her hangi durumdur.

$g(n)$ – başlangıç durumdan n durumuna kadar gidilmiş yolun maliyetidir.

$h(n)$ – n durumundan amaç durumuna kadar gereken maliyetin sezgisel tahminidir.

Bu bölümde en iyi öncelikli arama yönteminde kullanılan örneğin, A* arama yöntemi ile çözümü verilmiştir (Şekil 12).

Şekil 12. Azerbaycan'da Kalbajar ile Bakü arasındaki yol güzergahının belirlenmesinde A* yönteminin kullanılması.

Bakü ile diğer kentler arasındaki mesafeler Tablo 2 verilmiştir. Başlangıç durumu,

Kalbajar 600=0+600.

Şekil 13. Başlangıç durumu

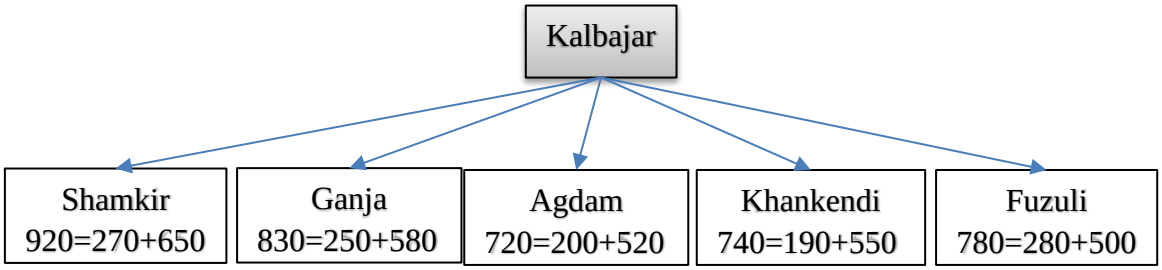
olarak verilmektedir. Burada,

Sol değer (600) – değerlendirme fonksiyonunun $f(n)$ değeri,

Sağ 1. değer (0)- başlangıç durumdan o anki duruma kadar alınan yol $g(n)$,

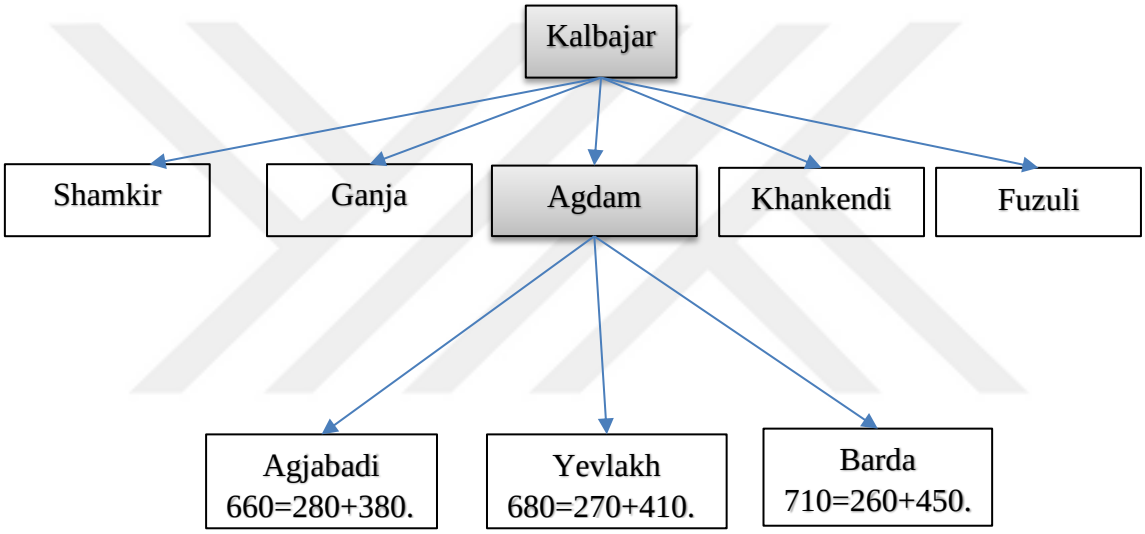
Sağ 2. değer (600)- o anki durumdan amaç durumuna kadar olan tahmini yol $h(n)$.

Öncelikle başlangıç düğümü olarak seçilen Kalbajar'ın açılımı ile diğer durumlar belirlenir ve bunların değerlendirme fonksiyonları belirlenir (Şekil 14).



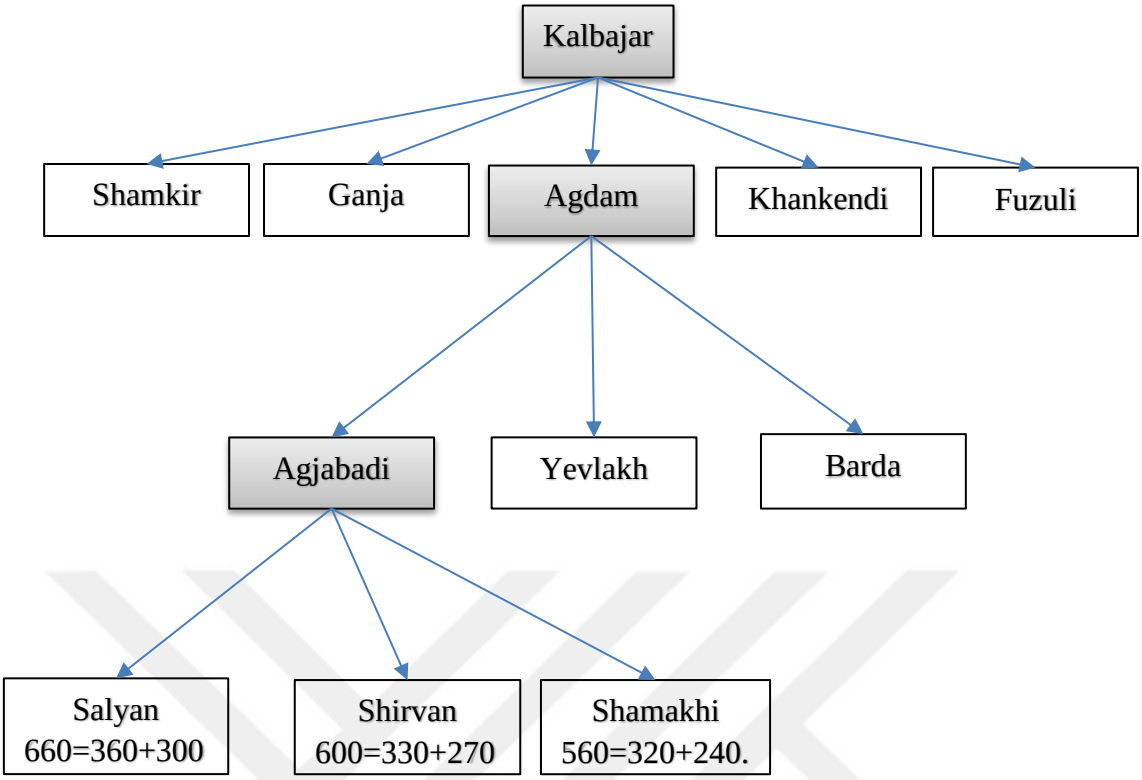
Şekil 14. Kalbajar'ın açılımı

Kalbajar'ın açılımından sonra değerlendirme fonksiyonu olarak en düşük maliyeti veren Agdam seçilerek bu düğümün açılımı yapılır (Şekil 15).



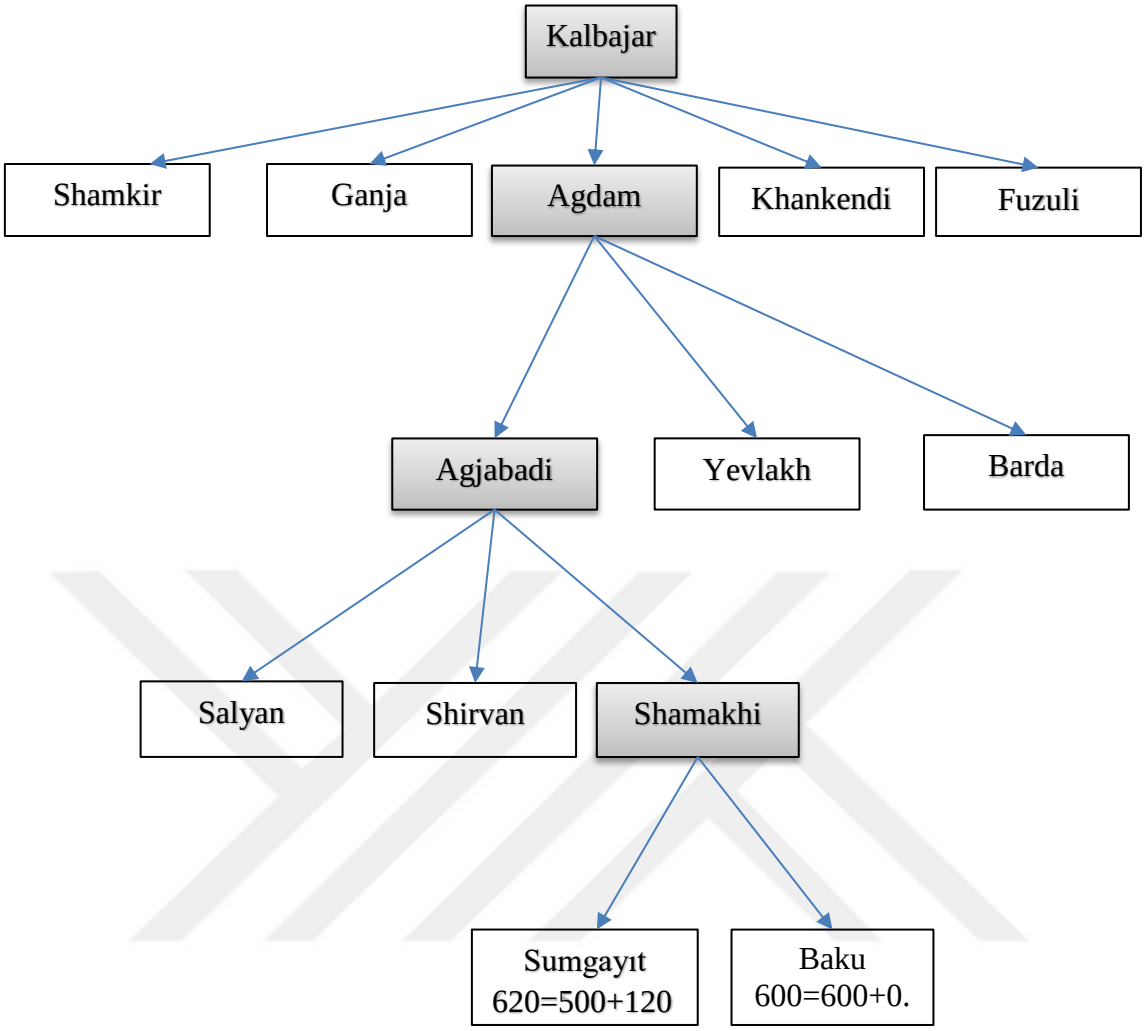
Şekil 15. Agdam'ın açılımı.

Agdam'ın açılımından sonraki düğümlere göre en düşük maliyetli değerlendirme değerine sahip şehir Agjabadi olduğundan bir sonraki açılan düğüm bu şehir olur (Şekil 16).



Şekil 16. Agjabadi'nin açılımı.

Agjabadi düğümünün açılımından sonra ki düğümlerin değerlendirme fonksiyonuna göre en optimal düğüm Shamakhi olduğu için bir sonraki düğüm olarak bu şehir seçilir (Şekil 17).



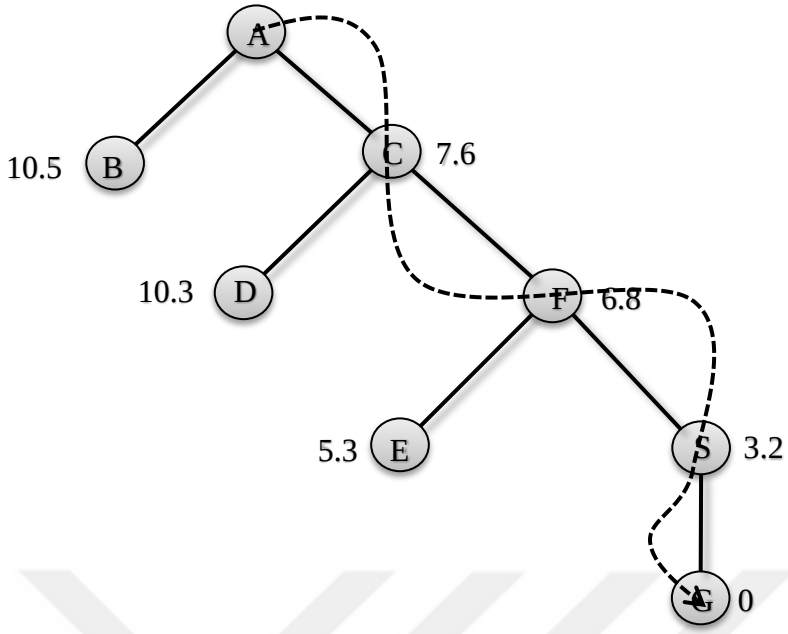
Şekil 17. Shamakhi'nin açılımı.

Shamakhi açılımından sonra Bakü'ye ulaşılmış olur. Değerlendirme fonksiyonuna göre de Bakü seçilir. Böylelikle A* arama yöntemine göre tüm yol güzergâhı belirlenmiş olur.

