JÜRİ VE ENSTİTÜ ONAYI

Uğur ÇINAR'ın **Sezgisel Metodlar ile Planlama Problemlerinin Çözümünde Aşırı Bilgilendirilebilir Sistemler Yaklaşımı** başlıklı **Bilgisayar Mühendisliği** Anabilim Dalı **Bilişim** Programındaki, Yüksek Lisans tezi 21.08.2003 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı-Soyadı	İmza
Üye (Tez Danışmanı)	**: Doç.Dr.AHMET BABANLI**	
Üye	**: Prof.Dr.YAŞAR HOŞCAN**	
Üye	**: Yrd.Doç.Dr.YUSUF OYSAL**	

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 27.08.2003.. **tarih ve** 27/3.... **sayılı kararıyla onaylanmıştır.**

İnstitü Müdürü
Prof.Dr. Orhan ÖZER
Fen Bilimleri Enstitüsü
Müdürü

ÖZET

Yüksek Lisans Tezi

SEZGİSEL METODLAR İLE PLANLAMA PROBLEMLERİNİN ÇÖZÜMÜNDE AŞIRI BİLGİLENDİRİLEBİLİR SİSTEMLER YAKLAŞIMI

UĞUR ÇINAR

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Bilişim Yüksek Lisans Programı

Danışman: Doç. Dr. Ahmet BABANLI
2003, 99 sayfa

Bu tezde çözümünde Yapay Zeka metotlarının kullanılması kaçınılmaz olan planlama problemleri ilgi alanında; kategori, nitelik, ve karmaşıklık düzeyi önemli olmaksızın her türlü Yapay Zeka probleminin çözümüne yönelik olarak kullanılabilecek Aşırı Bilgilendirilebilir bir Sistem ele alınmıştır. Yukarıda bahsi geçen sistemin yazılım uygulaması Visual Basic programlama dili kullanılarak gerçekleştirilmiştir. Yazılımın tasarımı, çalışma prensipleri, koda yönelik algoritmik yapıları ile kullanım yer ve yöntemleri açıklanmıştır. Örnek olarak 8 Yap-Boz'u problemi üzerinde sistemin çalıştırılması gösterilmiş, elde edilen sonuçların yorumlanması yapılmıştır.

Anahtar Kelimeler: Aşırı Bilgilendirilebilir Sistem, Yapay Zeka Problemleri, Planlama Problemleri, Nesne Tabanlı Programlama, 8 Yap-Boz'u.

ÖNSÖZ

Yapay Zeka konseptindeki ilk araştırma ve bulguların bilgisayarın keşfinden çok daha öncesine dayandığı bilinmektedir. Bu alanda insana heyecan veren en önemli gelişmeler elbette ki bilgisayarın hayatımıza girişiyle başlamış, araştırmacıların hayalleri ile düşünme prosesinin bilgisayarda modellenmesinin zorluk ve kısıtlamaları arasında gidip gelen dalgalı bir seyir izlemiştir. Sahip olduğumuz bilgisayar sistemlerinin işleme mekanizmasının bir sonucu olsa gerek ki, insan zekasını bir bütün olarak modellemeyi ele alan yaklaşımlar 1960'lı yıllardan itibaren gelecekte üretilecek farklı yapıdaki (tümüyle paralel çalışan) bilgisayarlar ile tekrar ele alınmak üzere ertelenmiş, araştırmalar belirli problemlerin çözümüne olanak sağlayacak sistemler üzerine odaklanmıştır. Bu gibi özerk sistemler üzerindeki gelişmeler aslında öylesine ümit vaat edicidir ki bazı konularda habersiz insanlar karşılarındaki muhatabın insan olduğundan şüphe dahi duymayabilirler. Bu tezde açıklanan Aşırı Bilgilendirilebilir Sistemler'in bir genel çözüm önerisi olmamakla beraber aslında genel çözüme ulaşma yolunda katmanlar halinde kaskatlanarak önemli bir yapı taşı olacağı değerlendirilmektedir.

Yapay Zeka nitelikleri taşıyan sistemlerin tasarımının, bilgisayar programcılığının yanında bu alana yönelik ayrı bir uzmanlık gerektiriyor olması, konunun ilgi alanına giren problemler kümesinin muazzam genişliği ile birleştiğinde gelişmenin olabileceğinden daha yavaş ilerleyişinin de sebeplerinden bir kısmı açıklanmış olmaktadır. Aşırı Bilgilendirilebilir Sistemlerin alana sunacağı bir diğer faydanın da çözüm projeleri geliştirme işinin daha geniş bir toplum kesimine yayılmasını sağlayabilecek olmasıdır. Bu ifadede yatan temel anlam Aşırı Bilgilendirilebilir Sistemler aracılığıyla bilgisayar programcılığı niteliğinden yoksun olan insanların da Yapay Zeka problemlerine çözüm projeleri üretebileceklerinin değerlendirilmesidir. Kaldı ki nihai amaç bir çözüm projesinin yine bir üst sistem tarafından üretilebilmesini sağlamaktır (Aşırı Bilgilendirilebilir Sistemlerin kaskatlanması). Ancak kaskatlanabilirlik ve yöntemleri ile ilgili konular bu tezin kapsamı dışında tutulmuştur.

Aşırı Bilgilendirilebilir Sistemler adı altında açıklanacak olan tasarımın planlama problemlerinin çözümüne yönelik olarak geliştirilmiş olmasının nedeni sadece bir başlangıç niteliği taşımasındandır. Bu tezin muhteviyatını inceledikçe okurun aklında birçok yeni fikir oluşabilecektir ki aslında bunun da konunun geliştirilebilirliği ile ilgili ilk ip uçlarını verdiği değerlendirilmektedir.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER ve KISALTMALAR DİZİNİ

ABS : Aşırı Bilgilendirilebilir Sistem

VT : Veritabanı

jpp : J-Plan Projesi dosya uzantısı

jpl : Eleman Kütüphanesi dosya uzantısı

GS : Tahmin edilen toplam masrafın min. edilmesi ile arama(Greedy Search)

A* : Tahmin edilen toplam yolun minimize edilmesi ile arama

IDA* : Ötelemeli Derinlikli A* araması

SMA* : Basitleştirilmiş Hafıza Bağımlı A* araması

Alt ve üst indisler

Y : Boyca Eksen

X : Ence Eksen

b : Dallanma faktörü

d : Çözüm derinliği

g : Toplam maliyet fonksiyonu

h : Sezgisel fonksiyon

1 GİRİŞ ve AMAÇ

Bu tezde planlama problemleri baz alınarak problemin tanımlanmasıyla ilgili her türlü verinin program kodu olarak değil de sonradan veri niteliğinde kısmen bulanık olarak tanımlanabilmesine ve bu yöntemle Yapay Zeka(AI) problemlerinin bilgisayara çözdürülebilmesine olanak tanıyacak Aşırı Bilgilendirilebilir bir sistemin tasarımlanabileceği savunulmaktadır.

Konuya yönelik birincil tasarım çalışmaları tamamlanmış olup her türlü planlama probleminin yapısal olarak tanımlanma ve çözüm ihtiyaçlarına cevap verebileceği değerlendirilmektedir. Bu yüzden hazırlanan yazılım; içine konulduğu kabın şeklini alabilecek kadar sıvı, kaptan çıkarıldığında ise şeklini muhafaza edebilecek kadar katı olan "Jöle" kıvamından esinlenilerek "J-PLAN" olarak adlandırılmıştır. J-PLAN kullanıcısına; planlama görevlerinde kullanılabilecek Uzman Sistemleri, kod düzeyinde bir yazılımın hazırlanabilmesi için gerekli bilgilerden ve hatta Yapay Zeka programcılığının detaylarından uzak kalarak kısa sürede hazırlayabilme imkanı tanımaktadır.

Aşırı Bilgilendirilebilir Sistemler(ABS) kavramının çıkış noktası ve ilk örneği olarak bu tezde sunulacak olan J-PLAN, temelde veri niteliğindeki bilgileri çalıştırılabilir bir arama algoritmasının girdilerine dönüştürme işlevini yürütmektedir. Henüz akademik bir nitelik taşıyan sistemin mantıksal yapısı, yazılım ve hatta donanım(bu göreve yönelik özel bir donanım) yönünden geliştirilebilmesini mümkün kılmaktadır. Ancak bu tezin kapsamı yukarıda ifade edilen hazır çalışmanın sunulması ile sınırlandırılmıştır.

Sistemin genelini tanımlamak üzere tezin sahibi tarafından kullanılan "Aşırı Bilgilendirilebilir" ifadesinin tartışmaya açık bir adlandırma olmakla beraber, düşünceyi iyi derecede yansıttığı değerlendirilmektedir. İlk bakışta bilgisayarlara zeki davranabilme yeteneği kazandırabilecek her türlü uygulamanın aslında yeterince bilgilendirilmiş bir sistem olduğu düşünülebilir. Dolayısıyla mevcut uygulama geliştirme ortamlarının kıyaslanmasına da değinilmiştir. Nesne tabanlı problem tanımlama arayüzünün verileri saklama biçimi veritabanı sistemleri ile benzerlikler göstermektedir. Ancak bir ABS ayrık bir veritabanı sistemi kullanmak yerine asıl amacı verilerin birbirleri ile tutarlılığını zorlamak ve onları kullanım

için uygun bir formata dönüştürmek olan bütünleşik bir veri depolama sistemi kullanır.

1.1 Planlama Problemlerine Genel Bakış

Plan yapmak hayatın olağan ve olağan dışı akışı içerisinde en iyiye ulaşmak için insanoğlunun geliştirmiş olduğu en eski ve bilinen en etkin sezgisel (heuristic) metoddur [1,2]. İyi hazırlanmış bir plan uygulamanın ara noktalarında karar verme sürecini büyük ölçüde kısaltır. [3-5]

Genellikle belirli bir amaç için bilinçli olarak yürütülen planlama faaliyeti, çoğu zaman da insan beyninin otomatik bir fonksiyonu olarak karşımıza çıkar. Yerini ve nasıl gidileceğini çok iyi bildiğiniz iş yerinize aracınızla giderken aslında farkında olmadan bir planı uygulamıyor musunuz ? Yada kaldırımda yürüyen bir çocuğun aniden yola fırlaması durumunda sizin değil, ayağınızın ortadaki pedala nasıl basacağına dair yapılmış hazır bir planı yok mu acaba ?[6]

Yaşadığımız gerçek dünyada kalırsak, olayların zaman içindeki akışını bir algoritma ile tam olarak hesaplamak mümkün olmayabilir[7], mükemmel planımız "eğer ... ise ... yok eğer ... ise ... yok eğer....." ifadelerinden oluşmak zorunda kalırdı. Oysa probleme yönelik bir mini dünyada sebep-sonuç ilişkisi ile akışı kontrol edebilir ve mükemmel planda "eğer" lere ihtiyacımız kalmazdı. Böyle bir mini dünyada hazırladığımız mükemmel planımızı(planı hazırlamak için yeterli zamanımız varsa) gerçek dünyada uygularken (bizi zeki olmaya zorlayan rakibimiz tarafından) beklemediğimiz bir akış yoluna zorlanırsak yapmamız gereken tek şey yeni bir plan olurdu. Bu yeni plan eskisinden daha kolay olacaktır çünkü biz ilk planımızı hep en kötüye göre(en zeki rakibe göre) yapmıştık. Yukarıdaki cümleleri özetlersek gerçek dünyada da mükemmel plan vardır ancak hazırlanması için yeterli zamanı bulmak yada sonuna kadar uygulamak mümkün olmayabilir.

Eğer bir zaman kısıtı olmasaydı Yapay Zeka ilgi alanına giren tüm problemlerin birer planlama problemi olduğu söylenebilirdi. Ancak planlama bir anlamda sonucu ön görmedir ki, herhangi bir sonucun oluşumundan önce yapılmış olması gerekir. Öyleyse planlama problemleri, Yapay Zeka problemleri kümesi

içinde sürecin başlamasından önce arzu edilen kalitede bir planın yapılabilmesi için gerekli zamanın olduğu problemler alt kümesi olarak tanımlanabilir.

Planlama problemleri farklı karmaşıklık düzeylerinde olabilirler ancak daha da önemlisi problemin yapısının ve çözümden beklenen niteliklerin aynı problem için kısa zamanlarda değişikliklere uğrayabilmesidir. Örüntü tanımak üzere geliştirilmiş ve bir firmada çalışan insanları tanıyan bir uzman sistemi ele alırsak işe yeni başlayan bir uzaylının resmini veritabanına yerleştirmek sistemin ana yazılımında değişikliğe ihtiyaç doğurmayan, kolay bir çözüm olabilirdi. Oysa aynı şirketin çalışanlarının vardiya planlamasını yapan bir uzman sistem için şirkete yeni atanan yöneticinin prensipleri uzman sistemin kod seviyesinde yeniden düzenlenmesini gerektirirdi.

Planlama problemleri genellikle geçmişe bağımlı kümülatif olarak değişen girdiler ihtiva ederler. Asıl amaç planı yapmak değil onu bir bilgisayar sistemine yaptırmak olduğuna göre, problemin tanımlanması yada tanımın güncellenmesi için geçen süre de tekrarlayan planlama ihtiyaçları için "gerekli zaman" içinden koparılan bir dilimdir. [8]

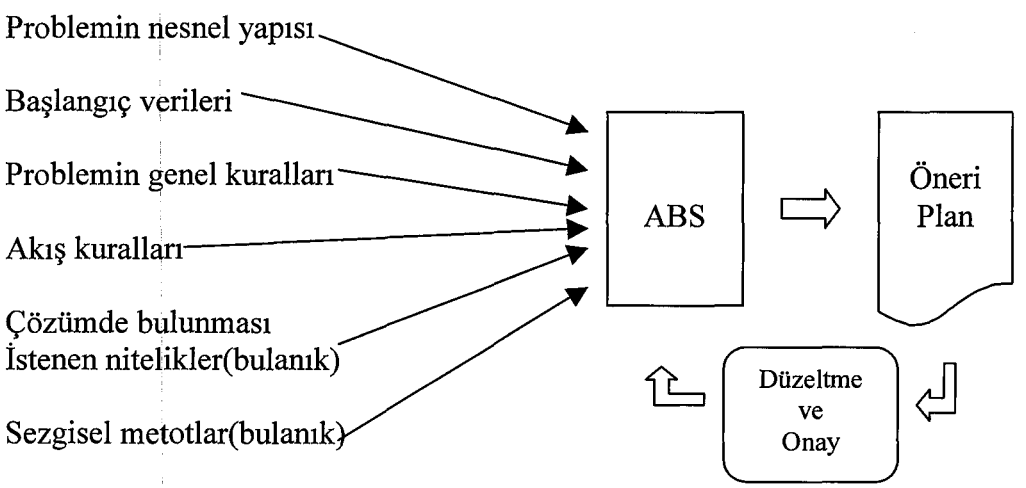
Sonuçta planlama problemleri için her türlü değişikliğe kod seviyesine inmeden uyarlanabilecek, özel bir uzman sisteme daima ihtiyaç vardır.

1.2 Aşırı Bilgilendirilebilir Sistem'in Tanımı

Aşırı Bilgilendirilebilir Sistem deterministik nitelik taşımayan, çözümü için bir arama algoritmasının kullanılmasını gerekli kılan problem türleri için özel olarak geliştirilmiş, problemin doğasından başlayarak olay akışının ve çözümün kurallarının, çözümün yada çözüm kümesinin net veya bulanık özelliklerinin, çözümün makul zaman ve hafıza harcaması ile bulunabilmesi için gerekli sezgisel metotların bünyesinde tanımlanabildiği ve nihai olarak yapılan tanımlamalardan otomatik olarak oluşturulan bir nesnenin (bu nesne problemin tüm tanımlamalarını özel bir formatta ihtiva eder) çözüm için işletilebilmesini sağlayan bir proje geliştirme ortamıdır.

ABS aynı zamanda tanımlama verilerinin tutarlılığını etkinlikle zorlar. Bu sayede kullanıcının yapması muhtemel hatalara karşı sistem bütünlüğünü korur. ABS'in çıktıları bir Öneri Plan niteliğindedir. Öneri plan uygulamaya

geçirilmeden önce uzman kişi tarafından onaylanmalıdır. Onaylama aşamasında plan üzerinde değişikliklerin yapılabilmesi ve değişen planın etkinlik, kurallara uygunluk ve hatta kriter olarak tanımlanmışsa ihtiva ettiği risk açısından otomatik olarak değerlendirilmesi beklenir. Yine bir ABS'in yeni planları, eski bir planın gerçekleşmiş uygulamalarından otomatik olarak doğan, kümülatif yada güncellenmiş veriler üzerinde gerçekleştirebilmesi beklenir. Bu sayede A.B.S bir planın sadece yapılmasını değil aynı zamanda uygulanması aşamalarını da denetleyebilen topyekün bir komuta kontrol sistemi olarak yapılandırılabilir.



Şekil 1.1 Aşırı Bilgilendirilebilir Sistem

1.2.1 Üst düzey programlama dilleri ABS kabul edilebilir mi ?

ABS tanımından yola çıkılarak aslında üst düzey programa dillerinin de birer ABS olduğu iddia edilebilir. Ancak bir ABS kendi görevlerini yapmak için özelleşmiştir. ABS Yapay Zeka problemlerinin kolaylıkla tanımlanabilmesini ve çözdürülebilmesini sağlayan hazır nesneler ile arama algoritmasının temel yapısını bünyesinde barındırır.

Program kodu yazmak da ayırt edici bir özelliktir. LISP ve PROLOG gibi Yapay Zeka uygulamalarında sıklıkla kullanılan mantıksal programlama dillerinde

program kodu temel bir yaklaşım iken ABS'de tanımlamalar görsel olarak yapılmaktadır.

Kısaca ifade etmek gerekirse eğer üst düzey programlama dilleri bu kategori içine alınacaksa öncelikle bir bilgisayarın donanım yapısı alınmalıdır(çünkü tanıma daha iyi uyum sağlar) yada diğer bir bakış açısıyla; bir mikroişlemcinin makine dili ne kadar üst düzey programlama dili olarak kabul edilebilirse bir üst düzey programlama dili de o kadar A.B.S olarak kabul edilebilir.

1.2.2 ABS'lerin veritabanı sistemleri ile kıyaslanması

ABS'lerin veritabanı(VT) sistemleri ile kıyaslanması hem içerisindeki veri alanlarının tanımlanma biçiminin VT sistemleri ile benzerlikler göstermesi hem de ABS içinde bir problemin tanımlanma biçimine ilişkin güzel ipuçları vermesi bakımından oldukça faydalı olacaktır.

Eğer bir eşleştirme yapılacaksa aşağıdaki çizelgenin anlamlı olacağı değerlendirilmektedir. Bu çizelgede bir VT sisteminin ve ABS'in kullanıcıya sunduğu araç ve kolaylıklar anlamsal benzerlikleri açısından yanyana getirilmiştir. VT sistemlerinde anlamı bilindiği kabul edilen terimlerin bir ABS içinde ne işe yaradığı açıklama alanına özetlenmiş, ABS için Sekiz Taşlı Yapboz problemi (8 Yap-Boz'u) üzerinde örnekler verilmiştir.

Çizelge 1.1 Veritabanı sistemleri ile kıyaslama

VT sistemi karşılığı	ABS karşılığı	A.B.S Örnekleri	Açıklama
Tablo Adı	Küme Adı	Konumlar Taşlar Çözümler	Tanımlanan nesnenin kategorik adı(çoğul)
Tablo tanımı	Tanım (Nesne tanımı)	Konum Taş Çözüm	Tanımlanan nesnenin kendisi (Adlandırmada Küme adı tekil kullanılır)
Alan	Özellik	X Endeksi Konumu Hamle Sayısı	Tanımın özellik nesnesi (Tanımdan bir nesne türetildiğinde nesnenin bu özelliğine değer atanabilir)
Alan değer tipi	Değer tipi	Sayı	Özelliğin değer tipini tanımlar
"Text" değer tipi	"Yazı"	"FBE123"	Karekter dizileri için kullanılır

Çizelge 1.1 Veritabanı sistemleri ile kıyaslama (devam)

VT sistemi karşılığı	ABS karşılığı	A.B.S Örnekleri	Açıklama
"Sayı" değer tipi (Real Number)	"Sayı"	4513.45	Gerçek sayı tipi veriler için kullanılır (tamsayılar için de kullanılabilir)
"Tarih" değer tipi	"Tarih"	27.01.1971	Tarih verileri için kullanılır. Bünyesinde zaman dilimi içermez
-	"Zaman"	125:50:34 (125 saat 50 dakika 34 saniye)	Zaman verileri için kullanılır. (Saat hanesi 23 den büyük değer de alabilir)
"Mantık" değer tipi (Boolean)	"Evet/Hayır"	Evet Hayır	Mantıksal Evet ve Hayır verileri için kullanılır
Sadece Dömen olarak hazırlanmış bir tablodan alınabilen verilerden oluşan değer tipi	"Tanım"	Konumlar Taşlar	Nesne değer tipli veriler için kullanılır. Değer tipinin hangi nesne kategorisi olarak atanacağı Küme belirtilerek yapılır. Bu küme elemanları ile bir dömen görevi üstlendiği gibi veri olarak atanan elemana da bir nesne olarak Özellik üzerinden erişilebilir
Anahtar	Anahtar	Bir Özellik için "Evet" yada "Hayır"	Anahtar kavramının ABS için-de çok özel bir anlamı vardır. Bu anlam bir VT sistemindeki anlamıdan farklılıklar gösterir. Tasarım bölümü içinde detayları açıklanacaktır.
Kayıt	Eleman	Konum11 Taş34 Çözüm1	Tanımdan (nesne tanımı) Küme si içinde türetilen bir gerçek nesnedir. Eleman'ın özelliklerine değer tipine uygun veriler atanabilir. Eleman aynı zamanda çözüm için işletilebilir bir nesnedir.

Çizelge 1.1 Veritabanı sistemleri ile kıyaslama (devam)

VT sistemi karşılığı	ABS karşılığı	A.B.S Örnekleri	Açıklama
-	Etki	"Hamle sayısını 1 artır" (Çözüm. Hamle Sayısı = Çözüm. Hamle Sayısı + 1)	ABS'e özel bir kolaylıktır. Problemin tanımlanmasında çözümün alt aşamalarının planın o anki özgün durumu üzerinde yapacağı etkinin kullanıcı tarafından tanımlanabilmesi için kullanılır. Otomatik plan üretiminde sistem tarafından değerlendirilir. ABS'in komuta kontrol rolün-de kullanılmasında uygulamadan doğan verilerin otomatik türetilmesi görevini üstlenir.
-	Seçim Kriteri	"Yanında boşluk olmayan taş oynanamaz"	Tanım Zincirinin Plan Grafına dönüştürülmesi esnasındaki kuralların bulanık olarak tanımlanabilmesini sağlar.
-	Kriter	"En az hamle ile çözümü bul"	Çözüm önerisi olarak otomatik oluşturulan planın taşıması istenen niteliklerin bulanık olarak tanımlanabilmesini sağlar. Ayrıca sezgisel metotlar Kriter aracılığı ile sisteme entegre edilebilir.
İlişki	Bağlı Küme	Oynanan Taş	Aslında bir grafı işaret ediyor olmasına rağmen problem tanımında ilişkiler Bağlı Küme ler aracılığı ile bir zincir olarak (Tanım Zinciri) tanımlanır.
Sorgu yürütme	Otomatik düşünme	-	Bir VT sistemi temel görevi olan uygun verileri seçme, düzenleme ve istenen formda kullanıcıya sunma görevini sorgular aracılığla yürütürken bir ABS temel görevi olan en uygun planı üretme işlemini "Otomatik Düşün" komutu ile gerçekleştirir

2 TASARIM

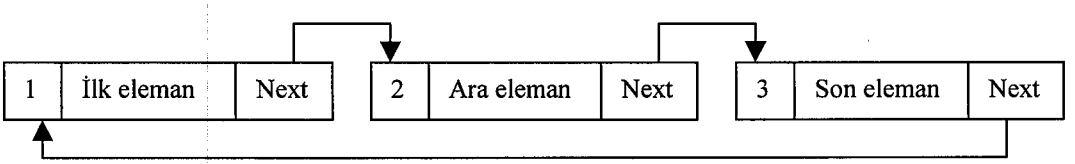
Bu teze konu olan ilk ABS'in tasarımı kendine özgü olup, bir ABS'den beklenen özellik ve fonksiyonaliteye odaklanmak kaydıyla elbette değişik tasarımların da yapılması mümkündür. Aşağıda açıklanacak olan tasarım bir hayata geçirme tarzıdır.

Tasarımın uygulamalarda daha performanslı sonuçlar üretebilmesi daha profesyonel programlama dillerinin kullanımı ile sağlanabilir. Ancak ilk aşamada kullanım kolaylığı ve okunabilirlik avantajları nedeni ile Visual Basic 6.0 programlama dili üzerinde yapılandırılmıştır. Dolayısıyla tasarım aşamaları açıklanırken programlama dilinin ne şekilde kullanıldığı değil mantıksal yapı üzerinde durulmaya çalışılmıştır. Yine de programlama dilinin kısıtlamaları nedeni ile bazı aşamalarda dolaylı yolların kullanılması zorunludur. Bu kısıtlamalar genellikle sistem alt yapısında dikkati çeker.

Tasarım aşamaları iki temel gurupta açıklanmıştır. Bunlar : Sistem alt yapısı ve sistemin temel bölümleri(problem tanımlama bölümü, yürütme bölümü, sonuç gösterim bölümü)dir.

2.1 Sistem Alt Yapısı

Problemin doğasından bağımsız bir sistem yaratılırken öncelikle çok elastik bir alt yapıya ihtiyaç vardır. Dolayısıyla doğrusal(linear) veri yapıları çoğunlukla yetersiz kalmaktadır. Zaten Yapay Zeka problemlerinin de doğrusal bir yapıda olmaları beklenemez. Yapılan tanımlamaların bir kayıt ortamında saklanabilmeleri gereksinimi de düşünülerek bu alt yapının bir Bağlı-Liste(Link-List) olması gerektiği değerlendirilmiştir. Bağlı listeler işaretçilerle(pointer) çalışmayı ve dinamik veri tanımlamalarını gerektirir oysa Visual Basic dinamik tanımlamalara kısıtlı, işaretçi kullanımına ise hiç destek vermez. Bu sorunu aşmak için kullanıcı tanımlı bir veri yapısı ve dolaylı bir adresleme yöntemi (dizi indisleri işaretçi olarak) kullanılmıştır. Bu yöntemin basit açıklaması aşağıdadır[9].



Şekil 2.1 Üç elemanlı bağlı-liste

Şekil 2.1 deki yapının oluşturulmasında kullanılan kod :

```
Type listeElemanıT
Next as long
Data as String
End type
```
Kullanıcı tanımlı veri yapısı

```
Dim listeElemanı() as listeElemanıT
```
Dinamik dizi değişkeni

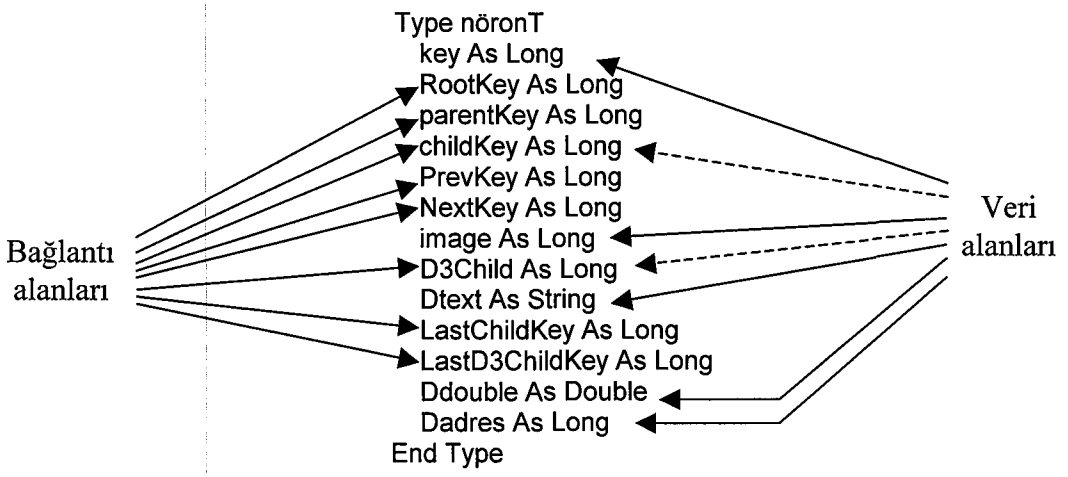
```
Redim preserve listeElemanı(1 to 3)
```
İlk üç Liste Elemanı

```
listeElemanı(1).Next = 2
listeElemanı(1).Data = "İlk eleman"

listeElemanı(2).Next = 3
listeElemanı(2).Data = "Ara eleman"

listeElemanı(3).Next = 1
listeElemanı(3).Data = "Son eleman"
```
Listenin bağlanması ve veri atamaları

```
Debug.Print listeElemanı(listeElemanı(listeElemanı(1).Next).Next).Data
```
ulaşım

(Yukarıdaki komut satırı Debug penceresine "Son eleman" yazdıracaktır.)

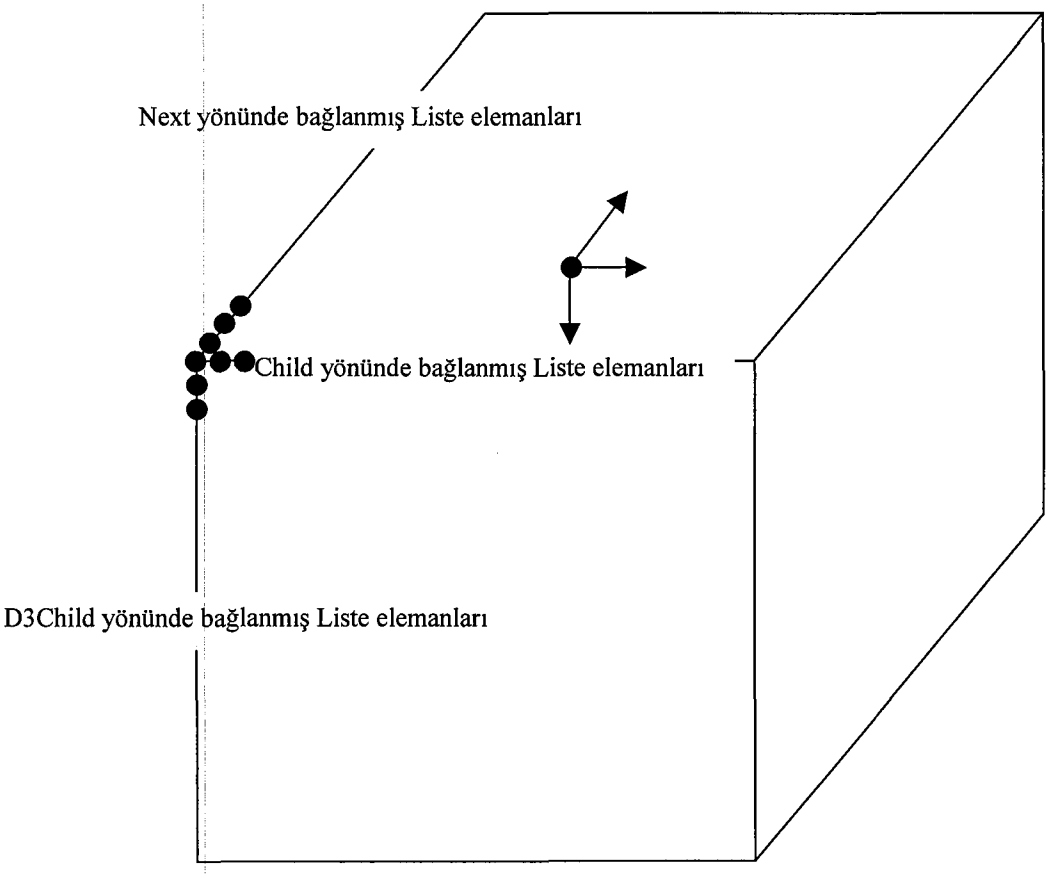
Bağlı-Liste yöntemi ile miktarı belli olmayan veri hafızada tutulabilir. Fakat örnekteki liste elemanının yapısı oldukça basittir. Bu yapının sistem içinde kullanımı için şekildeki gibi genişletilmesi gerekmektedir. Burada kullanılan "nöron" (nöronT) ifadesi sadece bir değişken ismi olup anlamı ile bağlantı kurulmaya çalışılamamalıdır.