1.2.2.3. Tepe Tırmanma

Sezgisel değerlendirme, derinine arama yöntemini dağa tırmanma yöntemine dönüştürür. Aslında bu arama yöntemi En iyisini aramaya çok benziyor ondan farkı geri dönüşünün (backtracking) olmamasıdır. Dağa tırmanma yöntemi daima en küçük değerlendirme fonksiyonuna doğru hareket eder. Değerlendirme fonksiyonu $f(n) = d(n)$ olarak kabul ediliyor ve $d(n)$ güncel durumdan hedef duruma olan mesafedir.

Dağa tırmanma yöntemine örnek olarak Şekil 18'deki güzergah seçimi verilebilir.

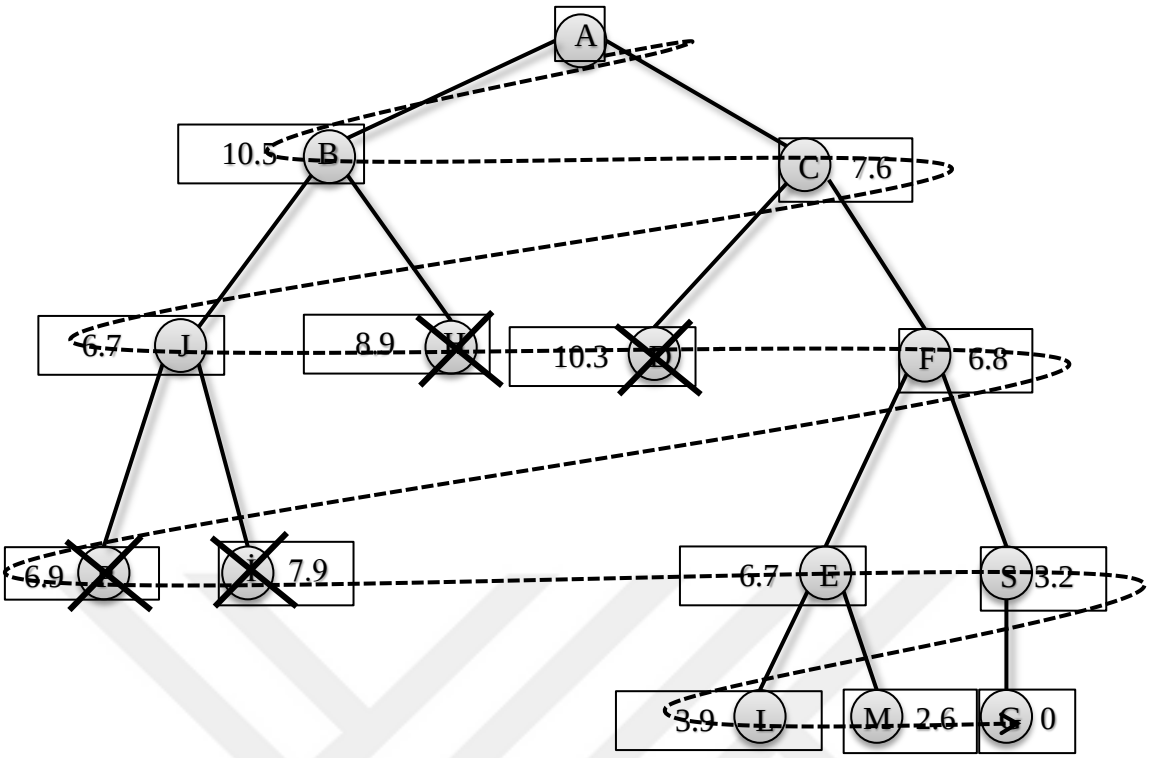


Şekil 18. Tepe tırmanma algoritmasının ağaç yapına göre gösterimi

Şekil 18'de düğümlerin yanındaki sayılar son (o anki) düğümlerden amaca kadar olan düz yolun uzunluğunu gösteriyor.

1.2.2.4. Işın Arama (Beam Search)

Enine aramaya benzemesine rağmen her seviyede genişleyen düğümlerin sayısında kısıtlama getirmekle zaman karmaşıklığı azaltmayı amaçlayan bir yöntemdir (Shapiro, 1992). Enine genişlenme sınırı w ($b >> w$) belirlenir. Bir sonraki genişlenmeler için en iyi w tane düğüm seçilerek düğümleri açılımı gerçekleştiriliyor (Şekil 19). Şekil 19'de $w = 2$ seçilerek ışın arama yöntemine göre arama stratejisi ağaç yapısında gösterilmektedir.



Şekil 19. Işın arama algoritmasının ağaç gösterimi.

Arama seviye-seviye gerçekleştiriliyor. Fakat her seviyede en iyi *w* (enine genişlenme sınırı) sayıda düğüm genişletilerek açılımlara devam edilir.

1.2.2.5. Dijkstra

Dijkstra Algoritması, graflar içinde birbirine bağlı düğümler arasında gidilebilen en kısa yolu bulmayı amaçlar. Algoritma, internet ağ trafiği protokolünü yönlendiren OSPF (Open Shortest Path First) protokolünde, oyun programlamada, ulaşım ağlarında kullanılır. Navigasyon cihazları iki nokta arasındaki en kısa yolu hesaplayabilmek için Hollanda' lı Matematikçi Dijkstra (1959) tarafından bulunan Dijkstra algoritmasını (Algoritma 1) kullanır. Bu algoritmanın temeli çizge teoremine (Graf Teorisi) dayanır. Kısa yol probleminde harita üzerindeki yerleşim yerleri (iller/ilçeler/köyler vb.) nokta olarak ve bu noktalara giden yollar çizgiler halinde ifade edilmektedir.

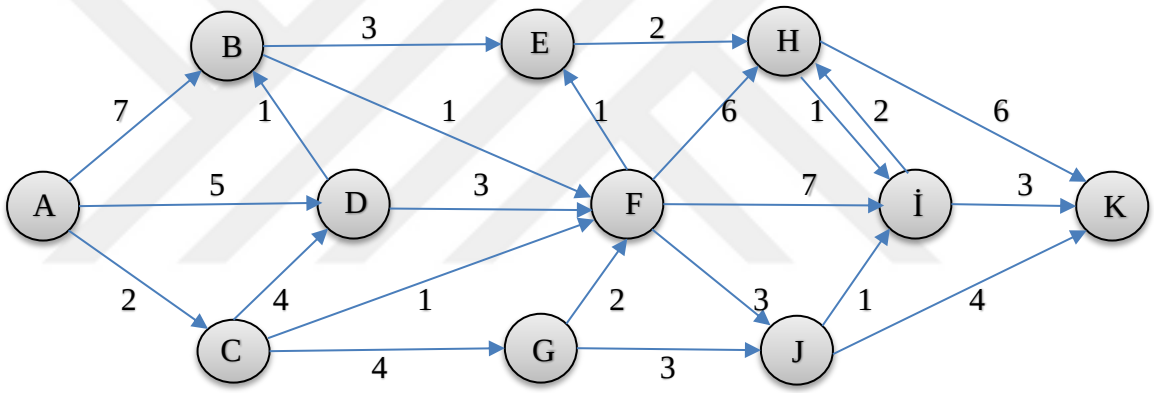
Algoritma 1. Dijkstra algoritması.

Adım 1. Kaynak düğüm belirlenir. Kaynak düğümden gidilebilen diğer düğümler seçilir.

Adım 2. Bu düğümlerden en az maliyete sahip olan işaretlenir, diğerleri aynı bilgiyle devam eder. Seçilmemiş düğümler için maliyet sonsuz olarak işaretlenir.

Adım 3. İkinci adımda seçilen maliyeti en az olan düğümden gidilebilen düğümler arasında aynı işlem uygulanır. Sonsuz işaretlenen düğümler bitinceye kadar devam edilir.

Dijkstra algoritmasının anlaşılabilirliğini artırmak için Şekil 20'deki örnek verilmiştir.



Şekil 20. Dijkstra algoritmasının graf üzerinde gösterimi.

Şekil 20'de grafta A düğümünden diğer düğümlere giden en kısa yolu Dijkstra algoritmasıyla bulanmaya çalışıldığında dikkat edilmesi gereken nokta; önceden işaretlenmiş düğümler için daha kısa mesafeye bakılmaz. Seçim her zaman işaretlenmemiş düğümler arasında yapılır (Tablo 3). Örneğin K'ya giden yol J ve I düğümlerinden daha kısa görünürken önceden işaretlenmediği için değiştirilmiştir. Bu algoritmaya göre işaretlenen düğümlerin komşu düğümlerinin yol mesafeleri Tablo 3'te verilmiştir.

Tablo 3. Dijkstra algoritmasının değerlendirme tablosu.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**İ**	**J**	**K**
	(0,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
A		(7,A)	(2,A)	(5,A)	(∞,A)	(∞,A)	(∞,A)	(∞,A)	(∞,A)	(∞,A)	(∞,A)
C		(7,A)		(4,C)	(∞,C)	(1,C)	(4,C)	(∞,C)	(∞,C)	(∞,C)	(∞,C)
F		(7,A)		(4,C)	(1,F)		(4,C)	(6,F)	(7,F)	(3,F)	(∞,F)
E		(7,A)		(4,C)			(4,C)	2/H	(7,F)	(3,F)	(∞,E)
H		(7,A)		(4,C)			(4,C)		2/İ	(3,F)	6/H
İ		(7,A)		(4,C)			(4,C)			(3,F)	3/İ

Bu tabloya göre A'dan sonra düğümler arası gidilen en kısa yol bulunabilir. Örneğin A – H arası gidilebilecek en kısa yol, tabloya göre A-C-F-E-H yolu izlenerek toplamda 2+1+1+2 olmak üzere 6 birim olarak bulunur.

1.3. Alt Kümeler

Sonlu örnek uzayları, sonlu sayıdaki elemandan oluşan bir kümeyi temsil eder. Bu kümeden rastgele çekilen herhangi bir örneklem aynı zamanda bir alt küme oluşturmaktadır.

Bu örneklemin belli bir kurala göre çekilmesi veya bu örnek uzayını bir kurala göre düzenlenmesi kurallı alt kümeler oluşmaktadır (Fischler ve Bolles, 1981). Kurallı alt küme tanımlamaları problemlere göre değişiklik göstermektedirler. Bu kurallı alt kümelerden en yaygın olanları permütasyon ve kombinasyon seçimleridir.

1.3.1. Permütasyon

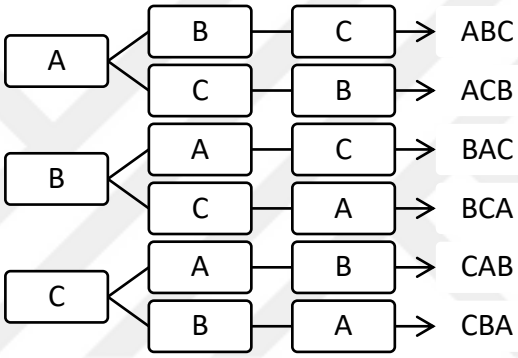
Bir kümedeki nesnelerin bir kısmının veya tümünü belli bir sıralamasına veya düzenlenmesine permütasyon denir (Hines vd., 2008).

Örnek olarak bir oturumda sunum yapacak 4 öğrenci kaç farklı sırada sunum yapabilir. Bu problemin çözümü için yapılacak en iyi yaklaşım ilk sıradan başlayarak koşullu bir yerleştirme yapmaktır. Yani birinci sırada 4 farklı, ikinci sırada 3 farklı, üçüncü sıra 2 farklı dördüncü sıra 1 farklı şekilde sunum yapılabilir. Bu durumda bunların çarpımı,

$$\mathrm{P} = 4 \cdot 3 \cdot 2 \cdot 1 = 24 \quad (1)$$

biçiminde permütasyonların sayısını verecektir.

Permütasyon yaklaşımında her nesne yalnızca bir kez kullanılabilir. Bu yaklaşım yerine koymandan yapılan örneklemeye bir örnek olarak verilebilir. Aynı zamanda hangi nesnenin hangi sırada geldiği de önemlidir (Akdeniz, 2016). Devşirme işlemini daha iyi anlaşılması için Şekil 21'deki ağaç yapısındaki gösterimi kullanılmaktadır. Bu ağaç yapısında {A, B, C} gibi üç elemanın 6 olası permütasyonu bulunduğu kolaylıkla bulanabilmektedir.



Şekil 21. Permütasyonların ağaç gösterimi

Şayet 3 tane eleman değil de n tane eleman verilmiş olsaydı, bu durumda aşağıdaki eşitlik yardımıyla tüm olası çözümlerin sayısı bulunabilir.

$$n \cdot (n-1) \cdot \ldots \cdot 3 \cdot 2 \cdot 1 = n! \quad (2)$$

Verilen kümedeki tüm elemanların kullanılması durumunda permütasyonların sayısı,

$$P(n) = n! \quad (3)$$

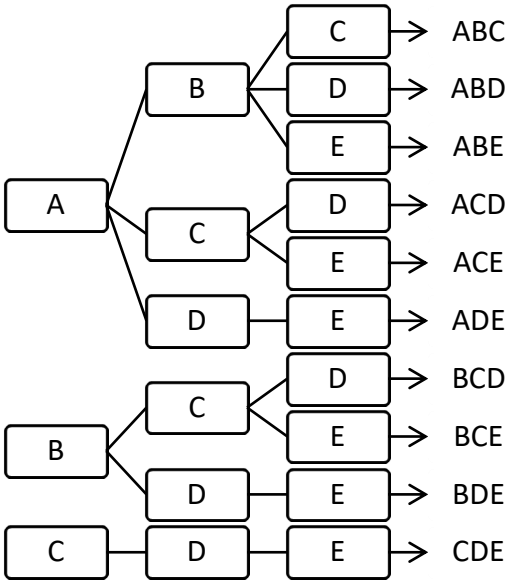
biçiminde bulunur. Verilen kümedeki tüm elemanları kullanılması yerine k tanesinin kullanılması ile elde edilebilecek tüm olası devşirmelerin sayısı,

$$P(n,k) = \frac{n!}{(n-k)!} \quad (4)$$

biçiminde bulunur.