Şekil 2.2 ABS'in bağlı-liste elemanı yapısı

ChildKey ve D3Child alanları hem işaretçi hem de veri alanı olarak kullanılabilirler. Ne maksatla kullanıldıkları ise Child ve D3Child yönündeki en son liste elemanına işaret eden LastChildKey ve LastD3ChildKey işaretçilerinin bir liste elemanı gösterip göstermediği ile belirlenebilir.

Şekil 2.2 de gösterilen bağlı-liste elemanı yapısını incelenirse değişik veri tiplerini saklayabilecek bir hafıza alanı yapısı olarak tasarımlandığı ve bağlantı alanlarının çok olması sayesinde lineer olmayan nesne tanımlamalarında ihtiyacı karşılayabilecek nitelikte olduğu görülecektir. Alt seviye ile bağlantıyı sağlayacak olan (Next,childKey ve D3Child) veri alanlarının birden fazla olması verinin üç-boyutlu bir mantıksal alana dağıtılabilmesini ve bu sayede yapının çok boyutlu graf şeklindeki tanımlamalarda kullanılabilmesini sağlar. Şekil 2.3 de gösterilen her üç yöndeki bir liste elemanına yine her üç yönde başka bağlı-liste elemanlarının bağlanabileceğine dikkat çekilmektedir.[9]



Şekil 2.3 Bağlantı uzayı yapısı

Bahsedilen dolaylı bağlı-liste yapısının iki önemli dezavantajı vardır. Bunlardan birincisi bağlı liste elemanlarının mantıksal olarak silinebilmesine karşın fiziksel olarak hafızada yer kaplamaya devam etmeleridir. Çünkü ulaşım lineer olmamasına karşın liste elemanlarının dizi değişken içinde doğrusal bir yapıda tutulmaktadır. Bir liste elemanının fiziksel olarak da silinebilmesi için dizi değişken içindeki o elemandan sonra gelen elemanların yukarıya taşınması gereklidir ki bu da zaman kaybına ve bağlantılarda karmaşıklığa yol açacaktır. Bu sorunun çözülmesi için silinen elemanların indislerini tutan bir tablo kullanılmıştır. Yeni liste elemanları öncelikle bu tablodan alınan yerlere yerleştirilir. Aşağıda tablonun yapısı ve tablodan boş alanın indisini döndüren fonksiyon gösterilmektedir.

```
Public freeKeys() As Long

Private Function freeKeyBul() As Long
  Dim cnt As Long
  cnt = UBound(freeKeys)
  If cnt Then
    freeKeyBul = freeKeys(cnt)
    ReDim Preserve freeKeys(cnt - 1)
  Else
    freeKeyBul = UBound(nöron) + 1
    ReDim Preserve nöron(freeKeyBul)
  End If
  nöronTR(freeKeyBul, 0) = freeKeyBul
End Function
```

Görüldüğü gibi eğer tabloda bir kayıt varsa bu kaydın indisi döndürülerek kayıt silinir. Eğer bir kayıt yoksa yeni liste elemanı için yeni hafıza alanı açılır ve bu yeni alanın indisi döndürülür. Tabloya kayıt ekleme işlemi ise bir liste elemanının silinmesi ile görevli yordam tarafından yürütülür.

İkinci dezavantaj ise ulaşım ile ilgilidir. Bir liste elemanına hem veri okuma hem de veri yazma işlemleri için başvurulabilir. Okuma ve yazma işlemleri ise beraberlerinde ilave işlemlerin de yapılmasını gerektirebilir. Bu işlemleri için ise liste elemanlarının indislerinin parametre olarak kullanılabildiği bir çift veri aktarma(transfer etme) fonksiyonu(property-function) kullanılmıştır.

```
Public Property Get nöronTR(key As Long, valNo As Long) As Variant
   On Error GoTo NöronNotExist
   With nöron(key)
    Select Case valNo
      Case 0
        nöronTR = .key
      Case 1
        nöronTR = .RootKey
      Case 2
        nöronTR = .parentKey
      Case 3
        nöronTR = .childKey
      Case 4
        nöronTR = .PrevKey
      Case 5
        nöronTR = .NextKey
      Case 6
        nöronTR = .image
      Case 7
        nöronTR = .D3Child
      Case 8
        nöronTR = .Dtext
      Case 9
        nöronTR = .LastChildKey
      Case 10
        nöronTR = .LastD3ChildKey
      Case 11
        nöronTR = .Ddouble
      Case 12
        nöronTR = .Dadres
    End Select
   End With
  Exit Property
NöronNotExist:
  Select Case valNo
    Case 8
      nöronTR = ""
    Case Else
      nöronTR = 0
  End Select
End Property
```

```
Public Property Let nöronTR(key As Long, valNo As Long, ByVal vNewValue As
                                                                    Variant)
  If key = 0 Then Exit Property
  Form_Ana.tbToolBar.Buttons(3).Enabled = True
  Form_Ana.mnuFileSave.Enabled = True
  With nöron(key)
   Select Case valNo
    Case 0
     If vNewValue Then
      .key = vNewValue
     Else
      cnt = UBound(freeKeys) + 1
      ReDim Preserve freeKeys(cnt)
      freeKeys(cnt) = .key
      .key = 0
      .RootKey = 0
      .parentKey = 0
      .childKey = 0
      .PrevKey = 0
      .NextKey = 0
      .image = 0
      .D3Child = 0
      .Dtext = ""
      .LastChildKey = 0
      .LastD3ChildKey = 0
      .Ddouble = 0
      .Dadres = 0
     End If
    Case 1
     .RootKey = vNewValue
    Case 2
     .parentKey = vNewValue
    Case 3
     .childKey = vNewValue
    Case 4
     .PrevKey = vNewValue
    Case 5
     .NextKey = vNewValue
    Case 6
     .image = vNewValue
    Case 7
     .D3Child = vNewValue
    Case 8
     .Dtext = vNewValue
    Case 9
     .LastChildKey = vNewValue
    Case 10
     .LastD3ChildKey = vNewValue
    Case 11
     .Ddouble = vNewValue
    Case 12
     .Dadres = vNewValue
   End Select
  End With
End Property
```

Yukarıdaki iki fonksiyon sırasıyla elemandan veri okuma ve elemana veri yazma işlemlerini yürütürler. Bir elemanın silinmesi yazma fonksiyonu içinde "key" alanı için "0" değeri girilmesi ile sağlanır. Bu iki fonksiyon aynı zamanda sistemi okunan/yazılan verinin tip uygunsuzluğu ve var olamayan liste elemanları üzerinde okuma/yazma işlemleri problemlerinden de soyutlar.

Yeni liste elemanları oluşturma işlemi yine bir fonksiyon aracılığı ile yapılmaktadır. Çünkü çoğu zaman liste elemanları teker teker oluşturulmaz. Bir sistem-tanımlı nesnenin yapısını oluşturan bir gurup liste elemanı aynı anda oluşturulur. Sistem içinde tanımlı olan bu nesneler ve yapıları ilerleyen bölümlerde detayları ile açıklanacaktır.

Alt yapı açıklanırken Bağlı-Listenin kayıt ortamına nasıl aktarılacağına da değinmek gerekir. Bir liste elemanının hafızada kapladığı yer, ihtiva ettiği karakter katarı(String) veri alanından dolayı değişebilir uzunluktadır. Kayıt yapısı ise dosyaya bu veri alanı en sona gelecek şekilde sıralı(sequential) olarak yazılır.[10] "key" veri alanında "0" değeri olan (mantıksal olarak silinmiş) liste elemanları kayıt işleminde dosyaya yazılmazlar. Hazırlanan dosya ".jpp" (J-Plan Projesi) dosya uzantısı ile ortama kaydedilir. Bir proje açılırken tüm liste elemanları aynı şekilde kayıt ortamından hafızaya alınır ve buradan kullanılır.

2.2 ABS'in Temel Bölümleri

ABS üç temel bölümde açıklanacaktır.

2.2.1 Tanımlama bölümü

Tanımlama bölümü ABS'in problem hakkında aşırı derecede bilgilendirilebildiği, kullanıcı ile en çok karşı karşıya olacak bölümüdür. Bu bölümde kullanılan rutinlerin büyük bölümü kullanıcı arayüzünün oluşturulması ile ilgilidir. Yapı ve doğasıyla birbirinden tamamen farklı olabilen problemlerin tanımlanabilmesi için sistemin kolay kullanılabilir ve son derece elastiki bir yapıda tasarımlanması gerekmektedir. Dolayısıyla bu noktada tanımlanacak sistem kullanım kurallarının belirlenmesi için uzun çalışmalar yürütülmüştür.

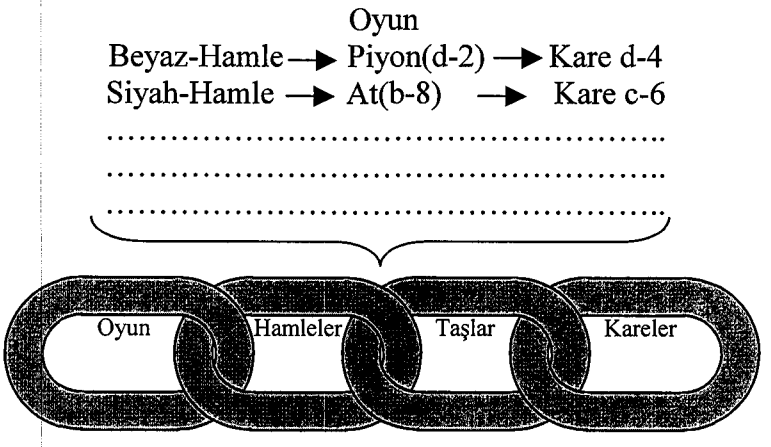
Nesneye dayalı tanımlamalar yapmak kullanıcının görsel kabiliyetlerini kullanması açısından tüm programcılık literatüründe kabul gören etkin bir

yöntemdir[11]. Bu yüzden problem tanımlama arayüzü Nesne tabanlı, görsel bir yapıda tasarımlanmıştır.

2.2.1.1 Problem tanımlama arayüzü

Problem tanımlama arayüzü nesneye dayalı tanımlamaların kolaylıkla yapılabilmesi için bir gurup hazır nesnenin kullanımını gerektirmektedir. Bu hazır nesneler sayesinde kullanıcı elastikiyetten ödün vermeden problemin doğasını sisteme aktarabilir.

Problem tanımlama arayüzünde temel mantık hazır nesneleri yapılandırarak problemin ana hatlarını yansıtan bir Tanım Zinciri'nin oluşturulmasıdır. Burada "nesne zinciri" değil "Tanım Zinciri" ifadesi kullanılmıştır çünkü zincir nesneler ile değil nesnelerin tanımları ile oluşturulur ve ana hipotez evrendeki tüm sistemlerin bir Tanım Zinciri ile ifade edilebileceğidir. Bu ifade biçimi bir kategorizasyon yaklaşımına işaret eder. Konuya satranç problemi üzerinde bir örnek verilecek olursa dört halkalı bir Tanım Zinciri yeterli olacaktır.



Şekil 2.4 Örnek Tanım Zinciri

Şekil 2.4 de görülen örnek Tanım Zinciri bir kez oluşturulduktan sonra işleme zincirdeki her bir tanım nesnesi için gerçek nesneleri türetmek ve kuralları tanımlamakla devam edilir.

Türet "Bu oyun" : Oyun

Türet "Beyaz-Hamle" : Hamle
Türet "Siyah-Hamle" : Hamle

Türet "Beyaz Şah" : Taş
Türet "Beyaz Vezir" : Taş
.....
.....

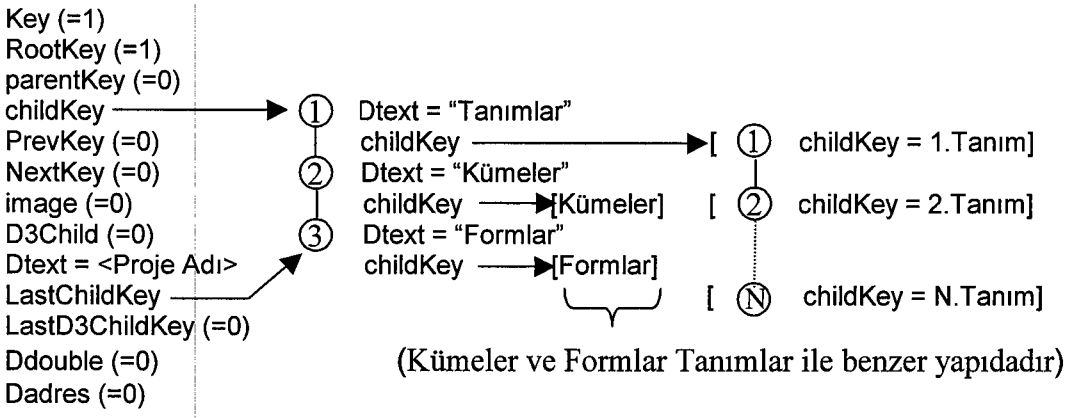
Türet "A-1 Karesi" : Kare
Türet "A-2 Karesi" : Kare
.....
.....

Kural-1 : Oyuna Beyaz-Hamle ile başlanır.
Kural-2 : Bir Beyaz-Hamle bir Siyah-Hamle sıra ile oynanır.
Kural-3 : Her hamlede mutlaka bir ve yalnızca bir Taş oynanır.
Kural-4 : Her Taş bulunduğu Kareden bir başka Kareye Oynanır.
.....
.....
.....
Kural-M : Siyah kurallara uyan her hamleyi(Taş-Kare ikilisi) yapabilir.
Kural-N : Beyaz oyunu kazanmasını sağlayacak hamleleri yapmalıdır.
(Bulanık kural)

Şimdi yukarıda örnekle açıklanan tanımlamaların yapılmasını sağlayacak temel ABS nesneleri açıklanacaktır. Daha önce belirtildiği gibi ABS'in temel nesneleri de yukarıda yapısı açıklanan liste elemanlarından oluşan bir bağlı-liste şeklindedir. Liste elemanlarının birçok bağlantı alanı bulunduran yapısı nedeni ile bir nesnenin yapısının gösterimi biraz karmaşık olabilir. Karmaşıklığı azaltmak maksadıyla sadece bağlantıları ve önemli verileri içeren alanlar gösterime dahil edilmiştir. Bağlantılar ok (→) işaretleri ile temsil edilmektedir. Bir bağlantı okunun ucundaki daire içine alınmış numaralar işaret edilen liste elemanlarını temsil etmekte fakat numara bu liste elemanının indisini değil sadece "Next" yönünde bağlanmış liste elemanlarının sıra numarasını göstermektedir. İndisler sadece bağlantıların yapılması için kullanıldığından indis numarasının ne olduğu gösterimde hiç önemli değildir. < > işaretleri arasında belirtilmiş veriler kullanıcı tarafından girilen zorunlu veriler, [ ] işaretleri arasında belirtilmiş veriler tanımlandıkça oluşturulan veriler, " " işaretleri arasında belirtilmiş veriler ise sistem tarafından yazılmış değişmez verilerdir.

Proje nesnesi

Bir projenin en temel verilerini, hazırlanmış Tanım, Küme, Form gibi nesneleri bir liste halinde tutan en temel nesnedir. Proje nesnesinin ilk bağlı liste elemanının indisi "1" dir ve projenin diğer tüm liste elemanları en üst noktada bu liste elemanına bağlıdır. Proje nesnesi Yeni bir proje oluşturulurken yaratılan yegane nesnedir. Sistem içinde gösterimi ve işlevleri yoktur. Oluşturulan tüm Tanım, Küme ve Form nesneleri bu nesneye bağlantılanırlar.