1.3.2. Kombinasyon

Bir kümedeki n elemandan sıra gözetmeksizin k tane elemanın seçilmesidir (Hines vd., 2008). Sıra önemsiz olduğundan permütasyonda olduğu gibi yer doldurarak seçilemez. Çünkü aynı elemanların farklı sırada gelecek şekilde bir alt küme oluşturabilir. Bunu önlemek için geliştirilecek yaklaşım, kümede sırasıyla verilmiş elemanlardan alt küme elde ederken sıralar bozulmadan seçilmesi gerekir. Bu durumda ilk sıradaki eleman ilk $(n - k + 1)$ elaman olacaktır. Eğer $(n - k + 1)$ elemandan sonraki elemanlar seçilirse kurala göre diğer alt küme oluşturmak için yeterli eleman kalmayacaktır. İkinci sıraya birinci sırada seçilen elemandan sonra gelen elemanların seçilmesi gerekir. Aynı yaklaşımla k tane eleman seçilebilir. {A,B,C,D,E} gibi beş elemandan oluşan bir kümeden birbirinin tekrarı olamayan üç elemanlı kümelerin kombinasyonu Şekil 22'de ağaç yapısında gösterilmiştir.



Şekil 22. Kombinasyonların ağaç gösterimi

Ağaçtan görüldüğü gibi elde edilen tüm kombinasyonlarda ana kümedeki sıraya aykırı bir durum bulunmamaktadır. Bu gösterime dayanarak kombinasyonun

hesaplanmasında A ile başlayanlar 6 tanedir. Bunların B ile devam edenleri 3, C ile devam edenleri 2 ve D ile devam edenler 1 olmaktadır. Dallanan bir faktöriyel çarpımına göre kombinasyon eşitliği,

$$C(n,k) = \frac{n!}{k!\,(n-k)!} \tag{5}$$

biçiminde verilmektedir.

Örnek olarak sınıflar arasında yapılacak bir yarışma için her sınıftan 3 öğrenci seçilmesi gerekmektedir. 12 kişilik bir sınıftan 3 öğrenci kaç farklı şekilde seçilebilir. Burada seçilecek kişilerin sırasının önemli olmadığına dikkat ediniz.

2. YAPILAN ÇALIŞMALAR

Bilgisayar algoritmalarındaki en önemli yapılardan birisi özyinelemeli algoritmalardır. Bu algoritmalar genelde derinine aram yöntemleri için tercih edilen bir yaklaşım olmasına rağmen diğer arama yöntemlerinde yetersiz kalmaktadır. C/C++ programlama dillerinde sorunu çözmek için gösterge (pointer) yaklaşımı kullanılmaktadır. Ancak yaklaşımın karmaşık oluşu, programın okunurluğunu olumsuz etkilemektedir. Probleme daha okunaklı bir yaklaşım getirmek için son yıllarda geliştirilen programlama dillerinde Yığın (Stack) ve Kuyruk (Queue) gibi dinamik liste yapıları eklenmiştir. Bu dinamik liste yapıları birçok probleme kolay ve anlaşılır çözümler getirmiştir. Dolayısıyla bu çalışmada dört işlem kombinasyon probleminin çözümü için dinamik liste yapıları kullanılmıştır. Dört işlem kombinasyon çözümüne geçmeden önce klasik permütasyon ve kombinasyon problemlerinin liste yapıları yardımıyla çözümleri üzerinde durulacaktır.

2.1. Alt Küme Algoritmaları

Olasılık teorisinin temel yaklaşımı, uygun sonuçlarının sayısının örnek uzayındaki tüm sonuçların sayısına oranı olarak verilmektedir (Montgomery ve Runger, 2010). Olasılık değerini bulmak için olasılık teorisi ile ilgilenen araştırmacılar uygun sonuçların ve olası tüm sonuçların sayısı matematiksel olarak hesaplamak zorundadır. Verilen bir olayın olasılığını hesaplamak her zaman kolay olmamaktadır. Bu durumda araştırmacılar sonuca ulaşmak için uygun sonuçları ve tüm olası sonuçları ağaç yapısı kullanarak bulma yoluna gitmektedirler. Eğer ele alınan sonlu küme küçük boyutlarda ise bulunacak sonuçlar ele çözülebilir. Ancak küme boyutu büyüdükçe çözüme ulaşmakta zorlaşmaktadır. Günümüzde bilgisayarların hem kapasite hem de hız açısından gelişmesi bu gibi problemlerin çözümünde kolaylık sağlamaktadır.

2.1.1. Permütasyon Algoritması

Permütasyon probleminde genelde olası tüm örneklemlerin sayısı ile ilgilenilir. Bu çalışmada ise olası tüm örneklemlerin kümesi ile ilgilenilmektedir. Verilen bir kitleden (kümeden) olası tüm sıralı örneklemleri elde etmek için aşağıdaki algoritma kullanılabilir.

Algoritma 2. Tüm permütasyonların bulunması.

```
Adım 1.    Permütasyonları bulunacak kümeyi {a} belirle,
Adım 2.    {kullanılanlar, kalanlar} şekilde bir küme çifti belirle,
Adım 3.    {a} kümesini {kalanlar} kümesine ata,
Adım 4.    {kullanılanlar, kalanlar} küme çiftini listeye ekle,
Adım 5.    Döngüyü başlat,
Adım 6.    Listede küme çifti yoksa Adım 13'e git,
Adım 7.    Listeden sıradaki {kullanılanlar, kalanlar} küme çiftini al,
Adım 8.    {kalanlar} kümesinde eleman yoksa {kullanılanlar} kümesini sonuçlar
           dizisine ekle, Adım 5'e git,
Adım 9.    {kalanlar} kümesindeki her bir elemanı ayrı ayrı {kullanılanlar}
           kümesine ekle,
Adım 10.   {kullanılanlar} kümesine eklenen elemanı {kalanlar} kümesinden
           çıkar,
Adım 11.   {kullanılanlar, kalanlar} kümesini listeye ekle,
Adım 12.   Her {kullanılanlar, kalanlar} kümesini aynı işlemi uyguladıktan
           sonra Adım 5'teki döngüye geri dön,
Adım 13.   Son olarak elde edilen sonuçlar dizisini geri göndererek programı
           sonladır.
```

Algoritma 1'de n elemandan oluşan bir s tam sayılar dizisinden başlangıçta tüm elemanlar tek tek yığına eklenir. Bir sonraki döngüde yığına eklenen eleman hariç diğer elemanlar tek tek eklenerek yığına katılır. Bu işlem tüm elemanlar diziye eklenerek bulunur.

2.1.2. Kombinasyon Algoritması

Kombinasyon hesaplanırken kullanılacak yaklaşım, verilen kümedeki elemanlara birer sıra indisi verilerek permütasyonları hesaplanır. Permütasyonları hesaplanırken büyük indise sahip eleman küçük indise sahip elemandan önce gelemez koşulu konursa permütasyonların içerisinden kombinasyonlar seçilebilir. Burada dikkat edilmesi gereken nokta alt küme eleman sayısı istenen k elemanı aşmamalıdır.

Algoritma 3. Tüm kombinasyonların bulunması.

```
Adım 1.    Kombinasyonda bulunacak kümeyi {a} belirle,
Adım 2.    {kullanılanlar, kalanlar} şekilde bir küme çifti belirle,
Adım 3.    {a} kümesini {kalanlar} kümesine ata,
Adım 4.    {kullanılanlar, kalanlar} küme çiftini listeye ekle,
Adım 5.    Döngüyü başlat,
Adım 6.    Listede küme çifti yoksa Adım xx git,
Adım 7.    Listeden sıradaki {kullanılanlar, kalanlar} küme çiftini al,
Adım 8.    {kalanlar} kümesinde eleman yoksa {kullanılanlar} kümesini sonuçlar
           dizisine ekle, Adım 5 git,
Adım 9.    {kalanlar} kümesindeki her bir elemanı seç, kullanılanlar kümesinde
           son elemanla {a} kümesindeki indislerine göre karşılaştır seçilen
           elemanın indisi, son elemanın indisinden büyükse {kullanılanlar}
           kümesine ekle,
Adım 10.   {kullanılanlar} kümesine eklenen elemanı {kalanlar} kümesinden
           çıkar,
Adım 11.   {kullanılanlar, kalanlar} kümesini listeye ekle,
Adım 12.   Her {kullanılanlar, kalanlar} kümesini aynı işlemi uyguladıktan
           sonra Adım 5'teki döngüye geri dön,
Adım 13.   Son olarak elde edilen sonuçlar dizisini geri göndererek programı
           sonladır.
```

Kombinasyonlarda her alt kümede her eleman bir kez kullanılmaktadır. Aynı alt küme içeresinde aynı eleman iki ve daha fazla kullanımı durumunda tekrarlı kombinasyon seçeneği gerçekleşmiş olmaktadır. Bu çalışmada ise yalnızca temel kombinasyon kullanılacaktır.

2.2. Dört İşlem Kombinasyon Problemi

Dört işlem kombinasyon problemi, verilen n adet sayının dört işlem kullanılarak elde edilebilecek tüm sayılar kümesini bulma işlemi olarak ifade edilebilir. Bu problemde verilen sayılar yalnızca bir kez kullanılabilirken dört işlem operatörleri gerekli olduğu kadar kullanılabilir. Dört işlem probleminde bir hedef sayı belirlenip bu sayıya verilen bir sürede ulaşılmaya çalışılırsa problem bir yarışma oyununa dönüşmüş olur.

Bu bölümde dört işlem kombinasyon probleminde kullanılmak üzere bazı yardımcı işlemler tanımlanacaktır. Matematiksel ifadeler ne kadar sayısal değerlerden oluşsa da bunların saklanması ve gösteriminde metinsel (string) gösterimlerin kullanılması gerekmektedir. Bu durumda metinsel biçimdeki ifadelerin ayıklanması ve bunların değerlerinin belirlenmesi gerekecektir. Aşağıdaki alt bölümlerde bu işlemleri gerçekleştirmek için yardımcı yöntemler verilmiştir.

2.2.1. Metinsel İfadeden Ağaç Yapısına Dönüşüm (String2Tree)

Metin olarak kayıt edilen matematiksel ifadenin ağaç yapısına aktarılmasında öncelikle her operatöre bir öncelik sırası verilmelidir. Bu çalışmada, işlem öncelikler Tablo 4'de verilmiştir. Matematikte "+" ve "-" aynı önceliğe sahipken bu çalışmada farklı öncelik değeri verilerek işlem sırasındaki karmaşıklık önlenmiştir. Aynı şekilde "*" ve "/" için de farklı öncelikler verilmiştir. Öte yandan aynı önceliğe sahip iki veya daha fazla operatör aynı ifade de olması halinde sırasıyla soldan sağa doğru işlemler gerçekleştirilir. Ayrıca matematiksel ifade de her ayraç içerisine girildiğinde operatörlerin öncelik değerine 10 eklenirken, her ayraçtan çıkıldığında öncelik değeri 10 eksilir.

Tablo 4. Öncelik atamaları

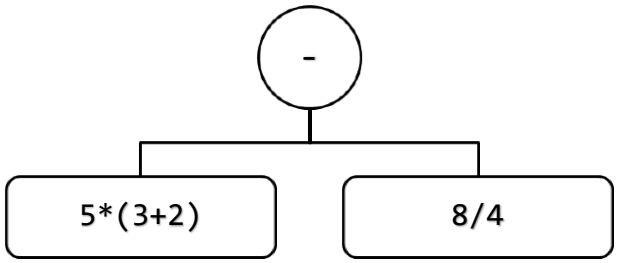
İşlem	**İşleç**	**Öncelik**
Toplama	+	1
Çıkarma	-	2
Çarpma	*	3
Bölme	/	4
Sol Ayraç	(	+10
Sağ Ayraç	)	-10

Örneğin $5 * (3 + 2) - 8/4$ matematiksel ifade bir metin dizisinde aşağıdaki tabloda verildiği gibi olsun. Bu tabloda operatörler dışındaki karakterin öncelikleri 0 olarak alınmıştır. Normalde "+" operatörünün öncelik değeri 1 olmasına rağmen ayraç içerisinde öncelik değeri 10 artarak 11 değerini almıştır.

Tablo 5. Örnek ifadenin öncelik dizisi

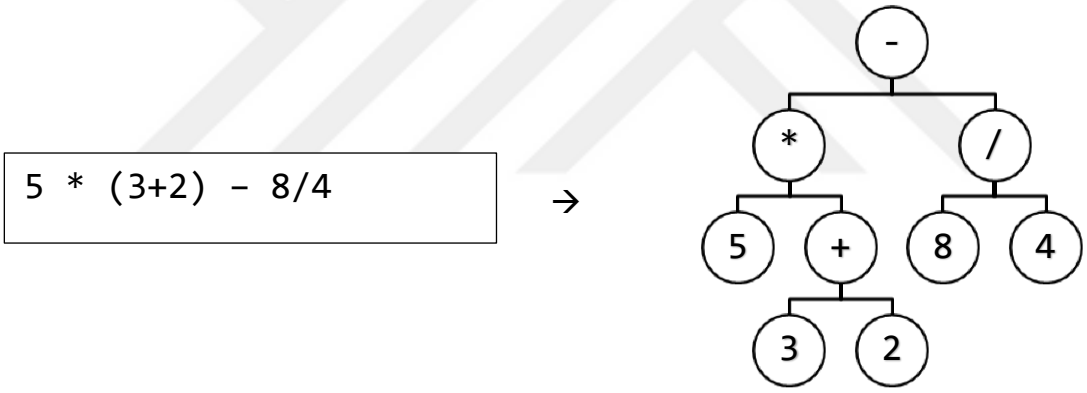
İfadeler	5	*	(	3	+	2	)	-	8	/	4
Öncelikler	0	3	0	0	11	0	0	2	0	4	0

Metin dizisinde sıfır olmayan en düşük önceliğe sahip operatör ağacın kök düğümü olurken bu operatörün solunda kalan ifade ile sağında kalan ifade bu düğümün alt (yavru) düğümleri olmaktadır (Şekil 23).



Şekil 23. Metinsel ifadeden ağaç yapısına dönüşüm

Sonraki adımlarda her alt düğüm kendi içerisinde alt dallanmalara ayrılarak alt düğüm içerisinde operatör kalmayana kadar devam eder. Sonuç olarak tüm uç dallanmalarda yalnızca sayılar kalmaktadır. Bazen uç dallanmalarda sayıların yanında ayraçlarda bulunabilmektedir. Eğer ifadenin içerisinde bir operatör yoksa o zaman ayraçlar bu ifadelerden silinerek ağaç son halini alır (Şekil 24).



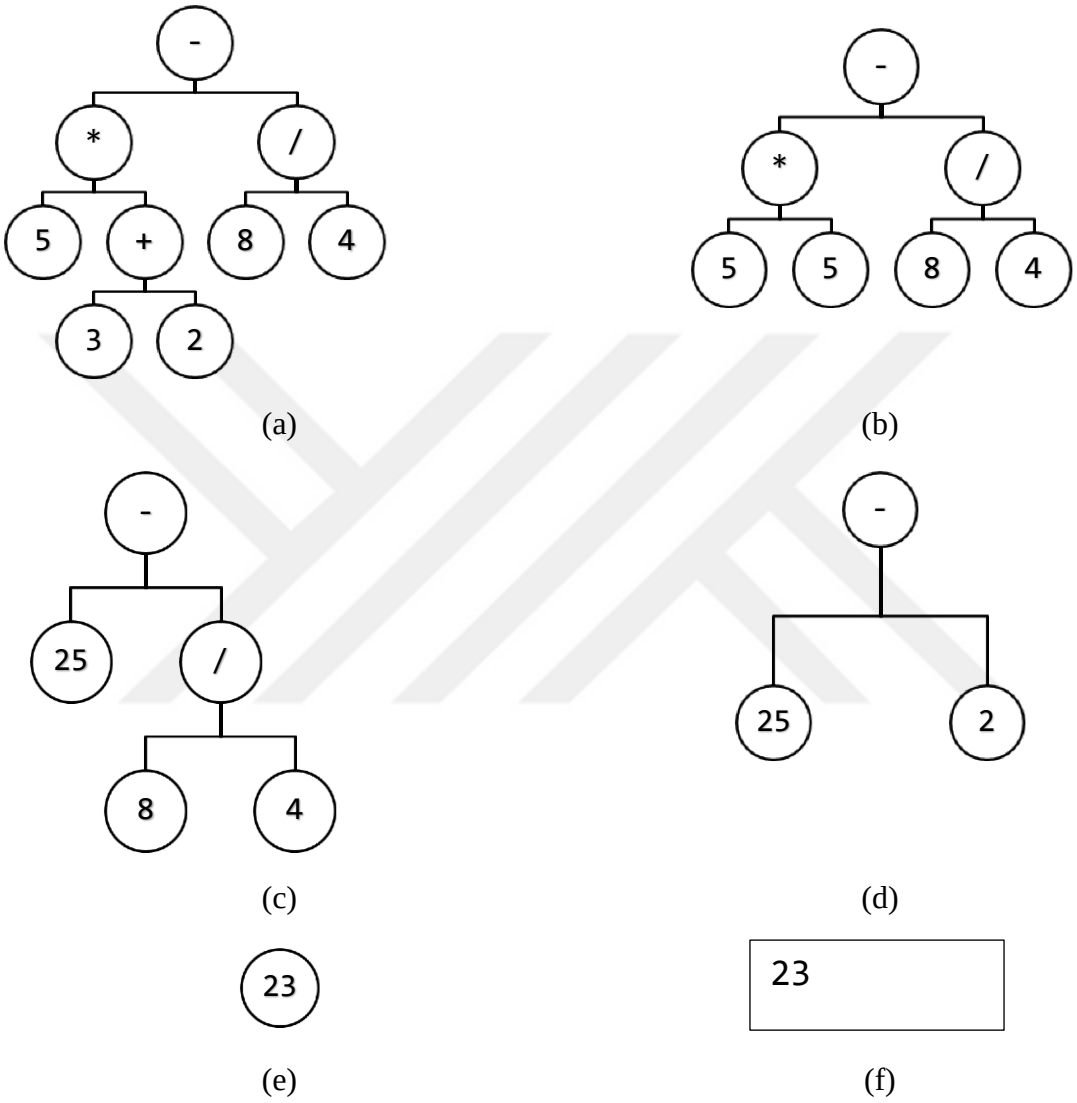
(a) (b)

Şekil 24. Metinsel ifadeden ağaç yapısına dönüşüm; (a) Metinsel ifade; (b) Ağaç yapısı

2.2.2. Ağaç Yapısından Değere Dönüşüm (Tree2Value)

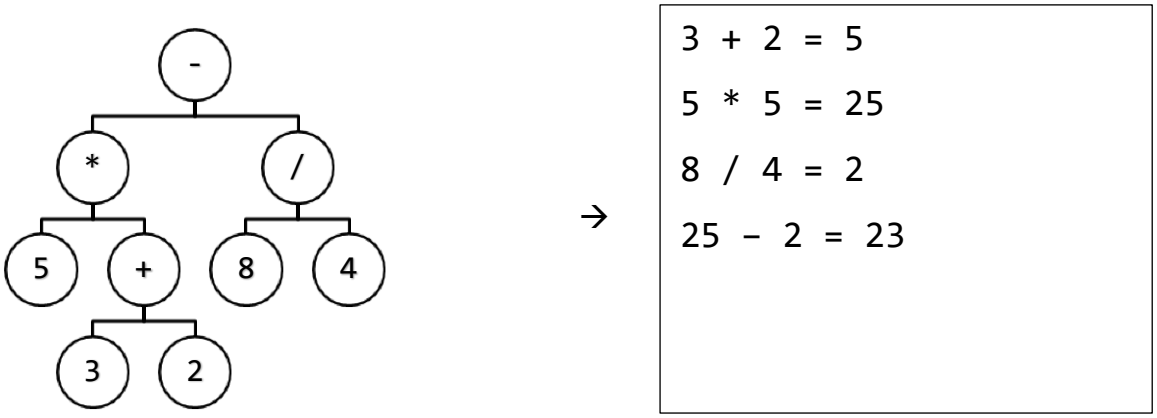
Metinsel olarak verilen bir matematiksel ifade bilgisayarda doğrudan hesaplanamaz. Bu hesaplamayı yapmak için önceki bölümde olduğu gibi metinsel ifade çözümlenerek bir ağaç yapısına dönüştürülür. Bu dönüştürme işleminde Yığın, Kuyruk ve Liste gibi veri yapılarından birisi kullanılabilir. Bu ağaç yapısında ara düğümler bir operatör içerirken, uç düğümler (yaprak düğümleri) birer sayı içermektedirler. Ağaç yapısına dönüştürülen matematiksel ifade en alt derinlikten başlayarak yukarı doğru (Şekil 25) işlem yapılarak sonuca ulaşılır. Her komşu yaprak düğümü bağladıkları operatör yardımıyla hesaplanır ve

çıkan değer operatörün bulunduğu düğüme yazılarak işleme sokulan yaprakların düğümleri silinir. En son en üste kalan düğüm bir yaprak düğümüne dönüşerek içerisinde bir değer barındırır ve bu değer ağacın işlem değeri olarak ifade edilir.



Şekil 25. Ağaç yapısından sayısal değere dönüşüm; (a) Başlangıç ağaç yapısı; (b) (3+2) işleminin yapılması; (c) (5*5) işleminin yapılması; (d) (8/4) işleminin yapılması; (e) (25-2) işleminin yapılması; (f) Geri dönüş değeri olarak 23 değerinin gönderilmesi.

Şekil 25'teki gibi bir ağaç yapısı bir değere dönüşürken her yapılan işlem adımı bir listeye (Şekil 26) kayıt edilerek yapılan işlemler daha net gözükebilir.

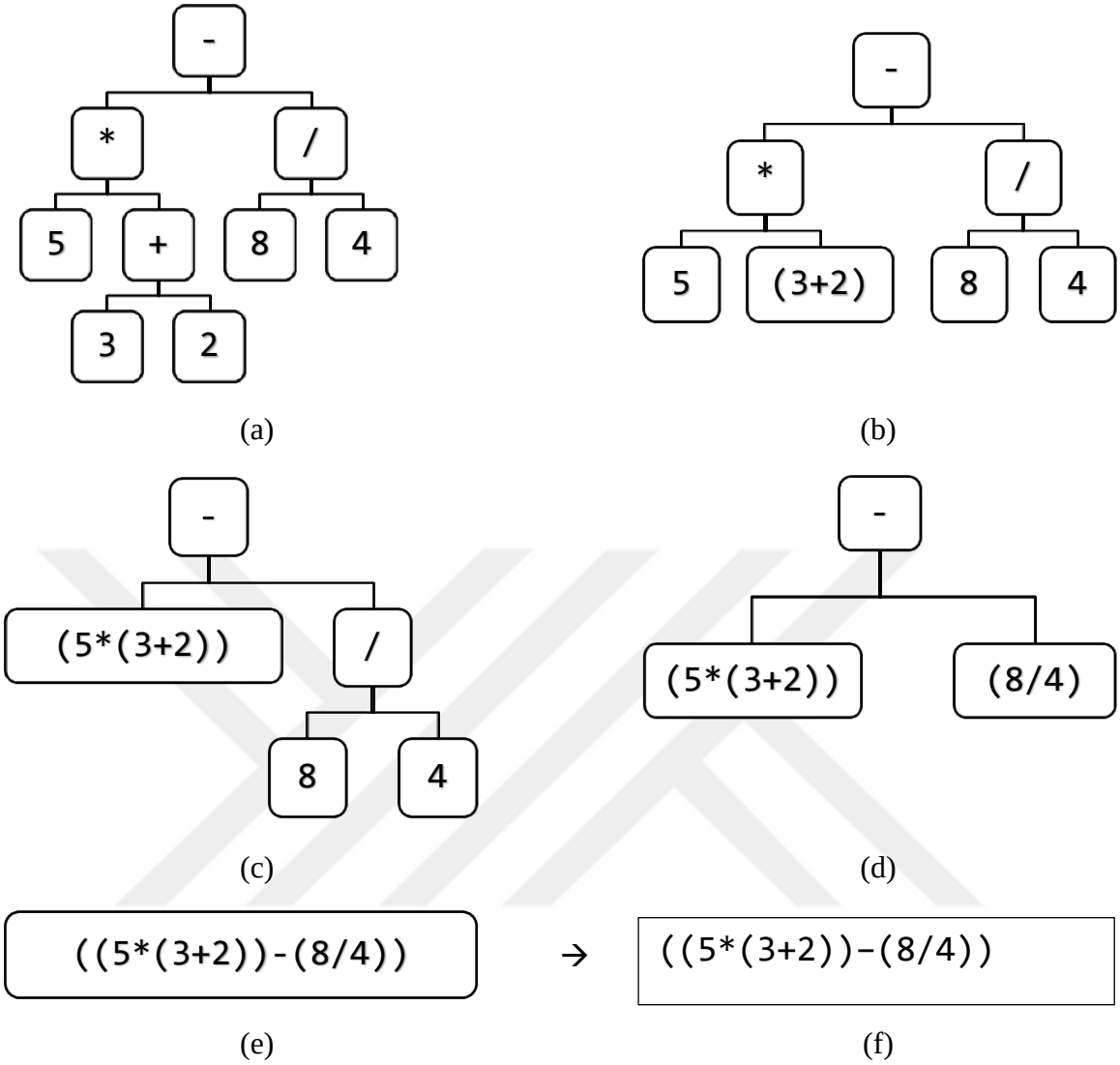


(a) (b)

Şekil 26. Ağaç yapısından adım adım işlemlerin gösterimi; (a) Ağaç yapısı; (b) Adım adım işlemler.