Şekil 2.5 Proje nesnesi mantıksal yapısı

Tanım nesnesi

Tanım nesnesi ABS içinde nesne tanımlanabilmesi görevini yürütür. Proje açıldığında görülebilir bir nesnedir. Tanım nesnesinin sistem tarafından tanımlanmış(hazır), kullanıcı tarafından değer atanabilen özellikleri aşağıdaki çizelgede gösterilmiştir.

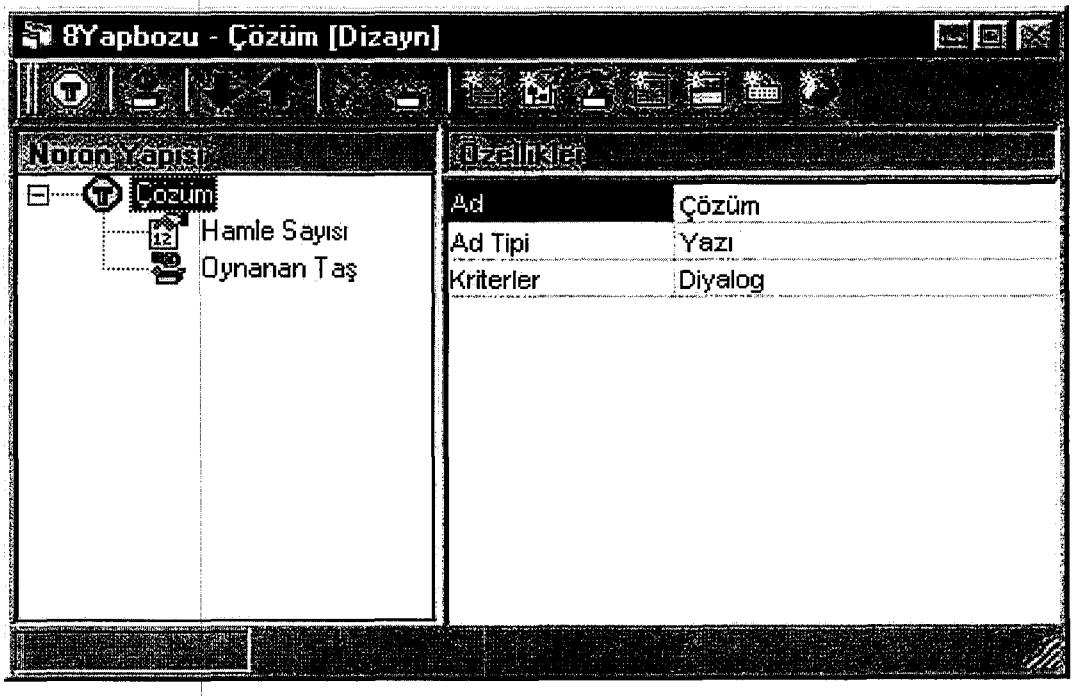
Çizelge 2.1 Tanım nesnesi hazır özellikleri

Hazır Özellik	Değer Tipi	Değer Ataması
Ad	<Karakter katarı>	Klavyeden giriş
Ad Tipi	<"Yazı" yada [Özelliğe işaretçi]>	Listeden Seçim
Kriterler	[Bağlı-Liste]	İletişim penceresi (dialog)
Etkiler	[Bağlı-Liste]	İletişim penceresi (dialog)

Ad özelliği Tanım nesnesinin adlandırılması için kullanılır. Her tanım nesnesinin diğer Tanım nesnelerinden farklı(unic) bir adı mutlaka olmalıdır. Ad

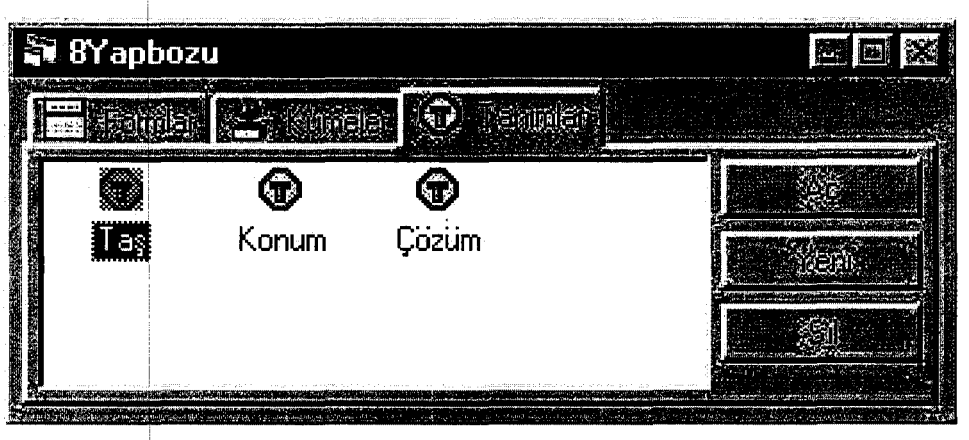
Tipi özelliği bir Tanım nesnesinden türetilen gerçek nesnelerin(Eleman) nasıl adlandırılacağını belirler. Değer olarak "Yazı" seçeneği yada Sayı, Zaman veya Tarih değer tipli kullanıcı tanımlı özelliklerden biri seçilebilir. "Yazı" seçeneği türetilen gerçek nesnelerin manuel olarak adlandırılmasını gerektirir. Diğer seçeneklerde ise gerçek nesneler otomatik olarak adlandırılırlar. ABS içinde bir Tanım'dan türetilen tüm gerçek nesnelere kalıtsal olarak aktarılmak üzere Tanım nesnesi için Kriterler ve Etkiler tanımlanabilir. Kriterler projelendirilen problemin kurallarının, bulanık beklentilerinin ve yine bulanık olarak çözüm için faydalı olabilecek sezgisel metotların tanımlanabilmesini sağlar. Etkiler ise çözümün alt aşamalarında Tanım nesnesinin sistemin durumu üzerinde yapacağı etkilerin tanımlanma yeridir. Tanım Zinciri içinde yer almayan Tanım nesnelerinin Kriterler ve Etkiler özellikleri yoktur. Tanım Zinciri içindeki Tanım nesnelerinin ise en alt halkasının Kriterler özelliği bulunmaz.

Tanım nesnesine sistem tarafından yüklenmiş görevler de vardır. Bunlar Küme Görünümüne geçme, Özellik Tanımlama, Küme Bağlama(Tanım Zincirinin oluşturulabilmesi için Bağlı Küme nesnesi aracılığıyla) ve Form Veri Kutusu(Bu görev bu tezde kapsanmayacaktır) tanımlamadır. Özellik nesneleri ve Bağlı Küme nesneleri bu görevler aracılığıyla Tanım nesnelerine iliştirilirler.



Şekil 2.6 Tanım nesnesinin Dizayn penceresindeki görünümü

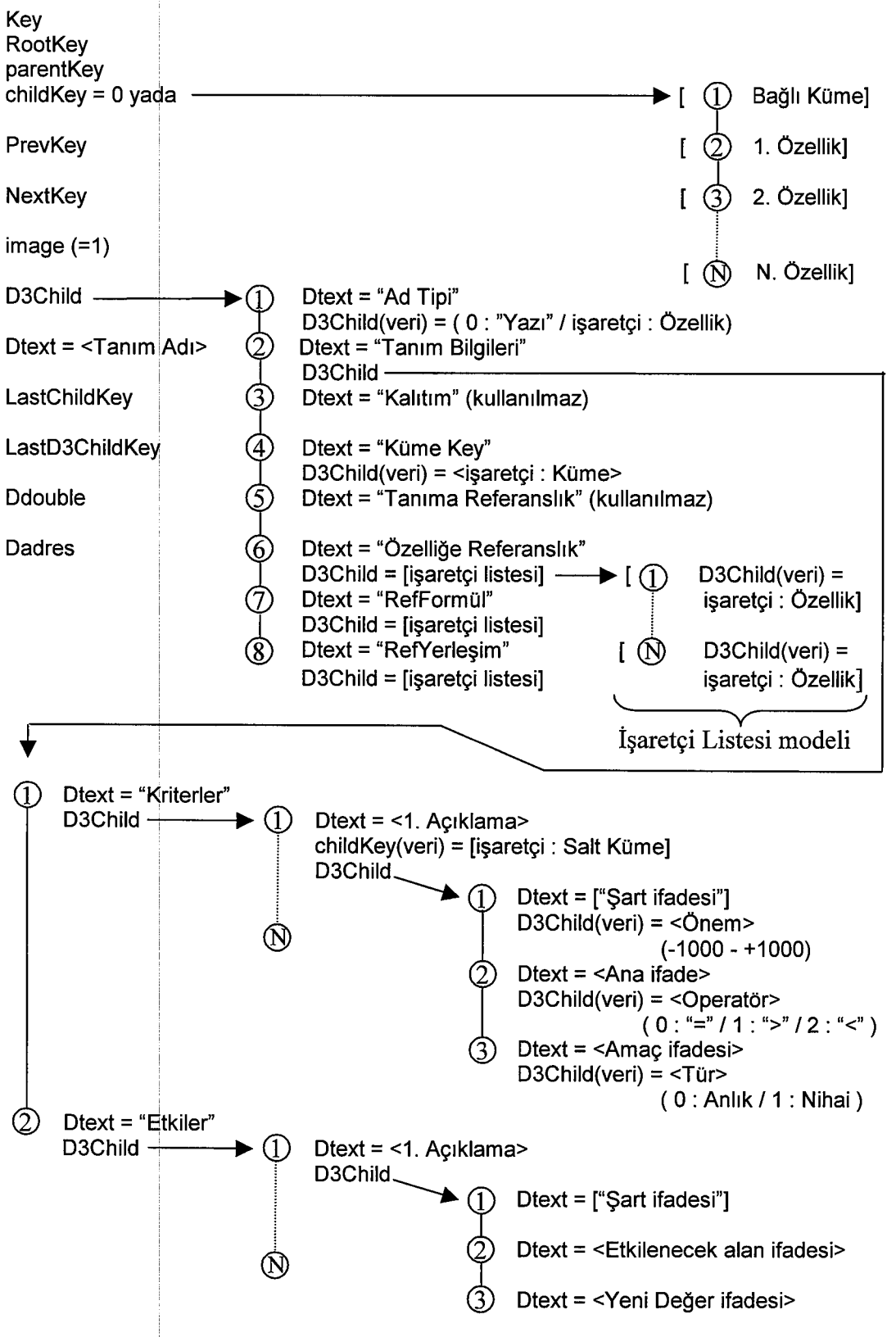
Proje penceresi içinde yeni bir Tanım nesnesi oluşturulabilir, bu Tanım nesnesi için Dizayn penceresi açılabilir. Kullanıcı tarafından oluşturulan Tanım nesneleri istenildiğinde silinebilirler. Tanım nesnelerine ait Küme nesneleri de otomatik olarak Tanım nesnesi ile birlilte oluşturulur ve birlikte silinirler.



Şekil 2.7 Tanım nesnelerinin Proje penceresindeki görünümü

Tanım nesnesinin mantıksal yapısında görüldüğü üzere nesne Özellik nesnesi, Bağlı Küme nesnesi, Kriterler ve Etkiler yapılarını üzerinde taşır. Kendisine ait Küme nesnesi ise Proje nesnesine bağlıdır ancak Tanım nesnesi, içinde bu Küme nesnesini gösteren bir işaretçi bulundurur. Tanım nesnesine bağlı diğer yapılar ise referans yapılarıdır. Bu yapılar aracılığı ile Tanım nesnesi Proje içinde yapılan değişikliklerden haberdar edilir ve gerekiyorsa veri tutarlılığını sağlamak üzere üzerinde otomatik değişiklikler yapılır. Bu yapılar sayesinde bir anlamda Tanım nesnesinin olayları sistem tarafından yönetilir.

Örneğin; bu tanım nesnesinden türetilen gerçek elemanlar bir başka Tanım nesnesinin bir Özellik nesnesi için Nesne Sunucusu olarak görev yapabilir. Bu durumda olan Özellik nesnelerinin indisleri "Özelliğe Referanslık" işaretçi listesi yapısı içinde tutulur ve Tanım nesnesi yada bu Tanım nesnesinden türetilen gerçek nesneler üzerinde kullanıcı tarafından yapılan değişiklikler hakkında ilgili Özellik nesnelerine bu listeden ulaşılarak kendileri üzerinde gerekli düzenlemeleri yapmaları için bilgi verilir.



Şekil 2.8 Tanım nesnesi mantıksal yapısı

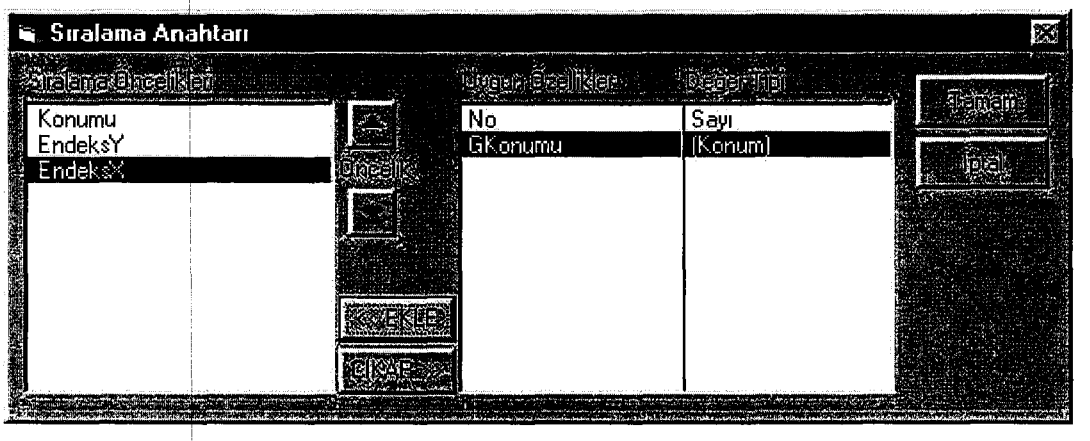
Küme nesnesi

Küme nesnesi adında da alaşılacağı gibi elemanları türetilmiş gerçek nesneler olan bir kümedir. Her Tanım nesnesi için sistem tarafından otomatik olarak bir Küme nesnesi oluşturulur ve kullanıcı tarafından bu Tanımdan türetilen nesneler kendisine ait Küme nesnesi içinde tutulur. Küme nesnesi Proje ve Dizayn pencereleri içinde görülebilir aktif bir nesnedir. Küme nesnesi elemanlarını kullanıcı tarafından yapılandırılabilir bir süzgeç(filtre) aracılığı ile sıraya dizebilir[9], bir başka Tanım nesnesinin bir Özellik nesnesi için nesne sunucusu olarak çalışabilir(Domain) ve en önemlisi Tanım Zinciri'ndeki bir üst Tanım nesnesi için çalışma zamanı(run time) Nesne Sunucusudur. Küme nesnesinin sistem tarafından tanımlanmış(hazır), kullanıcı tarafından değer atanabilen özellikleri aşağıdaki çizelgede gösterilmiştir.

Çizelge 2.2 Küme nesnesi hazır özellikleri

Hazır Özellik	Değer Tipi	Değer Ataması
Ad	<Karakter katarı>	Klavyeden giriş
Sıralama Yönü	<"Artan" yada "Azalan">	Listeden Seçim
Sıralama Anahtarı	[Bağlı-Liste]	İletişim penceresi (dialog)

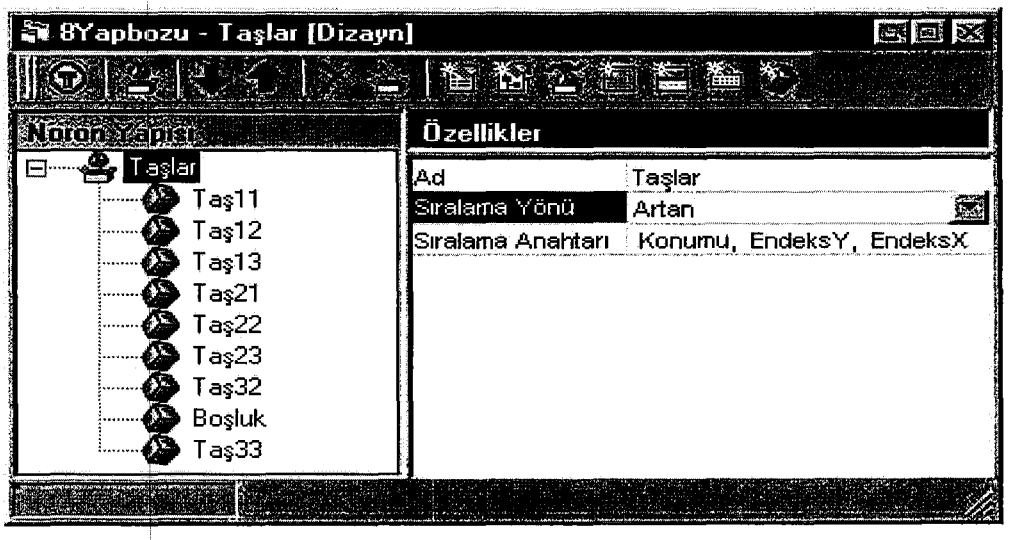
Ad ve Sıralama Yönü özelliklerinin işlevleri isimleri ile aynıdır. Sıralama Anahtarı özelliği ise kullanıcıya sıralamanın nelere göre yapılacağının tanımlanabildiği bir iletişim penceresi açar.