2.2.3. Ağaç Yapısından Metinsel İfadeye Dönüşüm (Tree2String)

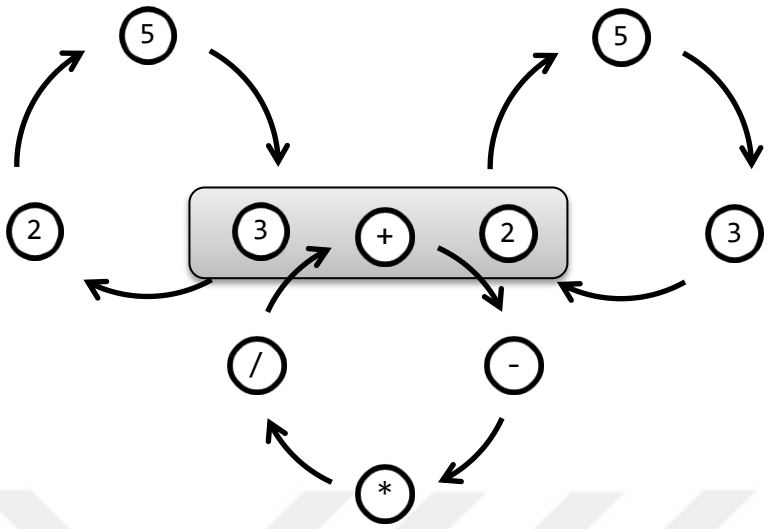
Kombinasyon problemlerini çözümünde genelde dinamik listeler kullanılmaktadır. Ancak bu yapıların saklanması ve gösterimi sıkıntılı bir süreçtir. Bu çalışmada dinamik listelerde ağaç yapısı saklanması gösteriminde kolaylık olması için metinsel yapıya dönüştürülmüştür (Şekil 27). Bu dönüşümde en alt yaprak düğümünden başlanarak bu düğümlerin bağlı oldukları operatörlerle birlikte oluşturdukları işlem öbeği operatörün bulunduğu düğüme kaydedilirken yaprak düğümleri listeden silinir. İşlem öbeği yerleştirilen operatör düğümü yeni durumda yaprak düğümüne dönüşür. Bu işleme devam edilerek tek bir düğüm kalana kadar devam edilir. Tek düğüm kaldığında ise bu düğüm geri gönderilerek metinsel ifade oluşturulmuş olur.



Şekil 27. Ağaç yapısından metinsel ifadeye dönüşüm; (a) Başlangıç ağaç yapısı; (b) (3+2) ifadesinin metne dönüşümü; (c) (5*(3+2)) ifadesinin metne dönüşümü; (d) (8/4) ifadesinin metne dönüşümü; (e) ((5*(3+2))-(8/4)) ifadesinin metne dönüşümü (f) tek düğüm kalan ağacın sonucunun metinsel geri dönüş değeri olarak gönderilmesi.

2.2.4. Tip-2 Ağaç Yapısı

Dört işlem kombinasyon problemi, permütasyon kombinasyon problemlerine oldukça benzemektedir. Ancak permütasyon ve kombinasyon problemlerinde bir kümeden belli bir kurala göre alt kümeler oluşturulurken dört işlemde probleminde farklı olarak iki kümenin kesişimi şeklinde bir liste oluşturulmaktadır (Şekil 28).



Şekil 28. Üç kümenin kesişimi ile oluşturulan işlem öbeği

Bu iki küme sayılar kümesi ve operatörler kümesi şeklinde tanımlanabilir. Sayılar kümesinin elemanlarının ikili permütasyonları ile operatör kümesinin tekli permütasyonları kullanılarak işlem öbekleri oluşturulabilir. Ancak sayılar kümesinin ikili permütasyonları operatörler kümesinin tekli permütasyonu ile birleştiğinde (3+2) ve (2+3) gibi bir işlem öbeği aynı işlem olduğundan tekrarlı alt kümeler oluşmasına neden olur. Problemi çözmek için "-" ve "/" operatörleri için permütasyon çiftleri kullanılırken, "+" ve "*" için kombinasyon çiftleri kullanılmıştır (Tablo 6).

Tablo 6. İşlem tabloları

+	A	B	C
A	*A+A*	**A+B**	**A+C**
B	B+A	*B+B*	**B+C**
C	C+A	C+B	*C+C*

(a)

*	A	B	C
A	*A*A*	**A*B**	**A*C**
B	B*A	*B*B*	**B*C**
C	C*A	C*B	*C*C*

(b)

-	A	B	C
A	*A-A*	**A-B**	**A-C**
B	**B-A**	*B-B*	**B-C**
C	**C-A**	**C-B**	*C-C*

(c)

/	A	B	C
A	*A/A*	**A/B**	**A/C**
B	**B/A**	*B/B*	**B/C**
C	**C/A**	**C/B**	*C/C*

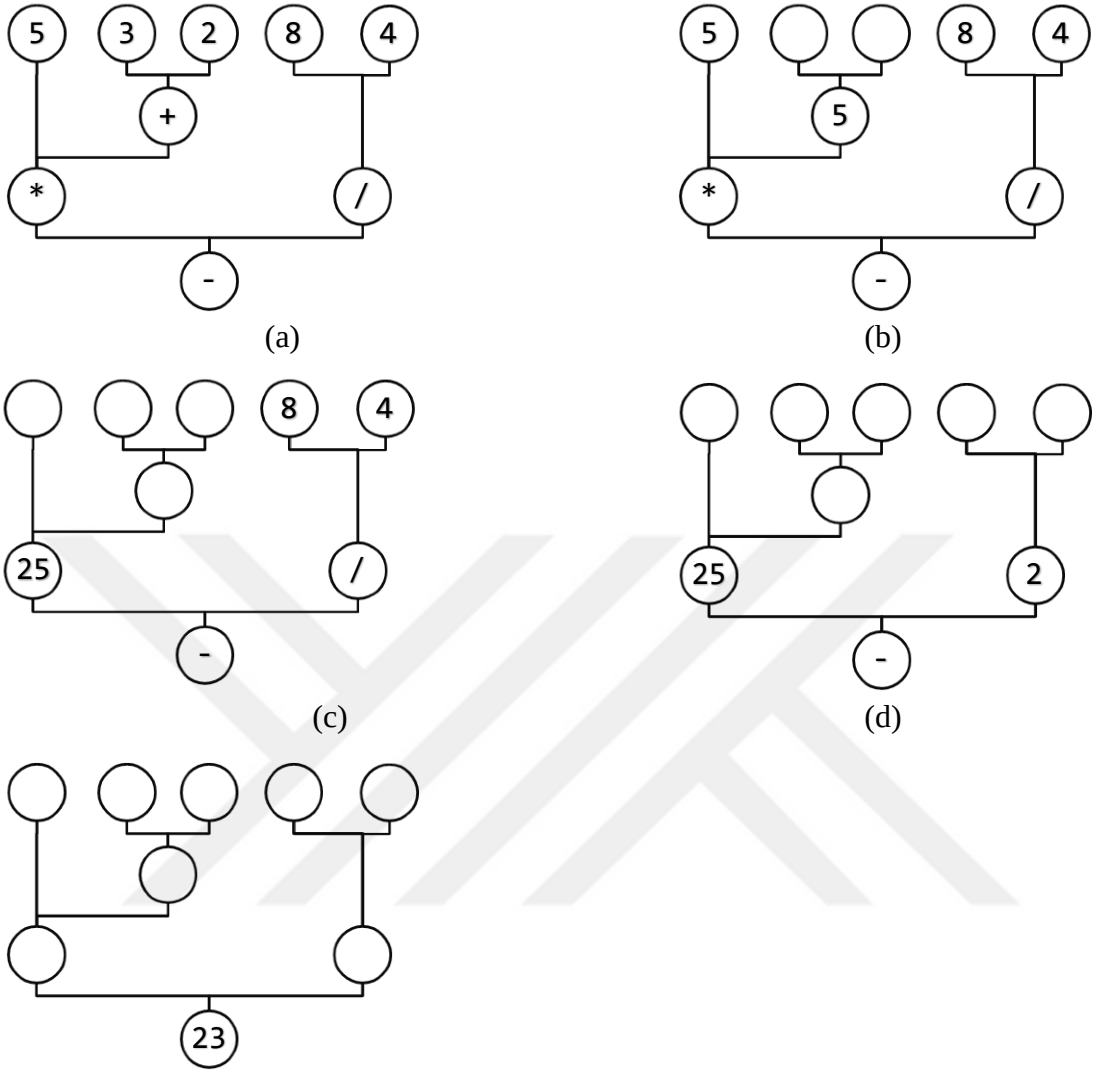
(d)

Tablo 6'da verilen işlem öbeklerinde {A,B,C} elemanlarında oluşan bir kümenin ikili işlem öbek sayısı belirlenirken kullanılabilir. Bu tablo yardımıyla istenilen işlem öbeği alt küme kuralı oluşturulabilir. Bu tabloda her operatörün oluşturduğu ikili işlem öbeklerinde eğik yazı biçimi ile gösterilen işlem öbekleri tekrarlı kombinasyon istenirse listeye eklenebilir. Bu çalışmada tekrarlı kombinasyon kullanılmamıştır. Öte yandan aynı işlem sonucu verme olasılığı olan işlem öbekleri düz yazı biçimi ile gösterilirken tekil işlem sonucuna sahip işlem öbekleri ise koyu yazı biçiminde gösterilmiştir.

Dört işlem kombinasyon probleminde birden çok küme kullanılması durumunda, permütasyon ve kombinasyon listelerinden farklı olarak liste içerisinde liste kullanma ihtiyacı doğmuştur. Bu durumda liste içerisinde liste kayıt edilmesi durumunda üst listenin sahip olduğu ağacın her bir düğümünde tekrar bir ağaç yerleştirilmesi gerekir.

2.2.5. Dört İşlem Kombinasyon Probleminin Çözümü

Dört işlem kombinasyon probleminin çözümü için geliştirilen tip-2 ağaç yapısındaki çözümün daha basit bir algoritma ile çözümü için alt ağaç yapısı metinsel biçimde kayıt edilerek listeye eklenir ve ağacın genişletilmesine (dallanmasına) devam edilir (Algoritma 4). İşlem sonucunda elde edilen metin listesi istenirse bölüm 2.2.1-2.2.3'te verilen işlemler kullanılarak sayısal değerler bulunabilir. Şayet metin listesi yerine işlem sonucu kullanılacaksa Şekil 29'daki ters ağaç yöntemi kullanılarak her işlem aşamasındaki düğümde sayılardan oluşan bir liste tutulur ve bir alt seviyeye geçildikçe işlem yapılan öbeklerin değeri listeye eklenir. Aynı zamanda işlem öbeklerinde kullanılan sayılar da listeden çıkartılır (Şekil 29).



Şekil 29. Ters ağaç yapısı ile ağaçtan değere ulaşmanın ağaçsal gösterimi; (a) başlangıç ağaç yapısı; (b) (3+2) işleminin yapılması; (c) (5*5) işleminin yapılması; (d) (8/4) işleminin yapılması; (e) (25-2) işleminin yapılması

Algoritma 4. Dört işlem kombinasyon probleminin tip-2 ağaç yapısı ile çözümü.

```
Adım 1.    Dört işlem yapılacak sayılar kümesini {a} belirle,
Adım 2.    Kalanlar diye bir küme tanımla
Adım 3.    Sayılar kümesini kalanlar kümesine ata,
Adım 4.    {kalanlar} kümesini listeye ekle,
Adım 5.    Döngüyü başlat,
Adım 6.    Listede kalan yoksa Adım 14'e giderek işlemi sonlandır,
Adım 7.    Listeden sıradaki {kalanlar} kümesini al,
Adım 8.    Kalanlar kümesindeki sayı adeti 1 ise bu sayıyı sonuç dizisine
           ekle,
Adım 9.    Kalanlar kümesindeki sayıların tüm ikili permütasyonlarını bul,
Adım 10.   Seçilen ikili permütasyon çiftindeki sayıları kalan listesinden
           çıkar,
Adım 11.   Bulunan permütasyon çiftlerine dört işlem uygulayarak çıkan
           sonuçları kalanlar kümesine ekle,
Adım 12.   Kalanlar kümesini listeye ekle,
Adım 13.   Her permütasyon çiftine aynı işlemi uyguladıktan sonra Adım 5'teki
           döngüye geri dön,
Adım 14.   Son olarak elde edilen sonuçlar dizisini geri göndererek programı
           sonladır.
```

Dört işlem kombinasyon probleminin basit bir örnek üzerindeki uygulaması için {5, 2, 3} elemanlarından oluşan bir küme ele alınsın. Birinci aşamada 3 elemanlı kümedeki elamanların ikili permütasyonları Tablo 7'deki gibi bulunur.

Tablo 7. Üç elemanlı kümenin ikili işlem öbekleri

İkili permütasyonlar	{5, 2}	{5, 3}	{2, 5}	{2, 3}	{3, 5}	{3, 2}
Kalanlar	3	2	3	5	2	5
Toplama işlemi	**{(5+2),3}**	**{(5+3),2}**	*{(2+5),2}*	**{(2+3),5}**	*{(3+5),2}*	*{(3+2),5}*
Çıkarma işlemi	**{(5-2),3}**	**{(5-3),2}**	**{(2-5),2}**	**{(2-3),5}**	**{(3-5),2}**	**{(3-2),5}**
Çarpma işlemi	**{(5*2),3}**	**{(5*3),2}**	*{(2*5),2}*	**{(2*3),5}**	*{(3*5),2}*	*{(3*2),5}*
Bölme işlemi	**{(5/2),3}**	**{(5/3),2}**	**{(2/5),2}**	**{(2/3),5}**	**{(3/5),2}**	**{(3/2),5}**

Tablo 7'de toplama ve çarpma işleminin bulunduğu satırlarda aynı sonucu veren işlemlerden yalnızca birisi seçildiği için bu satırlardaki sonuçlardan üçü alınırken, çıkarma ve bölme işlemlerinin bulunduğu satırlardaki tüm elemanlar alınmaktadır. Böylelikle üç elemanın ikili dört işlem kombinasyonu olarak 18 tane yeni küme bulunduğu gibi 18 tanede işlem yapılmış ve sonuç elde edilmiştir. Bu 18 işlemin sonucunda elde edilen yeni iki elemanlı kümelerin dört işlem kombinasyonu ise aşağıdaki listelerde verilmiştir.