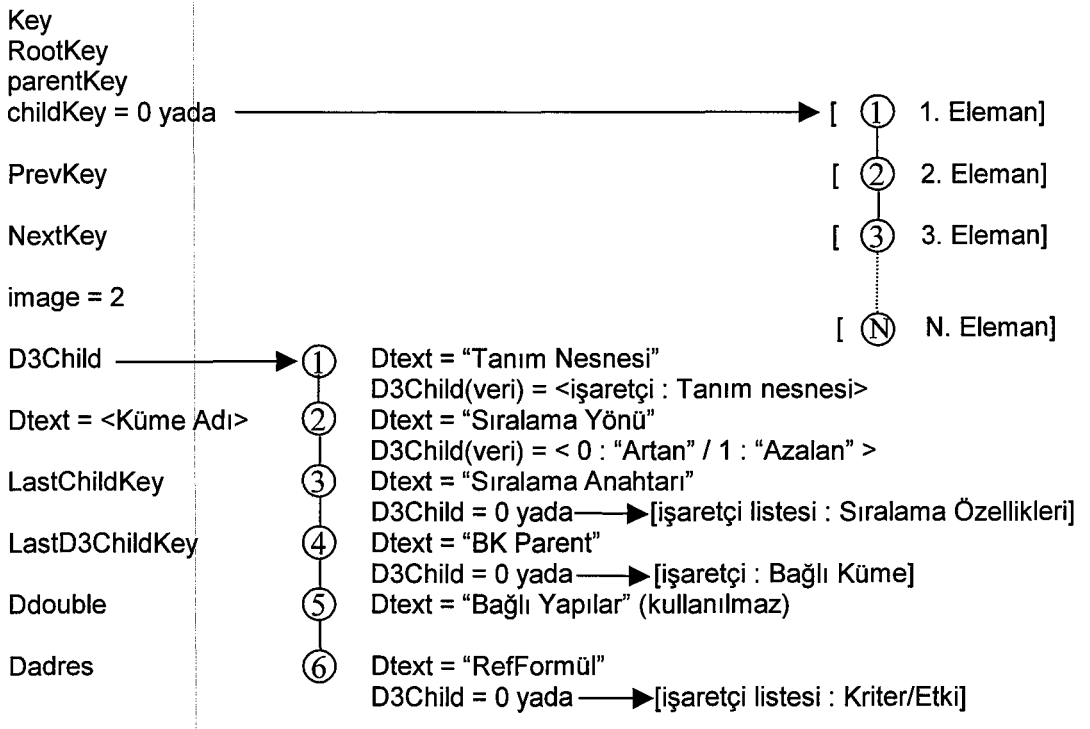
Şekil 2.9 Sıralama tanımlama penceresi

Sıralama için hiçbir tanımlama yapılmazsa elemanlar adlarına göre Türkçe alfabetik yapısında sıralanırlar. Değer tipi Nesne olan Özellik'ler sıralama anahtarı olarak kullanıldığında sıralama bu Özellik'e değer olarak atanmış nesnenin kendi kümesi içindeki sıralandırmasına göre yapılır. Sıralama Anahtarı iletişim penceresinde tanımlanmış sıralama özelikleri üzerinde öncelik düzenlemesi yapılabilir[9].

Küme nesnesine sistem tarafından yüklenmiş görevler Tanım Görünümüne geçme, Kümeye Eleman Ekleme ve Kümenin Tüm Elemanlarını Silmedir. Küme'ye Eleman eklendiğinde bu eleman Küme'nin Tanım nesnesinden türetilir. Eğer bu Tanım nesnesi Tanım Zinciri'nin en üst halkası ise Eleman ile birlikte bu elemanın çalışma zamanı kütüphanesi de otomatik olarak oluşturulur.



Şekil 2.10 Küme nesnesinin Dizayn penceresindeki görünümü



Şekil 2.11 Küme nesnesi mantıksal yapısı

Şekil 2.11 de görüldüğü gibi bir Küme nesnesi elemanları olan gerçek nesneleri mantıksal yapısı içinde taşır. "Tanım Nesnesi" bağlı liste elemanı içinde Küme'nin Tanım nesnesine bir işaretçi, "BK Parent" bağlı liste elemanı içinde Tanım Zinciri'ndeki bir üst Tanım nesnesinin Bağlı Küme nesnesine bir işaretçi ve "RefFormül" bağlı liste elemanı içinde de formülleri içinde Küme'nin adının geçtiği tüm Kriter ve Etkilere birer işaretçi bulunduran bir işaretçi listesi yapısı barındırır.

Özellik nesnesi

Özellik nesneleri sistemin mantıksal yapısı içinde Tanım nesnelerine bağlanmış olarak bulundurulur. Dolayısıyla yeni bir Özellik nesnesinin yaratılması görevi de Tanım nesnesine yüklenmiştir. Programlama dillerindeki sabit(constant) ve değişken(variable) tanımlama imkanları ABS içinde Özellik nesneleri aracılığı ile kullanıcıya sunulur. Özellik nesnesinin sistem tarafından tanımlanmış(hazır), kullanıcı tarafından değer atanabilen özellikleri aşağıdaki çizelgede gösterilmiştir.

Çizelge 2.3 Özellik nesnesi hazır özellikleri

Hazır Özellik	Değer Tipi	Değer Ataması
Ad	<Karakter katarı>	Klavyeden giriş
Özellik Türü	<"Değeri Değişebilir", "Değeri Sabit" yada Görünmez>	Listeden Seçim
Yerleşim	<"Genel", "Lokal" yada bir Üst Tanım>	Listeden Seçim
Değer Tipi	<"Yazı", "Sayı", "Tarih", "Zaman", "Evet/Hayır" yada bir başka Nesne>	Listeden Seçim
Başlama Değeri	<Karakter katarı] (Sadece sayısal tipler için zorunludur)	Klavyeden giriş
Adım	<Karakter katarı] (Sadece sayısal tipler için zorunludur)	Klavyeden giriş
İhmal Değeri	<Karakter katarı / işaretçi] (Sayısal olmayan tipler için zorunludur)	Klavyeden giriş yada Listeden Seçim
Anahtar	"Evet" yada "Hayır"	Listeden Seçim

ABS içindeki Özellik kavramı bir değişkene oranla oldukça karmaşık bir yapıdır. Bu karmaşıklık belirli noktalarda otomasyonun sağlanması için gerekli niteliklerden ve hafıza tüketiminde alınması gereken tedbirlerden ileri gelir.

Bir Özellik nesnesi Değeri Değişebilir, Değeri Sabit yada Görünmez olarak tanımlanabilir. Problem tanımlaması yapılırken bir fark yaratmayan bu tipler çalışma zamanında yada çalışmaya yönelik tanımlamalar(Kriterler ve Etkiler) yapılırken dikkate alınırlar. Çalışma zamanında Görünmez tipler hiç hafıza tüketmezken Değeri Sabit tipler Değeri Değişebilir tiplere nazaran çok daha az hafıza tüketirler.

Bir Özellik nesnesi belirli bir tanım içinde oluşturulmuş olmasına rağmen çalışma zamanında Tanım Zinciri'içindeki üst tanımlar içinde yada Tanım Zinciri'nin de üstünde bir yerde işlem görecek şekilde yapılandırılabilir. Nesne tabanlı programlama dillerinde bir nesnenin tanımında özellikler bulunur. Ancak bu özellikler hiçbir zaman içinde bulundukları nesneden dışarıya çıkmazlar[11]. Başlangıçta her ne kadar anlamsız gibi görünse de ABS içinde bu mümkün ve

gereklidir. Bu gerekliliği açıklamanın en iyi yolunun yine örnekler vermek olduğu değerlendirilmiştir.

Bir Yapay Zeka problemi çözülürken yada daha dar bir kapsamda bir plan üretilirken belirli bir bölüm içerisinde bir plan nesnesinin birden fazla kullanılması olasıdır[1]. Aylık bir plan problemi üzerinde yapılandırılan bir ABS projesini örnek olarak değerlendirecek olursak; Aylık Plan nesnesi 4 adet Haftalık Plan nesnesinden, haftalık plan nesnesi ise 7 adet Nöbetçi nesnesinden(her gün için bir nöbetçi) oluşabilir. Bu problem üç katmanlı olarak değerlendirileceğinden Tanım Zinciri'nin de üç halkalı olacağı açıktır; Ay, Hafta ve Nöbetçi. Her bir nöbetçi için bir hafta içinde tutulan nöbet sayısının önemli bir veri olabileceği değerlendirilirse[2], Nöbetçi nesnesi(Tanım nesnesi) tanımlanırken Özellik'lerinden birisinin "Haftada Tuttuğu Nöbet Sayısı" olması gerekir. Bir planda "Muzaffer" olarak adlandırdığımız kişinin ay içinde 12 adet nöbeti olduğu değerlendirilirse bu özellik de 12 adet olacaktır. Oysa Ay'daki Hafta adedi 4 tür ve 8 adet veri alanı gereksiz yere kullanılmamalıdır. Bu kullanım hem hafızayı fazla tüketir[1] hem de değerlendirmede karmaşıklığa sebep olur. İşte bu noktada ABS'in sunduğu bir imkandan yararlanılarak "Haftada Tutulan Nöbet Sayısı" Özelliği bir üst Tanım olan Hafta nesnesi içine "Yerleşim" özelliği aracılığı ile taşınır.

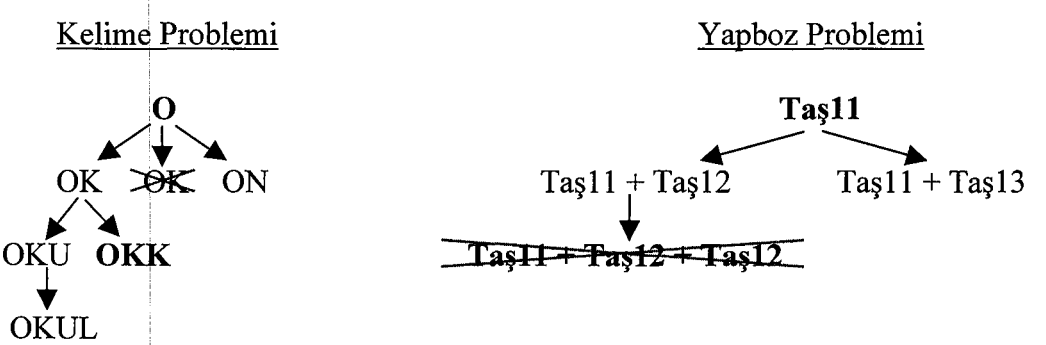
"Nesne" Değer tipinden kasıt Tanım nesneleridir. Bu aşamada ABS henüz diğer programlama dillerinde yada işletim sistemi içinde tanımlanmış nesneleri tanımamaktadır. Örneğin Nöbetçi nesnesi için "Fotoğrafı" adlı bir Özellik nesnesi için bir "Resim dosyası" tanımlanamaz.

"Başlama Değeri" ve "Adım" özelliklerine değerler atanarak Özellik nesnesinin alabileceği değerler filtre edilebilir, "İhmal Değeri" özelliği ile de bir özelliğin değerinin, gerçek nesne ilk türetildiğinde ne olacağı tanımlanabilir.

ABS içinde Anahtar özelliği veritabanı sistemlerindeki Anahtar kavramından oldukça farklı bir anlam taşır. Yapay Zeka problemlerinin arama algoritmaları yolu ile çözümünde tekrarlayan durumların yönetimi başlı başına bir araştırma

[1] Aylık bir nöbet planı içinde bir askerin birden fazla nöbetinin olması yada bir satranç oyununda bir taşın oyun sonuna dek defalarca oynanması.
[2] Örnekteki nöbet hizmetinin nasıl planlanacağını açıklayan kanun ay içindeki sayısı önemli olmaksızın bir kişiye haftada 3 ten fazla nöbet tutturulamayacağını açıkça belirtmiş olabilir.

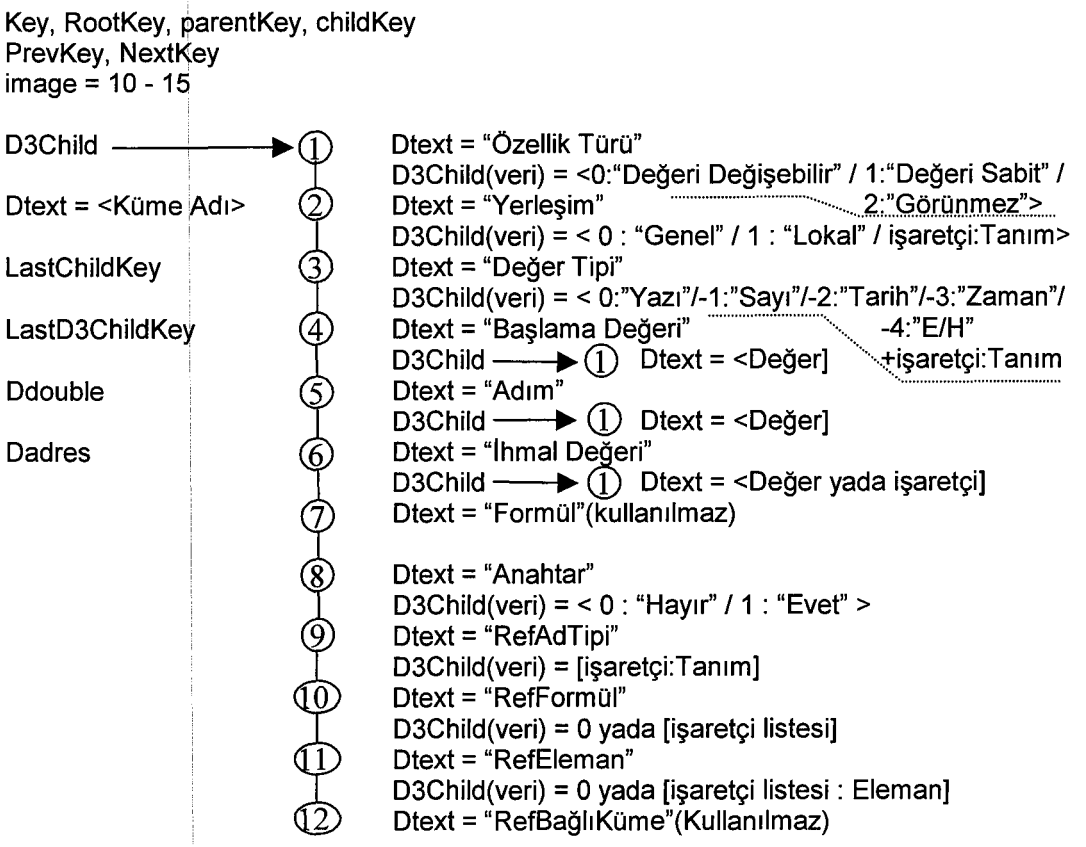
konusudur. Problemlemin doğası çözüm sisteminde kendine özgü bir yaklaşımın belirlenmesini zorunlu kılar. Oysa bir ABS'in problemin doğasına her durumda ayak uydurabilmesi beklenir. Çoğu problemde her yeni planlama aşaması öncekilerden farklı bir durum oluşmasına sebep olur yani çözüm ağacı içinde nesnelerin yerlerinin farklı olması farklı bir durum yaratır. Eğer ağacın daha önce değerlendirilmiş bir hali ile birebir aynı olan bir durum ile karşılaşılırsa yeni durum dikkate alınmaz. Fakat bazı problemlerde yeni bir aşama o anki durumu, ağaç şekli farklı olmasına rağmen daha önce değerlendirilmiş bir duruma geri döndürebilir. Örneğin harflerle anlamlı yada anlamsız kelimeler türeten bir proje ile 8 Yap-boz'u probleminin çözümü için bir proje ele alınırsa;



Şekil 2.12 Tekrarlayan durumlar

Şekil 2.12'de görülen Kelime probleminde "O" durumu ile "OKK" durumu farklı durumlardır çünkü farklı kelimelerdir oysa Yapboz probleminde "Taş11" durumu ile "Taş11 + Taş12 + Taş12" aynı durumdur. Çünkü bir taşı iki kez üst üste oynarsanız taşı önceki yerine geri götürmüş olursunuz ki bu da daha önce değerlendirilmiş bir durumdur. Öyleyse Yapboz probleminde "Taş" nesnesinin "Konumu" Özellik nesnesinin "Anahtar" özelliğinin değeri "Evet" olmalıdır. Bu işlem ABS'e taşların konumlarından oluşan pozisyonun da bir tekrarlama durumu olarak değerlendirmesini söyler. Anahtar özelliği sadece "Genel" değer tipli Özellik nesneleri için "Evet" olarak tanımlanabilir.