{ (5 + 2) , 3 } kümesinin sonraki işlem üçlüsü

(3 + (5 + 2)) (3 * (5 + 2)) (3 - (5 + 2)) (3 / (5 + 2)) ((5 + 2) - 3) ((5 + 2) / 3)

{ (5 - 2) , 3 } kümesinin sonraki işlem üçlüsü

(3 + (5 - 2)) (3 * (5 - 2)) (3 - (5 - 2)) (3 / (5 - 2)) ((5 - 2) - 3) ((5 - 2) / 3)

{ (5 * 2) , 3 } kümesinin sonraki işlem üçlüsü

(3 + (5 * 2)) (3 * (5 * 2)) (3 - (5 * 2)) (3 / (5 * 2)) ((5 * 2) - 3) ((5 * 2) / 3)

{ (5 / 2) , 3 } kümesinin sonraki işlem üçlüsü

(3 + (5 / 2)) (3 * (5 / 2)) (3 - (5 / 2)) (3 / (5 / 2)) ((5 / 2) - 3) ((5 / 2) / 3)

{ (5 + 3) , 2 } kümesinin sonraki işlem üçlüsü

(2 + (5 + 3)) (2 * (5 + 3)) (2 - (5 + 3)) (2 / (5 + 3)) ((5 + 3) - 2) ((5 + 3) / 2)

{ (5 - 3) , 2 } kümesinin sonraki işlem üçlüsü

(2 + (5 - 3)) (2 * (5 - 3)) (2 - (5 - 3)) (2 / (5 - 3)) ((5 - 3) - 2) ((5 - 3) / 2)

{ (5 * 3) , 2 } kümesinin sonraki işlem üçlüsü

(2 + (5 * 3)) (2 * (5 * 3)) (2 - (5 * 3)) (2 / (5 * 3)) ((5 * 3) - 2) ((5 * 3) / 2)

{ (5 / 3) , 2 } kümesinin sonraki işlem üçlüsü

(2 + (5 / 3)) (2 * (5 / 3)) (2 - (5 / 3)) (2 / (5 / 3)) ((5 / 3) - 2) ((5 / 3) / 2)

{ (2 - 5) , 3 } kümesinin sonraki işlem üçlüsü

(3 + (2 - 5)) (3 * (2 - 5)) (3 - (2 - 5)) (3 / (2 - 5)) ((2 - 5) - 3) ((2 - 5) / 3)

{ (2 / 5) , 3 } kümesinin sonraki işlem üçlüsü

(3 + (2 / 5)) (3 * (2 / 5)) (3 - (2 / 5)) (3 / (2 / 5)) ((2 / 5) - 3) ((2 / 5) / 3)

{ (2 + 3) , 5 } kümesinin sonraki işlem üçlüsü

(5 + (2 + 3)) (5 * (2 + 3)) (5 - (2 + 3)) (5 / (2 + 3)) ((2 + 3) - 5) ((2 + 3) / 5)

{ (2 - 3) , 5 } kümesinin sonraki işlem üçlüsü

(5 + (2 - 3)) (5 * (2 - 3)) (5 - (2 - 3)) (5 / (2 - 3)) ((2 - 3) - 5) ((2 - 3) / 5)

{ (2 * 3) , 5 } kümesinin sonraki işlem üçlüsü

(5 + (2 * 3)) (5 * (2 * 3)) (5 - (2 * 3)) (5 / (2 * 3)) ((2 * 3) - 5) ((2 * 3) / 5)

{ (2 / 3) , 5 } kümesinin sonraki işlem üçlüsü

(5 + (2 / 3)) (5 * (2 / 3)) (5 - (2 / 3)) (5 / (2 / 3)) ((2 / 3) - 5) ((2 / 3) / 5)

{ (3 - 5) , 2 } kümesinin sonraki işlem üçlüsü

(2 + (3 - 5)) (2 * (3 - 5)) (2 - (3 - 5)) (2 / (3 - 5)) ((3 - 5) - 2) ((3 - 5) / 2)

{ (3 / 5) , 2 } kümesinin sonraki işlem üçlüsü

(2 + (3 / 5)) (2 * (3 / 5)) (2 - (3 / 5)) (2 / (3 / 5)) ((3 / 5) - 2) ((3 / 5) / 2)

{ (3 - 2) , 5 } kümesinin sonraki işlem üçlüsü

(5 + (3 - 2)) (5 * (3 - 2)) (5 - (3 - 2)) (5 / (3 - 2)) ((3 - 2) - 5) ((3 - 2) / 5)

{ (3 / 2) , 5 } kümesinin sonraki işlem üçlüsü

(5 + (3 / 2)) (5 * (3 / 2)) (5 - (3 / 2)) (5 / (3 / 2)) ((3 / 2) - 5) ((3 / 2) / 5)

Üç elemanlı kümenin ikili işlem sayısı 18 olurken, bu 18 işlem sonucunun üçlü işlem sayısı her biri için 6 tane sonuç vermektedir. Sonuç olarak üçlü bir kümenin üçlü işlemlerinin sayısı 6*18 = 108 olarak bulunmaktadır. Ayrıca önceki adımda ikili işlem sonucu elde edilen sonuçlarda dahil edilirse üç elemanlı bir kümenin tüm dört işlem kombinasyon sayısı 18+108 = 126 olarak bulunur.

2.2.6. Dört İşlem Kombinasyon Sayısının Belirlenmesi

Dört işlem kombinasyon probleminde sayılar kümesi ile operatörler kümesinin kesişim (karma) kümesine göre elde edilen toplam işlem öbeklerinin sayı ve bunlardan elde edilen yeni sayıların sayısı belirlemek için permütasyon sayısına ihtiyaç vardır. Bu durumda n elemandan oluşan bir kümeden elde edilecek işlem sayısı,

$$D(n,2) = 3.P(n,2) \tag{6}$$

biçiminde bulunmaktadır. Burada n elemanın öncelikle ikili permütasyonu $P(n,2)$ bulunur. bu permütasyondan çıkarma ve bölme işlem permütasyonlarının hepsi alınırken, toplama ve çarpma işlemlerinin yarı alındığı için 4 kat olan permütasyonların işlem sayısı 3 kata inmektedir. Aynı şekilde n elemanlı bir sayı kümesinden üçlü işlem sayıyı ise

$$D(n,3) = 3^2.P(n,2).P(n-1,2) \tag{7}$$

biçiminde hesaplanabilir. İşleme devam edilerek n elemanlı bir sayı kümesinden elde edilecek n'li işlem sayısı

$$D(n,n) = 3^{n-1}.P(n,2) \ldots . P(4,2).P(3,2).P(2,2) \tag{8}$$

biçiminde hesaplanabilir. Formülasyonu genelleştirirsek,

$$D(n,k) = 3^{k-1} \prod_{t=n-k+2}^{n} P(t,2) \tag{9}$$

biçiminde bir eşitlik elde ederiz. Elde edilen bu eşitlikle n elemanlı bir kümeden k işlemlerin sayısı bulunmuş olur. Şayet n elamanlı bir sayı kümesinden elde edilecek tüm işlemlerin sayısı bulunmak istenirse,

$$D(n) = \sum_{k=2}^{n} \left(3^{k-1} \prod_{t=n-k+2}^{n} P(t,2) \right) \tag{10}$$

eşitliği kullanılabilir. (10) eşitliği kullanılarak sayı kümesinin 2, 3, 4, 5, 6 ve 7 elemanlı olması durumunda işlem sayıları bulunarak Tablo 8'de gösterilmiştir.

Tablo 8. Dört işlem sonuç sayıları

D(2,2)	6
D(2) = D(2,2)	6
D(3,2)	18
D(3,3)	108
D(3) = D(3,2)+D(3,3)	126
D(4,2)	36
D(4,3)	648
D(4,4)	3888
D(4) = D(4,2)+D(4,3)+D(4,4)	4572
D(5,2)	60
D(5,3)	2160
D(5,4)	38880
D(5,5)	233280
D(5) = D(5,2)+D(5,3)+D(5,4)+D(5,5)	274380
D(6,2)	90
D(6,3)	5400
D(6,4)	194400
D(6,5)	3499200
D(6,6)	20995200
D(6) = D(6,2)+D(6,3)+D(6,4)+D(6,5)+D(6,6)	24694290
D(7,2)	126
D(7,3)	11340
D(7,4)	680400
D(7,5)	24494400
D(7,6)	440899200
D(7,7)	2645395200
D(7) = D(7,2)+D(7,3)+D(7,4)+D(7,5)+D(7,6)+D(7,7)	3111480666

3. BULGULAR VE SONUÇLAR

Bu çalışmada dört işlem kombinasyon problemi için önerilen algoritmanın başarımını test etmek için benzetim çalışmaları yapılmıştır. Bu benzetim çalışması için Intel® Core ™ i7-3630QM CPU @ 2.40GHz işlemci ve 8GB bellek içeren 64 bitlik bir bilgisayar kullanılmıştır. Yazılım olarak Visual Studio 2012 C# yazılımı kullanılmıştır.

Benzetim parametresi olarak 1 ile 9 arasında rastgele sayılar seçilmiş ve bunlardan elde edilen sonuçları kayıt edilmiştir. Seçilen sayı adetine göre ortalama hesaplama zamanları Tablo 9'te verilmiştir.

Tablo 9. Önerilen algoritmanın hesaplama zamanı

Kullanılan sayı adeti	Deneme sayısı	Üretilen sonuç sayısı	Ortalama Hesaplama Zamanı (sn)
2	100	6	0.00025359
3	100	126	0.00094052
4	100	4572	0.03307742
5	100	274380	66.66526977

Tablo 9'da kullanılan sayı adeti kadar 1ile 9 arasında rastgele sayılar çekilerek bunlardan elde edilen sonuçların hesaplama zamanı bulunmuştur. Bu işleme 100 defa tekrar edilerek ortalama hesaplama zamanına ulaşılmıştır. Bu tabloda sayı kümesindeki elemanlar artıkça üretilen sonuç sayısı ve hesaplama zamanı üstel olarak artış gösterdiği gözlenmiştir. Bu denemeler sonucunda elde edilen sayı dizilerinden elde edilen istatistikler Tablo 10'da verilmiştir. Bu tabloda n elemanlı bir sayı kümesinden elde edilen işlem sayısı N sütununda verilmiştir. Sonraki sütunlarda 100 deneme sonucunda elde edilen tüm sayıların en küçük en büyük sonuç değerleri gösterilmiştir. Bu değerlere bakacak olursak küçük sayılarda sola bir çarpıklık olduğu görülmesine rağmen büyük sayılarda bu oran değişmiş ve simetrik bir dağılım gösterdiği söylenebilir. Bunun başlıca nedeni rastgele seçilen tüm sayı değerlerinin pozitif değerli olmasından kaynaklanmaktadır. Sayı adeti artıkça bu farklılık ortadan kalkmıştır. Tepe değerine bakıldığında ise genelde sıfır civarında seyrettiği söylenebilir. Öte yandan eğer dört işlem yapılan bir işlemde doğal olarak a/∞, $0/0$, $0 * \infty$ gibi sonsuz ötesi işlem sonuçlarının çıkması da beklenmelidir. Bu

durumda sonsuz ve tanımsız işlem sonuçlarının sayısı ise 6. ve 7. sütunlarda verilmektedir. İşlem sonucunda bulunan aralık içerisinde sonuçlar elde edilmesine rağmen bir işlem yarışmasında olduğu gibi 100 ile 1000 arasında değerlerle ilgilenildiği için bunların tüm kitle içerisindeki oranı ise 8. sütunda verilmiştir.