[1] Yapay Zeka arama algoritmaları oldukça fazla hafıza tüketirler. Dolayısıyla yapılan her bir tanımlama kendi başına fazla hafıza tüketmese dahi arama algoritması içinde bu tüketim binlerce katına çıkabilmektedir.

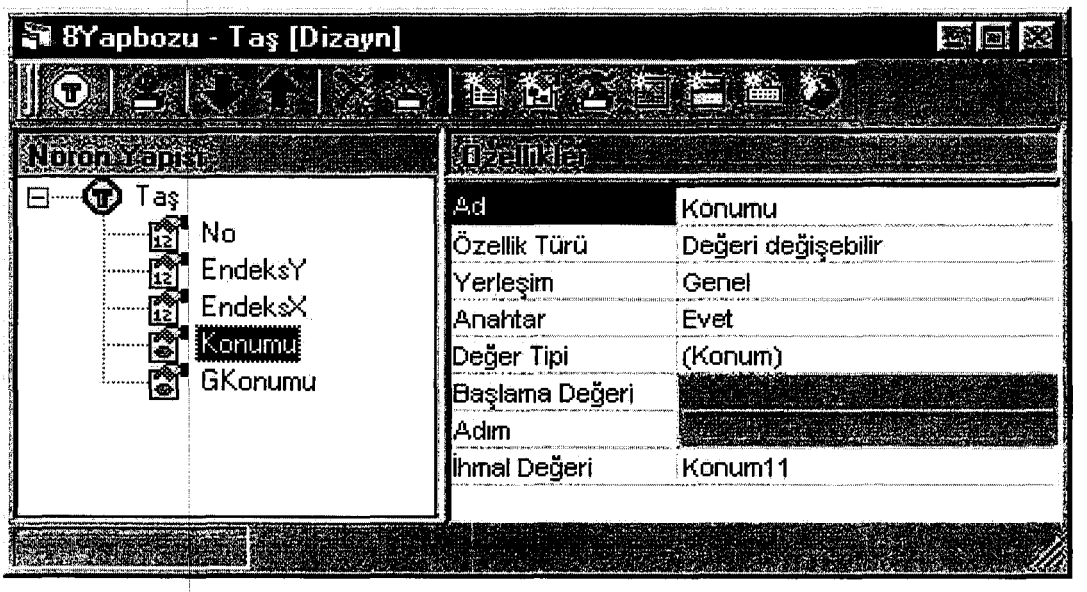


Şekil 2.13 Özellik nesnesi mantıksal yapısı

Özellik nesnesinin ve bazı diğer nesnelerin yapısında görüldüğü gibi bir alana veri olarak atanan değer numaralanmış olarak belirli anlamlar taşıyabildiği gibi aynı alana işaretçi olarak görev yapan bir değer de konulabilmektedir. Bu kullanım Özellik nesnesinin Değer Tipi özelliğinde açıkça göze çarpar. 0 ve -1, -2, -3, -4 gibi negatif değerler sırasıyla "Yazı", "Sayı", "Tarih", "Zaman" ve "Evet/Hayır" tiplerini ifade ederken, pozitif değerler Tanım nesnelerine işaret eden birer işaretçidir.

Bir Tanım nesnesinden geçek bir nesne(Eleman nesnesi) türetilirken aslında Tanım nesnesine bağlı Özellik nesneleri de gerçeklenir. Eleman nesnesi içinde Özellik nesnesinde yapılan tanımlamalar ışığında içine kullanıcı tarafından değer atanabilen veri alanları oluşturulur. Bu alanları gösteren işaretçiler ise Özellik nesnesinin "RefEleman" bağlı liste elemanına iliştirilmiş bir işaretçi listesi yapısında saklanır.

Şimdiye değin yapı ve fonksiyonları açıklanan Proje, Tanım ve Küme nesneleri kök nesneler olup Dizayn penceresi içinde hep en üst nesne olarak görüntülenirler. Oysa Özellik nesnesi de dahil olmak üzere bundan sonra açıklanacak olan sistem nesneleri kök değildirler.



Şekil 2.14 Özellik nesnesinin Dizayn penceresindeki görünümü

Yukarıdaki örnek şekilde görüldüğü gibi kullanıcı tarafından tanımlanmış olan Konumu(Özellik nesnesi), yine kullanıcı tarafından tanımlanmış diğer Özellik nesneleri olan No, EndeksY, EndeksX, Gkonumu ile birlikte Taş Tanımı içinde yer alır ve Değer tipi bir başka kullanıcı tanımlı nesne olan Konum nesnesinin elemanlarıdır. İhmal Değeri ise gerçeklenmiş bir Konum nesnesi(Eleman) olan Konum11 olarak atanmıştır.

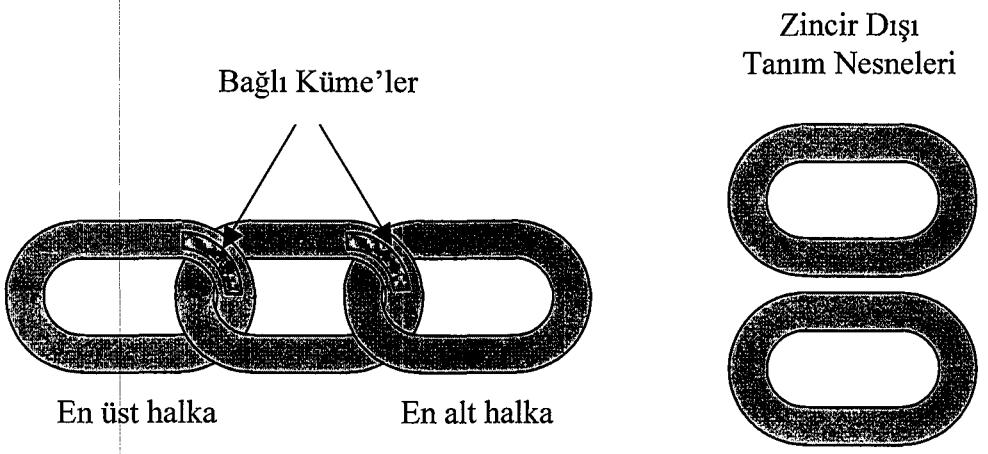
Tanım nesnesinin "Özellik Tanımla" görevi ile oluşturulan Özellik nesneleri Tanım nesnesi altında görülebilir nesnelerdir ve istendiğinde silinebilirler.

Bağlı Küme nesnesi

ABS içinde problem tanımlama mantığının, problemin çatısını oluşturan yapıyı gerçekte bir ağaç şeklinde dahi olsa birbirine ilişkilendirilmiş Tanım nesnelerinden oluşan bir Tanım Zinciri şeklinde tanımlamaya dayandığı, aynı zamanda tamamen bağımsız(Tanım Zinciri'ne dahil olmayan) Tanım nesnelerinin

de olabileceği daha önce de belirtilmiştir. Bağlı Küme nesnesi bu zincirin halkalarının birbiri içinden geçtiği noktaları temsil eder.

Bir zincir halkasının kendinden önceki ve kendinden sonraki halkaya bağlandığı iki nokta vardır(ara halka). Bu bağlantı noktaları üst halkanın alt bağlantısı ile alt halkanın üst bağlantısında bir kesişim kümesi oluşturur. Dolayısıyla tek bir tanımlama yapmak yeterlidir. ABS içinde halka bağlantıları daima halkanın alt ucunda tanımlanır. Bu durumda en alt halkanın bağlantısı tanımlanmayacak yani Tanım Zinciri'nin en alt halkasını oluşturan Tanım nesnesinin bir Bağlı Küme'si olmayacaktır.



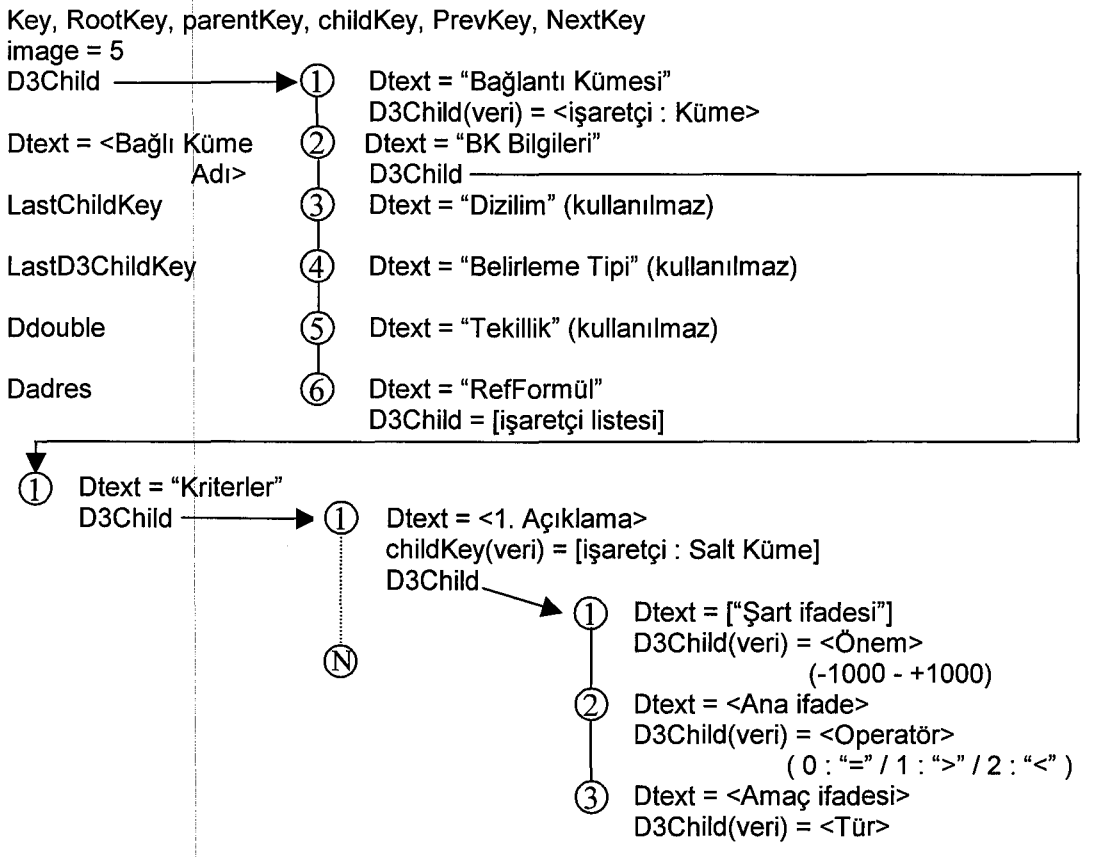
Şekil 2.15 Bağlı Küme nesnesinin mantıksal konumu

Bağlı Küme nesnesinin sistem tarafından tanımlanmış(hazır), kullanıcı tarafından değer atanabilen özellikleri aşağıdaki çizelgede gösterilmiştir.

Çizelge 2.4 Bağlı Küme nesnesinin hazır özellikleri

Hazır Özellik	Değer Tipi	Değer Ataması
Ad	<Karakter katarı>	Klavyeden giriş
Bağlantı Kümesi	< Küme'ye işaretçi >	Listeden Seçim
Kriterler(seçim)	[Bağlı-Liste]	İletişim penceresi (dialog)

Her ne kadar Tanım Zinciri kavramı açıklanırken sadeceTanım nesnelerinden söz edilmiş olsa da gerçekte bir Tanım nesnesi bir alt Tanım nesnesi Bağlı Küme nesnesi aracılığı ile ilişkilendirilirken alt Tanım nesnesinin bizzat kendisinin değil Küme'sinin kullanılması daha anlamlıdır. Tanım Zinciri bir tanımlamadır. Gerçekte ifade ettiği yapı bir ağaç yapısını temsil ettiğinden gerçeklenme esnasında tek bir halka olarak tanımlanan alt Tanım nesnesinin yerini bu tanımdan türetilmiş birden fazla gerçek nesne alabilecektir[1]. Dolayısı ile bağlantı nesne tanımı yerine o nesne tanımının Küme nesnesi kullanılarak yapılır. Bu yönteme kullanıcının daha rahat adapte olabileceği değerlendirilmiştir.

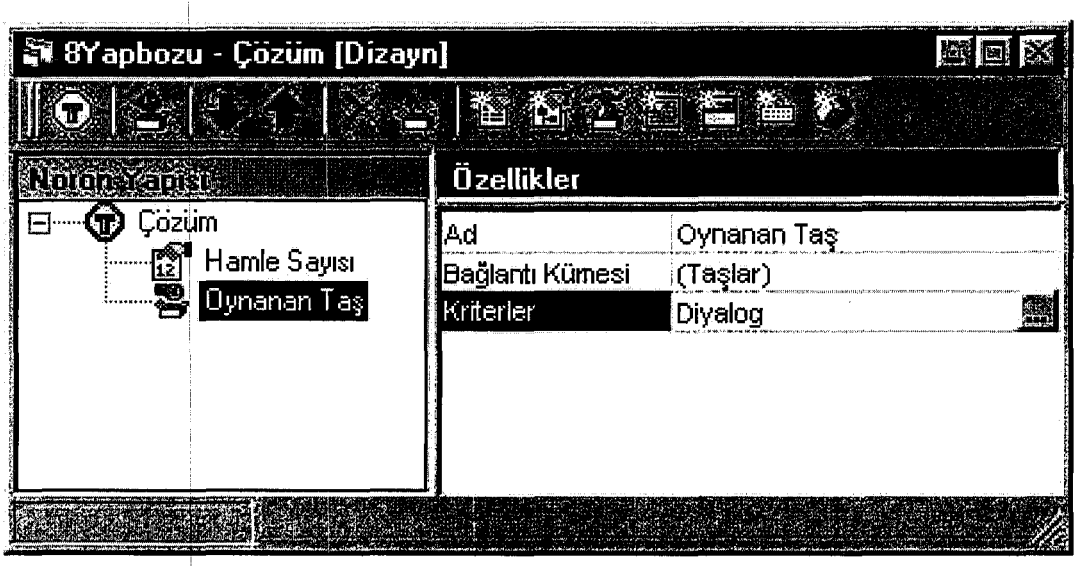


Şekil 2.16 Bağlı Küme nesnesi mantıksal yapısı

[1] "Ay – Hafta – Nöbetçi" problemi ele alınacak olursa çözüm önerisi olarak oluşturulacak planda "Ay" nesnesinin altına 4 adet "Hafta", "Hafta" nesnesinin altına ise 7 adet "Nöbetçi" nesnesi gelecektir.