Tablo 10. Önerilen algoritmanın 100 denemede çıkan sonuç istatistikleri

n	N	En küçük sonuç	En büyük sonuç	Sonuç tepe değeri	Sonuz olan sonuç sayısı	Tanımsız olan sonuç sayısı	100-1000 arasına düşen sayı yüzdesi
2	6	-8	81	1	0	0	0
3	126	-77	504	1	60	0	1.174603
4	4572	-571	3456	0	4950	112	1.680665
5	274380	-4.50E+15	4.50E+15	0	363340	6528	2.638399

n: kullanılan sayı adeti; N: her denemede üretilen sonuç sayısı veya yapılan işlem sayısı

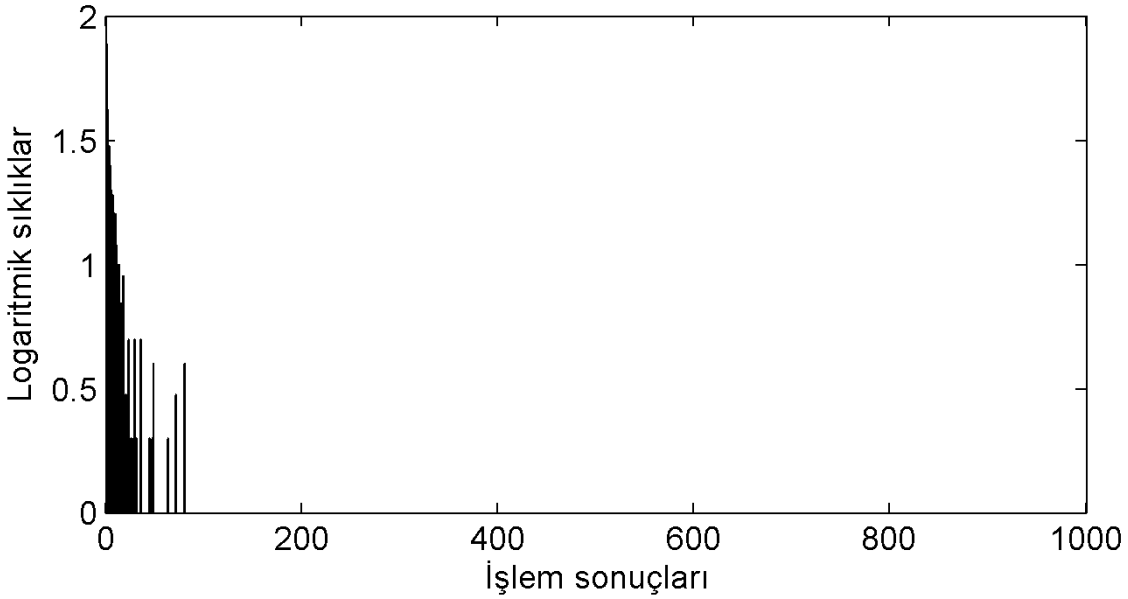
Tablo 10'da bulunan istatistiklere ek olarak bu sayıların 0 ile 1000 arasındaki histogramları aşağıdaki şekillerde (Şekil 30-Şekil 33) verilmiştir. Bu şekillerde 100 deneme sonucunda elde edilen sayıların histogramları bulunmuştur. Bulunan frekans değerleri 0 civarında yüksek bir oran gösterdiğinden devam eden değerlerdeki frekansların gösterimi engellediğinden bu frekansların 10 tabanında logaritması alınmıştır.

Şekil 30'da 2 elemanlı bir sayı kümesinin oluşturduğu sonuçların histogramı görülmektedir. Bu histogramda en büyük 9*9=81 olacağından daha büyük sayılarda frekans gözlenmemektedir.

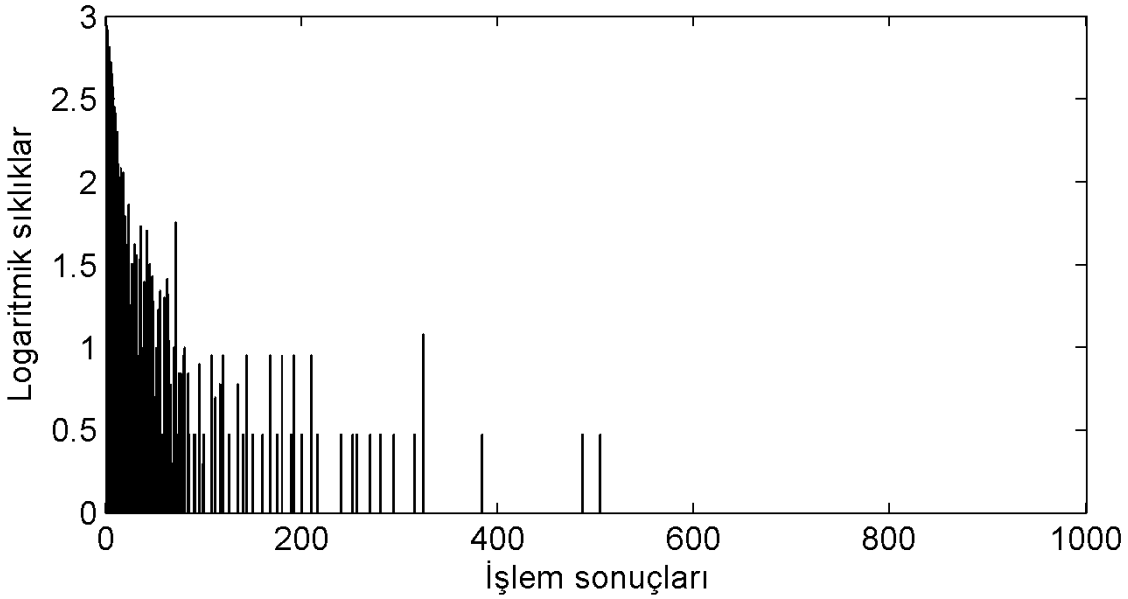
Şekil 31'de 3 elemanlı bir sayı kümesinin oluşturduğu sonuçların histogramı verilmiştir. Bu histogramda en büyük değerin 9*9*9=729 çıkması beklenir. Bu benzetimde ise en büyük 504 olarak bulunmuştur.

Şekil 32'de 4 elemanlı bir sayı kümesinin oluşturduğu sonuçların histogramı verilmiştir. Bu histogramda 1000 değerinin geçtiği söylenebilir. Ancak tüm frekans aralığının dolmadığı görülmektedir.

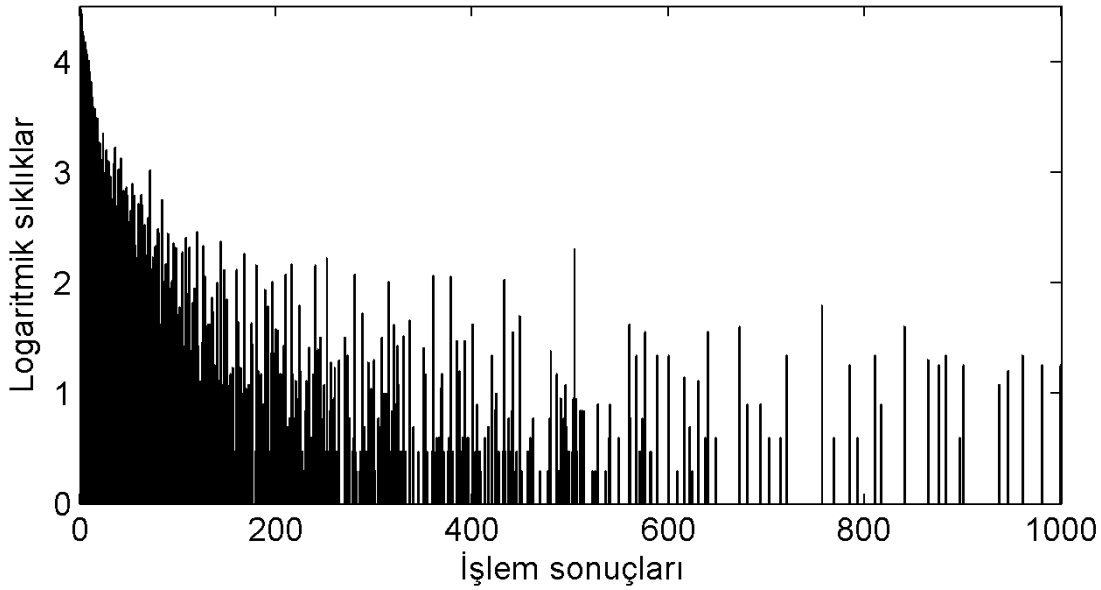
Şekil 33'te 5 elemanlı bir sayı kümesinin oluşturduğu sonuçların histogramı verilmiştir. Bu histogramda 1000 değerinin geçtiği görüldüğü gibi tüm frekans aralığının dolduğu söylenebilir.



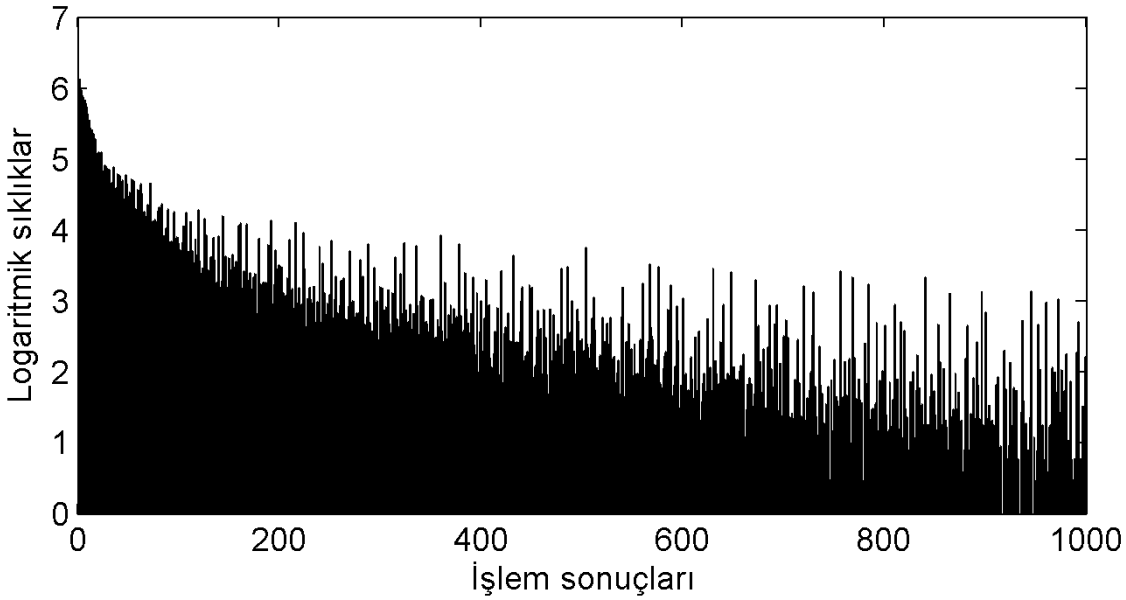
Şekil 30. Sayı adeti 2 olan işlem sonuçlarının logaritmik histogramı



Şekil 31. Sayı adeti 3 olan işlem sonuçlarının logaritmik histogramı



Şekil 32. Sayı adeti 4 olan işlem sonuçlarının logaritmik histogramı



Şekil 33. Sayı adeti 5 olan işlem sonuçlarının logaritmik histogramı

3.1. Geliştirilen Yazılımın Uygulamaları

TRT'de düzenlenen bir kelime-bir işlem yarışmasında ve İngiliz televizyonu kanal 4'de çözümü yarışmacılar tarafından bulunamayan bazı problemlerin çözümü aşağıdaki şekillerde verilmiştir.

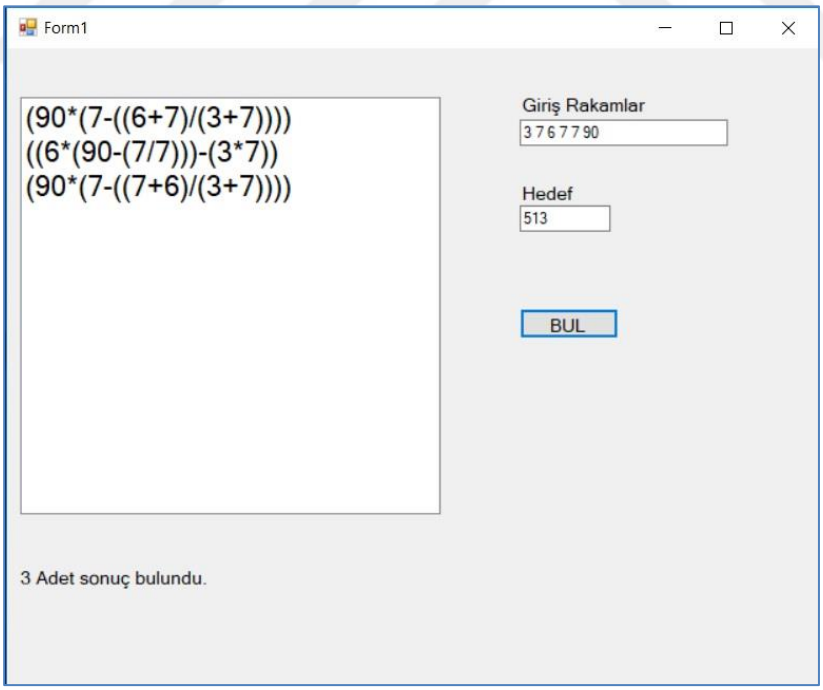
Şekil 34'te TRT tarafından yapılan yarışmada hedefi 513 verilen oyunun tam çözümü hiçbir yarışmacı tarafından bulunamamıştır. Geliştirilen yazılım ile toplam 3 tane tam çözümün olduğu görülmektedir.

Şekil 35'te TRT tarafından yapılan yarışmada hedefi 718 verilen oyunun tam çözümü hiçbir yarışmacı tarafından bulunamamıştır. Geliştirilen yazılım ile toplam 25 tane tam çözümün olduğu görülmektedir.

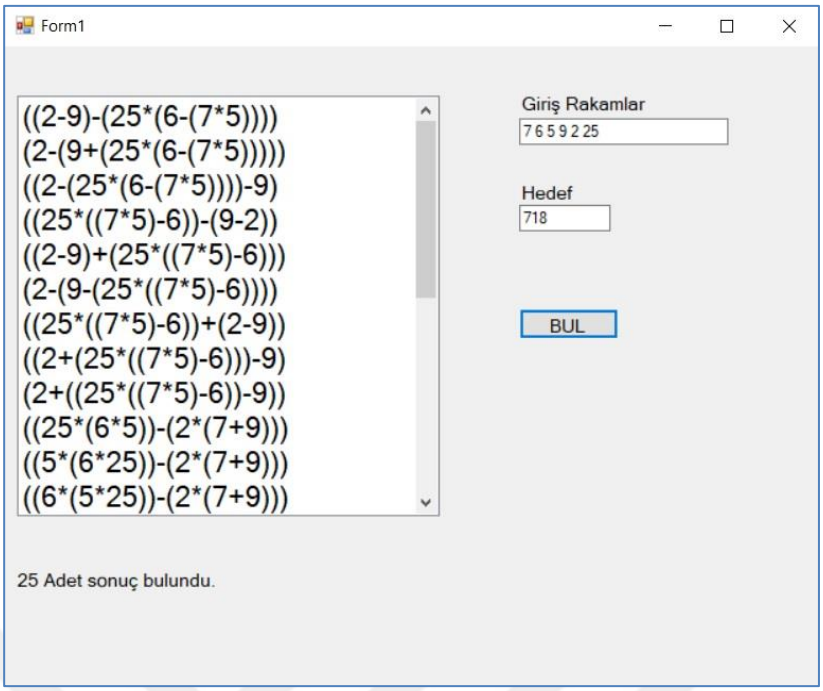
Şekil 36'da TRT tarafından yapılan yarışmada hedefi 876 verilen oyunun tam çözümü hiçbir yarışmacı tarafından bulunamamıştır. Geliştirilen yazılım ile toplam 78 tane tam çözümün olduğu görülmektedir.

Şekil 37'de TRT tarafından yapılan yarışmada hedefi 489 verilen oyunun tam çözümü hiçbir yarışmacı tarafından bulunamamıştır. Geliştirilen yazılım ile toplam 1 tane tam çözümün olduğu görülmektedir.

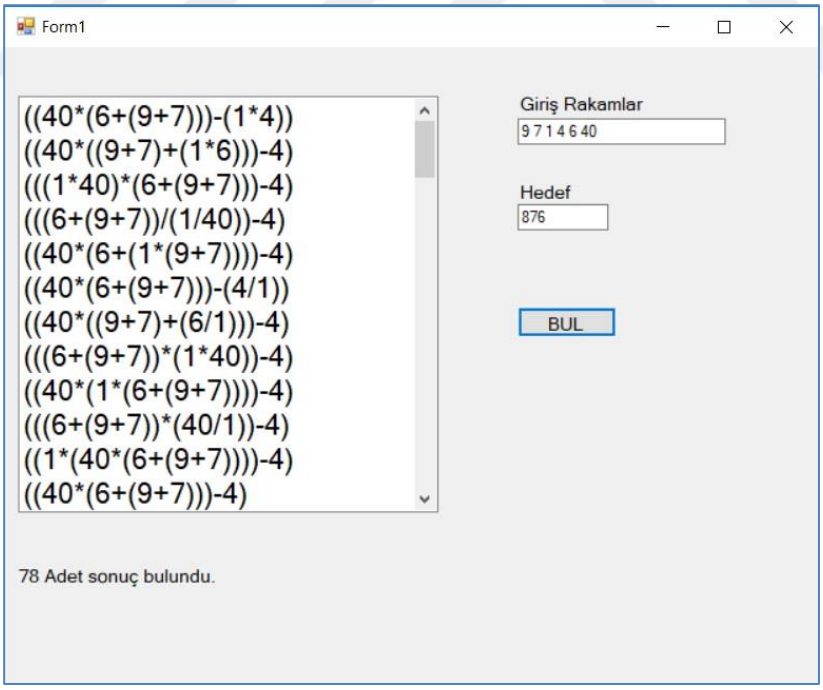
Şekil 38'de İngiliz televizyonu Kanal 4 tarafından yapılan yarışmada hedefi 770 verilen oyunun tam çözümü hiçbir yarışmacı tarafından bulunamamıştır. Geliştirilen yazılım ile toplam 241 tane tam çözümün olduğu görülmektedir.



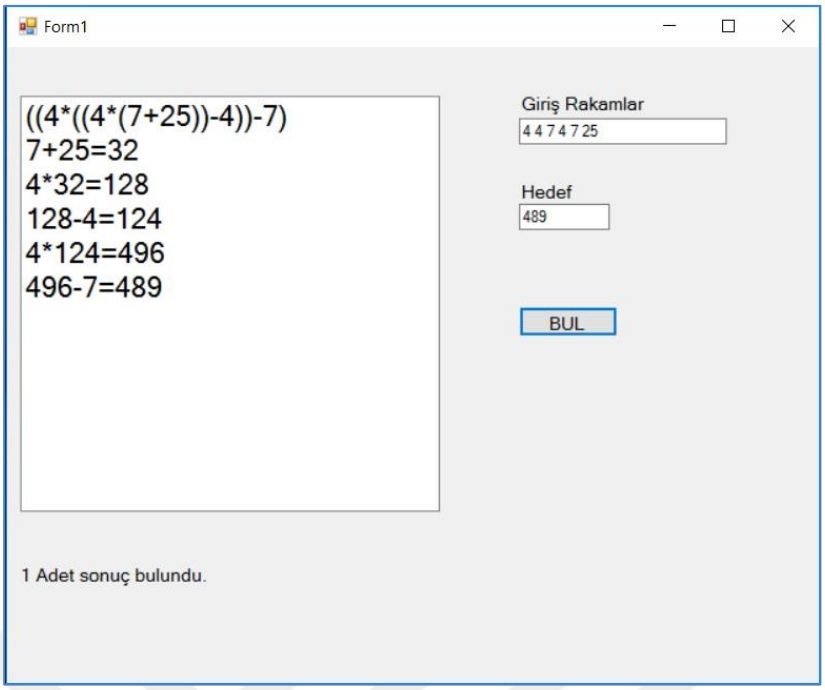
Şekil 34. TRT'de düzenlenen ve hedefi 513 olan problemin çözümü



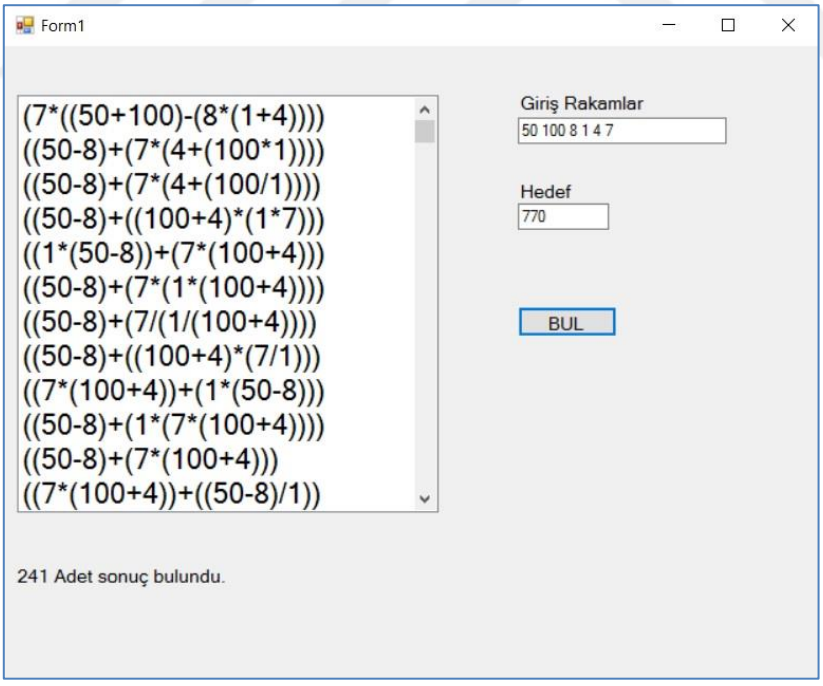
Şekil 35. TRT'de düzenlenen ve hedefi 513 olan problemin çözümü



Şekil 36. TRT'de düzenlenen ve hedefi 513 olan problemin çözümü



Şekil 37. TRT'de düzenlenen ve hedefi 489 olan problemin çözümü



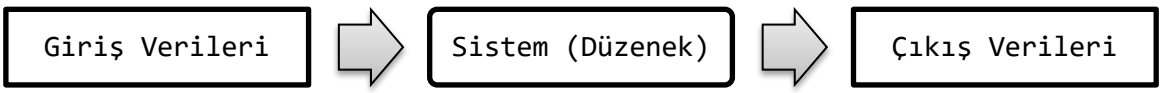
Şekil 38. Kanal 4'te düzenlenen ve hedefi 770 olan problemin çözümü

4. ÖNERİLER

Veri yapıları genelde tek bir kümenin alt kümlerini veya bir problemin olası tüm çözümlerini bularak en iyi sonuca ulaşmaya çalışır. Bu çalışmada geliştirilen Tip-2 ağaç yapısı ise birden çok kümenin kesişimini temel aldıkları için çok boyutlu veri yapısı problemlerinin çözümü için kullanılabilir. Örnek olarak, bir "Survivor" yarışmasında iki gurup arasında ikili bir yarışma yapıldığında bir çok etap için kaç farklı biçiminde eşleştirme yapılabilir gibi sorulara için tip-2 ağaç yapısı kullanılabilir.

Dört işlem problemi, belli bir sayıda verilen tam sayılar yardımıyla gerçekleştirilmektedir. Problemler gerçel sayılar veya karmaşık sayılar uzayına genişletilerek yapılabilir.

Bu çalışmada sadece dört işlem operatörü kullanılarak bir kombinasyon problemi yapılmıştır. Ancak farklı operatörlerinden işleme katılması söz konusu olduğunda yaklaşım bu probleme göre genişletilebilir. Farklı operatörlerin yanı sıra farklı fonksiyonlarda ağaç yapısına dahil edilirse bu durumda fonksiyonel programlama kapsamında bir probleme dönüşecek olan çalışma sayesinde ters mühendislik çalışmaları da çözülebilir. Ters mühendislik problemini Şekil 39'daki gibi gösterirsek, sistem yapısı ve çıkış verileri bilinirse giriş verilerini, sistem yapısı ve giriş verileri bilinirse çıkış verileri kestirilebilir. Ayrıca giriş ve çıkış verileri bilindiğinde sistemi bulmak için sistem modelinin bilinmesi gerekir. Sistem modeli bilindiğinde sadece katsayılar için bir çözüm geliştirilebilir. Ancak sistem modeli de bilinmediğinde özellikle doğrusal olmayan bir sistem modeli olduğunda çözümü kestirebilmek için önerilen yaklaşım kullanılabilir.



Şekil 39. Ters mühendislik probleminde sistem (düzenek) yapısı

Problem en fazla 5 rakam kullanılarak benzetimleri yapılmıştır. Ancak daha güçlü bilgisayarlar yardımıyla daha fazla rakam veya sayı kullanılarak başka işlemler de yapılabilir.

Hedef sayı verildiğinde hedefe ulaşan kısa çözümleri bulmak için sezgisel yaklaşımlar geliştirilebilir. Öte taraftan sadece tam sayılı sonuçlar ile ilgilenildiğinden ondalıklı sayılar elenebilir.

Oyun kuralları gereği seçilen tüm sayılar pozitif değerli seçilerek işlem yapılmaktadır. Oysa negatif değerli sayılar ile karmaşık sayılar içinde benzer yaklaşımlar geliştirilebilir.

5. KAYNAKLAR

Akdeniz, F., 2016. Olasılık ve İstatistik, Akademisyen Kitabevi, Ankara.

Alliot, J.-M., 2015. The (Final) countdown, CoRR, abs/1502.05450.

Colton, S., 2014. Countdown numbers game: Solved, analysed, extended, Proceedings of the AISB symposium on AI and Games.

Defays, D., 1990. Numbo: A study in cognition and recognition, CCAI(J. Integrated Study Artificial Intelligence Cognitive Sci. Appl. Epistemol.), 7, 2, 217-243.

Dijkstra, E. W., 1959. A note on two problems in connexion with graphs, Numerische mathematik, 1, 1, 269-271.

Fischler, M. A. ve Bolles, R. C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM, 24, 6, 381-395.

Ginsberg, M., 2012. Essentials of artificial intelligence, Newnes.

Hines, W. W., Montgomery, D. C., Goldman, D. M. ve Borror, C. M., 2008. Probability and statistics in engineering, John Wiley & Sons.

Hutton, G., 2002. The countdown problem, Journal of Functional Programming, 12, 06, 609-616.

Jones, M. T., 2015. Artificial Intelligence: A Systems Approach: A Systems Approach, Jones & Bartlett Learning.

Mogos, A.-H. ve Florea, A. M., 2008. Solving the Countdown Problem Using Automatic Composition of Semantic Web Services, Symbolic and Numeric Algorithms for Scientific Computing, 2008. SYNASC'08. 10th International Symposium on, 310-315.

Montgomery, D. C. ve Runger, G. C., 2010. Applied statistics and probability for engineers, John Wiley & Sons.

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

Nilsson, N. J., 2014. Principles of artificial intelligence, Morgan Kaufmann.

Russell, S. J. ve Norvig, P., 2010. Artificial Intelligence (A Modern Approach), Prentice Hall.

Shapiro, S. C., 1992. Encyclopedia of Artificial Intelligence, John Wiley & Sons, Inc..

Demir, M., http://www.kodumundunyasi.net/yapay-zeka/yapay-zeka-arama-ile-problem-cozme.html Yapay Zeka: Arama ile Problem Çözme. 27 Aralık 2016.

ÖZGEÇMİŞ

Elçin AĞAYEV, 14 Ağustos 1993 tarihinde Bakü'de doğdu. İlk öğrenimini 2010'da 158 nolu orta mektebinde tamamladı. Aynı yıl Azerbaycan Teknik Üniversitesi, Aftomatika ve Komputer Texnikası Fakültesi, Komputer Elmleri Bölümüne yerleşti ve 2014 yılında bu bölümden mezun oldu. 2014 yılında Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstatistik ve Bilgisayar Bilimleri Anabilim dalında tezli yüksek lisans programına başladı. Uluslararası sempozyumda sunulmuş bir bildirisi bulunmaktadır.