Bağlı Küme nesnesinin de tanımlanabilir Kriterleri vardır. Bu Kriterler Tanım yada Eleman nesneleri için tanımlanan kriterlerden bir yönü ile farklılık gösterir. Bu fark yüzünden Bağlı Küme nesnesinin Kriterleri açıklamalarda "Seçim Kriterleri" olarak adlandırılmıştır. Gerçekten de Seçim Kriterleri aslında çözüm öneri planı üretilirken alt nesnelerin seçiminde kullanılacak kriterlerdir.

Tanım nesnesine yüklenmiş "Küme Bağla" görevi ile her Tanım nesnesi içine sadece bir adet oluşturulabilen Bağlı Küme nesnesi Tanım nesnesi altında görülebilir bir nesnedir ve kullanıcı tarafından istendiğinde silinebilir.



Şekil 2.17 Bağlı Küme nesnesinin Dizayn penceresindeki görünümü

Eleman nesnesi

Eleman nesnesi ABS'in sayıca en çok ve en önemli görevlerini üstlenen, yapısı en karmaşık nesnesidir. Eleman nesnesi bir tanım nesnesinden türetilen gerçek nesneleri temsil eder. Nesne tabanlı programlama dillerinde sıklıkla verilen "Kedi" örneği Eleman nesnesini tek cümlede açıklamaktadır. Eğer "Kedi" bir Tanım nesnesi ise "Kediler" Küme nesnesi içindeki "Tekir" ve "Minnoş" onun Eleman nesneleridir.

Eleman nesnesi bünyesinde sistem tarafından tanımlanmış hazır özelliklerle birlikte aynı zamanda kullanıcı tarafından tanımlanmış ve değer atanabilen özellikleri de dinamik olarak barındırır. Eleman nesneleri türetildikleri Tanım nesnelerine sıkı sıkıya bağlıdırlar. Bir Tanım nesnesine yeni bir Özellik nesnesi

eklendiğinde bu yeni özellik Tanımın Küme'sinde daha önceden yaratılmış tüm Eleman nesnelerine otomatik olarak yansıtılır.

Tanım Zinciri'nin en üst Tanım nesnesinin Elemanları kendi başlarına proje içinde yapılmış tüm tanımları, kısacası problemin tüm yapısını ihtiva ederler. Bu Eleman nesneleri aynı zamanda problemin çözümü için çalıştırılabilir birer uygulamadır ve çalıştırıldıklarında "Öneri Plan"a dönüşürler.

Eleman nesnesinin sistem tarafından sunulan hazır özellikleri sadece Ad, Kriterler ve Etkiler'dir. Önceki bölümlerde kısmen değinilen ve ileride detayları açıklanacak olan Kriterler ve Etkiler Tanım nesnelerine tanımlanabildiği gibi Eleman nesnelerine de diğer Eleman nesnelerinden bağımsız olarak tanımlanabilirler. Eleman nesnesinin sistem tarafından tanımlanmış(hazır), kullanıcı tarafından değer atanabilen özellikleri aşağıdaki çizelgede gösterilmiştir.

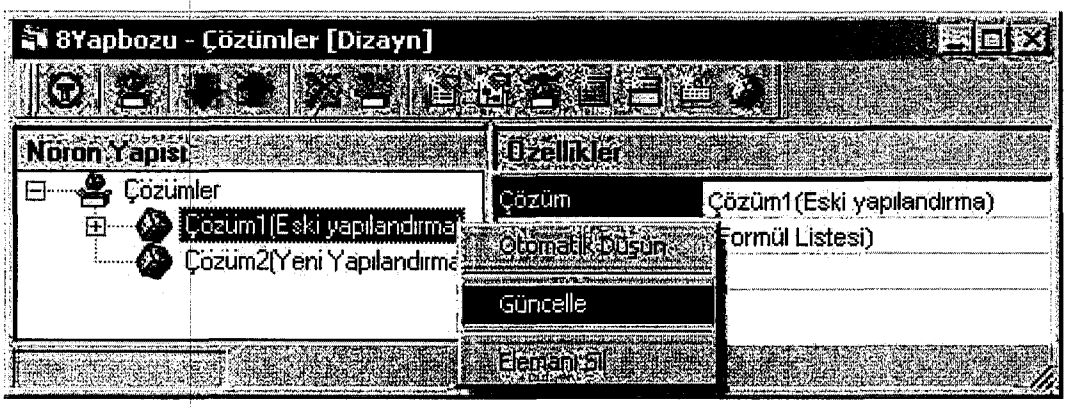
Çizelge 2.5 Eleman nesnesinin hazır özellikleri

Hazır Özellik	Değer Tipi	Değer Ataması
Ad	<Karakter katarı>	Klavyeden giriş
Kriterler	[Bağlı-Liste]	İletişim penceresi (dialog)
Etkiler	[Bağlı-Liste]	İletişim penceresi (dialog)

Yukarıdaki çizelgede gösterilen özelliklere ilave olarak Eleman nesnesi kullanıcı tarafından tanımlanmış özellikleri de ihtiva eder.

Eleman nesnesine sistem tarafından yüklenmiş görevler, Kendini güncelleme ve Otomatik Düşünmedir. ABS içinde tanımlanan bir problemin doğasına ait tüm bilgileri ihtiva eden Tepe Eleman nesneleri bu bilgilerini otomatik olarak güncellemezler. Çünkü problem tanımı üzerinde yapılan mantıksal ve yapısal değişiklikler çoğu kez bu değişikliklerin yürürlüğe girmesinden sonra oluşturulan planlar için geçerlidir. Yeni çıkan bir kanuna göre suç sayılan ve cürümü bu kanundan önce işlenmiş eylemler için bir kişinin suçlu sayılamayacağı ilkesini hatırlayınız[TC. Anayasası]. Tıpkı burada olduğu gibi eski yapılandırmaya göre hazırlanmış ve uygulanmış planlar da değişiklik yapılandırmasına göre kurallara uymayabilir. Ancak uygulama aşamaları eskide kalmıştır ve durumun ne geri dönüşü vardır ne de bu eski uygulamalar hakkında eleştiri yapmanın anlamı... İşte bu yüzden Eleman nesneleri otomatik olarak değil, kullanıcı tarafından verilen bir

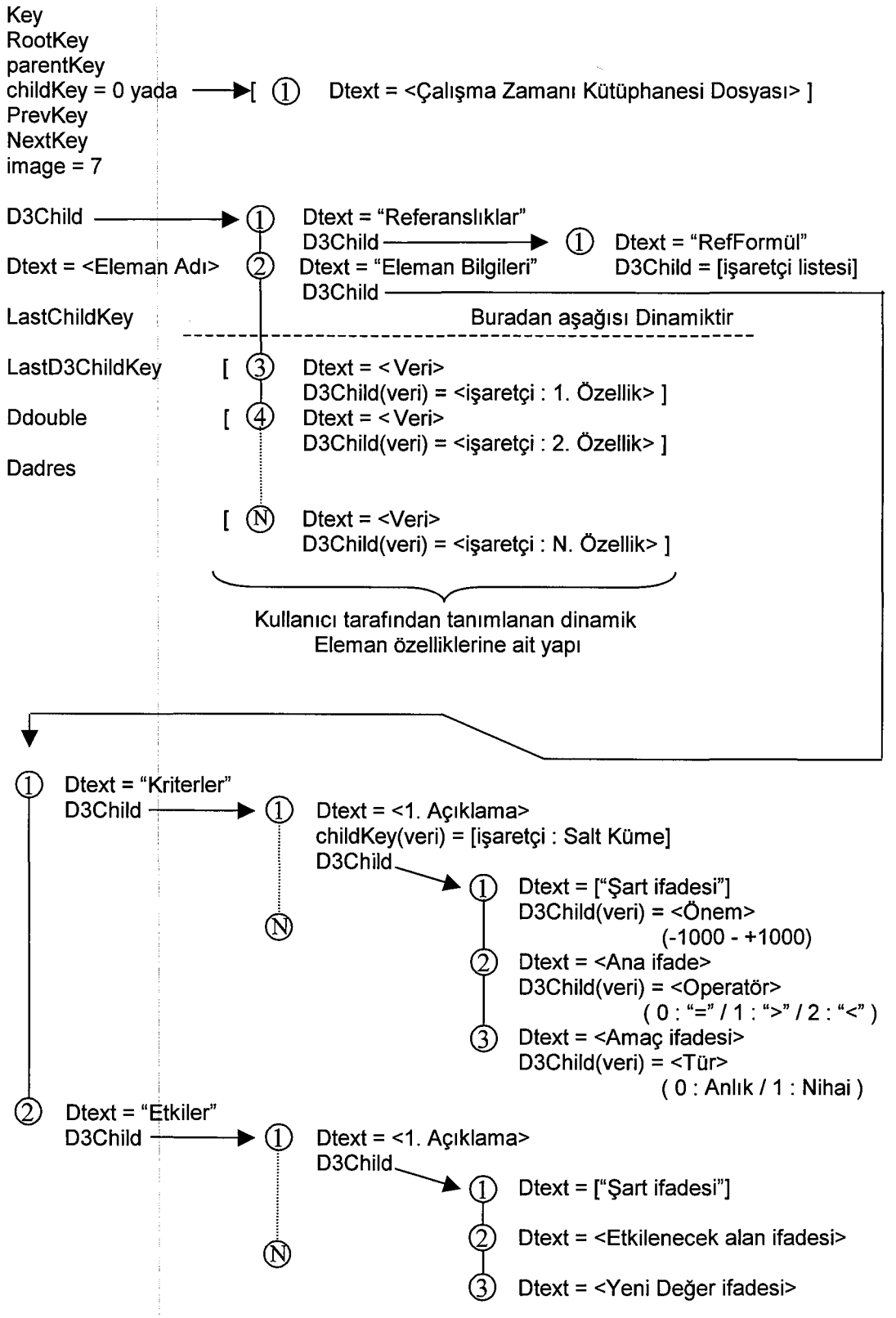
komut ile güncellenirler. Problem yapılandırması değişikliğe uğramış bir projenin eski elemanları güncellenmedikçe Otomatik Düşünme işlemini eski yapıya göre yürütürler[1].



Şekil 2.18 Eleman nesnesinin Dizayn penceresindeki görünümü

Eleman nesnesi yapısında Eleman bağlı-liste elemanının childKey alanına bağlı Çalışma Zamanı Kütüphanesi ve yapısı ilerde Yürütme bölümü içinde detayları ile açıklanacak karmaşık bir yapıdır. Bu bölümde üzerinde durulacak konuyu daha çok dinamik Eleman özelliklerine ait yapı oluşturur. Bu yapı Eleman özelliğine atanan veriyi karakter katarı tipinde tutan, türediği Özellik nesnesine de bir işaretçi ile bağlanan bir yapıdır. Eleman nesnesi bu yapıdan türediği Tanım nesnesindeki Özellik nesnelerinin sayısı kadar bulundurur.

[1] Bu işlem planlama sistemleri söz konusu olduğunda çok anlamlıdır. Planlama problemleri tanımlanırken bir planın daima geleceğe yönelik olarak hazırlandığını hatırlayınız.



Şekil 2.19 Eleman nesnesi mantıksal yapısı

Kriter ve Etki tanımlamaları

"Kriter - Etki" yaklaşımı ABS'in nesnel tanımlamalar ile bütünleşen en önemli mantıksal yapısıdır. Nesnel tanımlamalar ile fiziksel anlamda tanımlanan problemin, kriterler aracılığı ile kuralları, üreteceği çözümde bulunması istenen nitelikler ve hatta çözüme ulaşmayı kolaylaştıracak sezgisel metotlar tanımlanabilirken, etkiler aracılığı ile problem akışı içindeki olayların problemin durumu üzerinde yapacağı değişiklikler tanımlanabilir.

ABS'içinde Tanım nesnelerine, Eleman nesnelerine ve Bağlı Küme nesnelerine kriterler tanımlanabilmektedir. Tanım ve Eleman nesnelerine tanımlanan kriterler Genel Sonuç Kriterleri, Bağlı Küme nesnelerine tanımlanan kriterler ise Seçim Kriterleri olarak adlandırılmıştır. Genel Sonuç Kriterleri ile Seçim Kriterleri arasında tanımlama şekli ve yapı bakımından bir fark bulunmazken kriterlerin üzerine uygulandığı nesneler ve ulaşılabilirlik açısından fark bulunmaktadır. Etkiler ise kriterlerden farklı olarak Bağlı Küme nesnelerine tanımlanamazlar.

Satranç oyununun ara aşamaları üzerinde bir örnek verilecek olursa; bir taşın bir başka taşı alması söz konusu olduğunda alacak taşın alınacak taşın konumuna gidip gidemeyeceği SeçimKriterleri ile, hamlenin yapılmasıyla hem alan taşın konumunun değişmesi ile hem de alınan taşın oyun dışı kalmasıyla satranç tahtası üzerinde olan değişiklikler Etkiler ile ve yeni durumun her iki rakip taraf için oluşturduğu avantaj ve dezavantajın kıymetlendirilmesi ise Genel Sonuç Kriterleri ile tanımlanır.

Kriterler ile Etkiler işlevsel olarak birbirlerinden oldukça farklı kavramlar olmasına karşın kullanım prensipleri birbirlerine büyük benzerlikler gösterir. Yukarıdaki örnekten de anlaşılabileceği gibi kriterler aracılığı ile bulanık tanımlamalar yapılabilmektedir. Etkiler ise belirli tanımlamalardır.

Bu noktada bulanıklık konusunun ve ABS içinde nasıl işlendiğinin detaylandırılması gerekli görülmüştür. Deterministik metotlarla çözümün bulunması mümkün olmayan problemlerde çözüm kümesinde bulunması istenen niteliklerin tanımlanması bulanık tanımlamaların yapılabilmesini gerektirir. Matematiksel bir ifadede eğer değişkenlerin alabileceği değerler kümesi sınırlandırılmış ise tanımlamada göreceli ifadeler kullanılabilir.

Örneğin :

$$\left\{ \begin{array}{l} \text{Minimum } \mathbf{X}\,; \\ \\ \mathbf{X}^2 + 1 = 5 \end{array} \right\}$$

X'in çözüm kümesi {-2} dir. Denklem gurubunda "$\mathbf{X}^2 + 1 = 5$" ifadesi olmaksızın "Minimum X" ifadesi anlamsızdır. Çünkü "Minimum X" göreceli bir ifadedir. Yine de "$\mathbf{X}^2 + 1 = 5$" denklemi deterministiktir ve denklem gurubunun çözümü kolayca bulunabilir.

Minimum, maksimum, küçüktür, büyüktür vb. ifadelere de genellikle vertabanı sistemleri yada dizi işlemleri gibi işleme sokulacak değerlerin neler olduğu ve sınırlarının belirli olduğu yada deterministik ifadeler ile bulunabildiği alanlarda rastlanır. Oysa bir Yapay Zeka problemi sözkonusu olduğunda göreceli ifadeler sıklıkla karşılaşılan ifadelerdir fakat işleme sokulacak değerler ve sınırları deterministik olmayan diğer denklemler çözülmeden bilinemez. Kaldı ki bu denklemlerin çözüm kümelerinin sınırları bilinmeyen bir değerden başlayarak astronomik rakamlara ulaşabilmektedir[1].

Bir arama algoritmasının olabilecek en zeki çözüme ulaşması, sezgisel metotlar ile bilgilendirilse dahi kabul edilebilir süre ve hafıza gereksinimi sınırlarının dışına çıkabilmektedir. Bundan dolayıdır ki problemin doğasına bağlı olarak en iyi değil de iyi sayılabilecek çözümlere makul süre ve hafıza harcaması ile ulaşabilen algoritmalar(complete but non optimal) ile yetinmek gerekebilir. Klasik arama stratejilerini kullanan ABS elbette sihirbaz değildir ve optimal olmayan bir çözümü de üretebilmelidir. Bu noktada ABS'in elastiki tanımlama mantığı yine işe yaramakta, algoritmanın kullanacağı strateji de kullanıcı tarafından tanımlanabilmektedir.

Arama stratejilerinin bir uzmanlık alanı olduğunu söylemek pek de yanlış olmaz[12]. Bu durumda hem kullanıcının bu bilgilerden soyutlanması hem de arama stratejilerini tanımlayabilmesi bir tezat gibi görünmektedir. Gerçekte arama stratejisi yapılan tanımlamaların doğal bir sonucu olarak oluşturulur. Dolayısıyla

[1] Durum tekrarı yapılmaksızın 10 hamle ile bozulmuş bir Rubik Küpü'nün 10 hamleden daha fazla hamle ile çözüme giden en az 3.57×10^{12} adet çözümü vardır. Bu rakam hesaplanan en az rakamdır ve gerçekte çok daha fazla olabilir. Bu hesaplamayı yaparken şanslıyız çünkü en azından en zeki çözümün 10 hamleden oluştuğunu biliyoruz(bu genelde bilinmeyen bir veridir).

kullanıcıya düşen sadece "ben olsaydım nasıl çözmeye çalışırdım" sorusunun cevabını tanımlamalarına yansıtmasından ibarettir.

Bir kriter oluşturulurken kullanıcıdan mümkün olmasa bile en zeki çözümden beklentilerini tanımlaması fakat bu beklentinin gerçekleşmesinin ne kadar önemli olduğunu da tanımlaması istenir. Örneğin bir yapbozu hiç hamle yapmadan çözmesini istemek ve bunun 1000 üzerinden 1 birim önemli olduğunu belirtmek aslında Minimum Hamle Sayısı'nı ifade etmenin bir başka yoludur.

$$\left\{ \begin{array}{l} \text{Problem} \\ \text{Minimum Hamle Sayısı;} \\ \text{Amaç durum} \end{array} \right\} = \left\{ \begin{array}{l} \text{Problem} \\ \text{Hamle Sayısı} = 0, \text{Önem} = 1/1000 \\ \text{Amaç Durum} \end{array} \right\}$$

Yukardaki ifade şekli çözümden beklentilerin sayısı birden fazla olduğunda ve bu beklentiler birbirlerinin istendiği şekilde gerçekleşmesini engelliyor olduğunda anlam kazanır[1] ve bulanık bir ifade şekline dönüşür.

Önem derecesinin 1000 olması(1000 / 1000) zorunluluğu ifade eder ve çözümde mutlaka sağlanmaya çalışılır. "Zorunlu" olarak tanımlanmış bir beklentinin sağlanamaması çözümsüzlüğü ifade etmektedir[2]. Önem dereceleri negatif olarak da tanımlanabilirler. Negatif önem dercesi istenmeyen bir durumu niteler. Önem dercesi 0 olan bir kriter ise çözümde hiç dikkate alınmaz.

Kriterler çözümün alt aşamaları için de(kurallar olarak) tanımlanabilirler. Bir alt aşama için tanımlanmış zorunlu kriterler çözüme ulaşılmadan sağlanamayabilir. Bu durumda ya problemin bir çözümü olmadığı belirtilecek ya da o an için aşılmasına müsade edilip nihai çözümde aranacaktır[3].

---

[1] Sadece az para harcamak istiyorsanız bunu ne kadar istediğiniz önemli değildir, işinize yarayacak en ucuz ürünü satın alırsınız. Fakat hem az para harcamak hem de kaliteli bir ürün satın almak istiyorsanız ürünün fiyatının ve kalitesinin sizin için ne kadar önemli olduğunu ayrı ayrı belirtmelisiniz. A.B.S. size hangi ürünü almanız gerektiğini söyleyecektir.

[2] Her problemin belirli şartlarda bir çözümünün olması gerekli değildir. Belirli şartlarda çözümün olmadığını bilmek de aslında bir çözüm sayılır.

[3] Bir yap-boz'un çözümünde tüm taşların kendi orijinal yerlerinde olmaları bir zorunluluktur. Ancak bozulmuş bir yap-boz için bu durum ilk hamlede sağlanamayabilir.

Bu fark Kriter Türü olarak tanımlanır. "Anlık" olarak tanımlanan kriterler zorunlu olduklarında ara aşamalarda aşılamazken, bir kriter zorunlu olarak tanımlansa dahi eğer türü "Nihai" ise ara aşamalarda değil, fakat çözümde sağlanıyor olması beklenir. Zorunlu olmayan kriterler için(1 – 999) "Anlık" tanımlaması geri dönüşü olmayan bir optimumdan uzaklaşmayı "Nihai" tanımlaması ise ilerleyen aşamalarda bu uzaklaşmanın telafi edilebileceğini anlatır.

Bir kriter tanımında temel alanlar Ana Parametre ve Karşılaştırma Değeri'dir. Bu iki alan aralarında uygulanacak olan bir operatör ile geriye değer döndüren matematiksel bir fonksiyona benzetilebilir. Kullanılabilecek operatörler ise eşitlik(=), büyüklük(>) ve küçüklük(<)tür. Ana Parametre ve Karşılaştırma Değeri arasında kullanılan operatörlerin işlevleri programlama dillerinde kullanımlarından biraz farklı olarak; eşitlik eşit ifadeler için "0" değerini, farklı ifadeler içinse farkı geri döndürürken, büyüklük ve küçüklük yalnızca Evet/Hayır değeri geri döndürür. Bir listeden seçilen bu ara operatörlerin ifadeler içinde kullanılan "=", ">" ve "<" operatörleri ile karıştırılmaması gerekir zira bunlar programlama dillerinde olduğu gibi sadece Evet/Hayır değeri döndürürler.[11] Etkiler için ise ara operatör daima atama(=) operatörüdür ve Ana Parametre ile Karşılaştırma Değeri alanlarının adları sırasıyla Etkilenecek Alan ve Yeni Değer olarak kullanılır.

ABS içinde kriter ve etkiler tanımlanırken kullanılabilecek matematiksel operatörler ve tahditleri aşağıdaki tip-işlem-sonuç çizelgesinde gösterilmiştir. Bu tip-işlem-sonuç tablosu sistemin bugünkü haline işaret etmekte olup gelecekte daha çok işlevsellik için geliştirilebileceği değerlendirilmektedir.

Çizelge 2.6 Tip-İşlem-Sonuç tablosu

Operatörler / Parametreler	Atama Op. (=)	Ara Op. (=,>,<)	Üs ^	*	/	+	-	Kont. =,>,<	Mantık &, |
Yazı - Yazı	Yazı	E/H				Yazı		E/H	
Yazı - Sayı	Yazı							EH(=)	
Yazı - Tarih	Yazı							EH(=)	
Yazı - Zaman	Yazı							EH(=)	
Yazı - E/H	Yazı							EH(=)	
Yazı - Nesne	Yazı							EH(=)	
Sayı - Yazı	Sayı							EH(=)	
Sayı - Sayı	Sayı	Fark, EH	Sayı	Sayı	Sayı	Sayı	Sayı	E/H	Sayı
Sayı - Tarih						Trh			
Sayı - Zaman	Sayı	Fark, EH	Sayı	Sayı	Sayı	Sayı	Sayı	E/H	Sayı
Sayı - E/H	Sayı	Fark, EH	Sayı	Sayı		Sayı	Sayı	E/H	Sayı
Sayı - Nesne								E/H	
Tarih - Yazı								EH(=)	
Tarih - Sayı						Trh	Trh		
Tarih - Tarih	Tarih	Fark, EH					Sayı	E/H	
Tarih - Zaman						Trh	Trh		
Tarih - E/H						Trh	Trh		
Tarih - Nesne								E/H	
Zaman - Yazı								EH(=)	
Zaman - Sayı	Zaman	Fark, EH	Zm	Zm	Zm	Zm	Zm	E/H	Zm
Zaman - Tarih									
Zaman - Zaman	Zaman	Fark, EH	Zm	Zm	Zm	Zm	Zm	E/H	Zm
Zaman - E/H		Fark, EH	Zm	Zm				E/H	Zm
Zaman - Nesne								E/H	
E/H - Yazı	E/H							EH(=)	
E/H - Sayı	E/H	Fark, EH	Sayı	Sayı		Sayı	Sayı	E/H	Sayı
E/H - Tarih									
E/H - Zaman	E/H	Fark, EH	Zm	Zm				E/H	Zm
E/H - E/H	E/H	Fark, EH	Sayı	E/H		Sayı	Sayı	E/H	E/H
E/H - Nesne									
Nesne - Yazı								EH(=)	
Nesne - Sayı								E/H	
Nesne - Tarih								E/H	
Nesne - Zaman								E/H	
Nesne - E/H									
Nesne - Nesne	Nesne	Fark, EH						E/H	

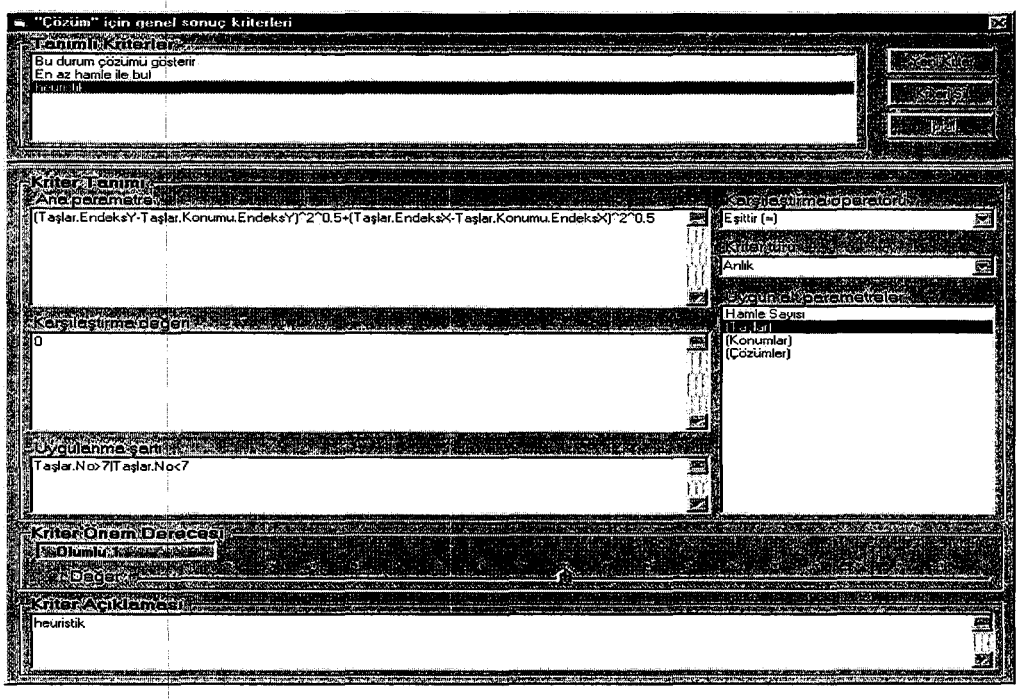
Tanım nesnesi içinde tanımlanan ve Tanımın tüm gerçek nesnelerine uygulanan kriterler ve etkilerden bazı durumlarda belirli özellikteki elemanların muaf tutulmaları istenebilir. Böyle bir durumda kriterin veya etkinin Tanım nesnesinden çıkarılıp muaf olacaklar hariç her bir Eleman nesnesine tek tek tanımlanması gerekir. Bu işlem hem ABS'in elastikiyet prensiplerine aykırı, zahmetli bir iştir hem de akıcılığı bozar. Gerçek nesnenin bir özelliğinin değeri işleyiş aşamalarında da değişebileceğinden bu değişikliğin gerçek nesneyi kriter yada etkiden muaf tutup tutmayacağına çözüm arama aşamaları devam ederken de

karar verilebilmelidir. Bu amaçla kriter ve etkiler tanımlanırken beraberinde geriye sadece Evet/Hayır değeri döndüren bir fonksiyon olarak işlev gören Uygulanma Şartı ifadesi de tanımlanabilmektedir. O andaki duruma uygulanan bu Şart ifadesi eğer "Evet" değeri döndürmüyorsa Kriter yada Etki uygulamaya sokulmadan işlemlere devam edilir.

Bir nesne için tanımlanan kriterlerin uygulanması kendi içinde bir sırayı gerektirmezken etkilerin hangi sıra ile uygulanacağı önemlidir. Bu amaçla bir nesne için tanımlanan etkilerin öncelik sırası değiştirilebilmektedir.

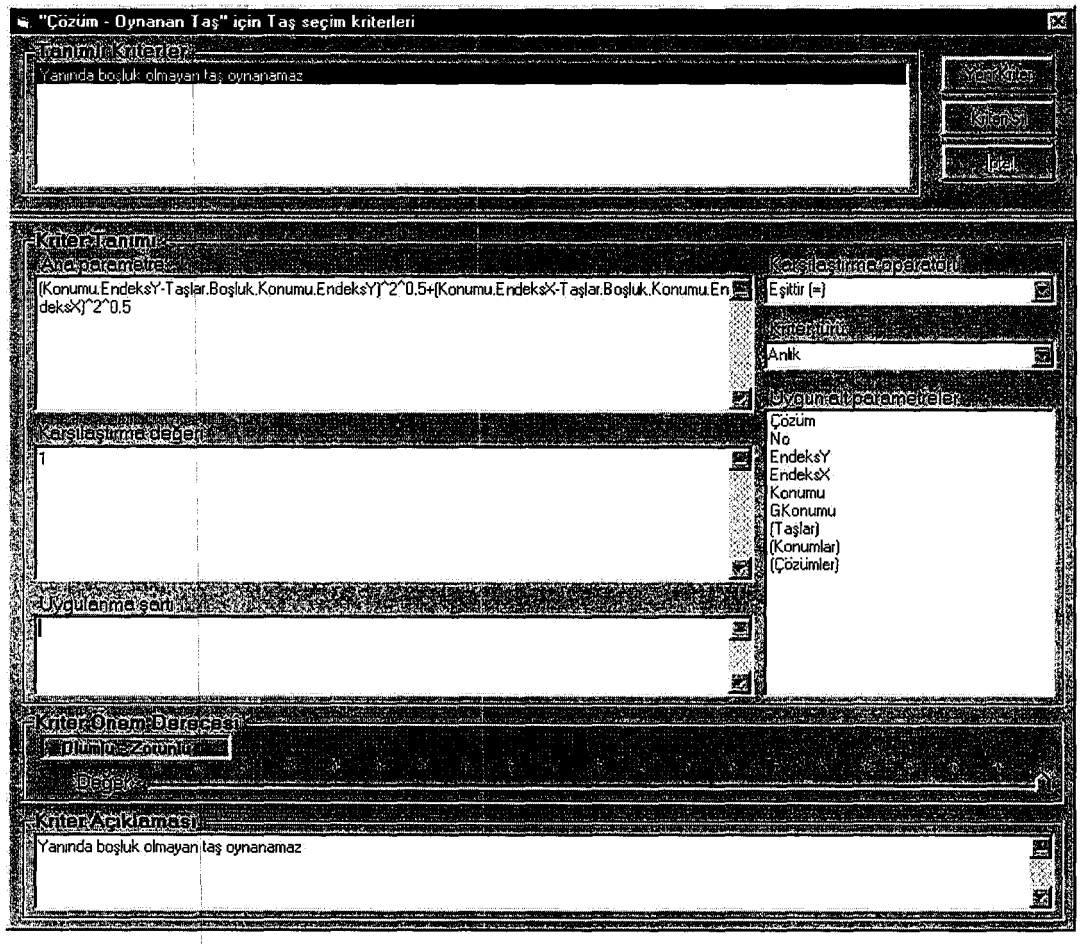
Son olarak tüm kriter ve etkiler için okunabilirliği arttırmak amacıyla veri girişinin zorunlu olduğu bir "Açıklama" alanı oluşturulmuştur. Bu açıklama alanına doğal dille yazılan ifade hem kriter yada etkinin adını hem de manuel bir çözüm adımının tanımlanan kurallara aykırı olması durumunda görüntülenecek hata mesajının metnini oluşturur.

İşlem aşamalarında kullanılan değişmez sıra; önce Seçim Kriterleri, daha sonra Etkiler ve son olarak da Genel Sonuç Kriterleri şeklindedir. Kriter ve etkilerin tanımlanması aşamasında ABS artık kullanıcıyı yönlendirebilecek derecede problemi algılamıştır. Hatalı tanımlamalara müsade edilmezken ulaşılabilir veriler de seçim için "Uygun Alt Parametreler" penceresinde listelenir.



Şekil 2.20 Kriter tanımlama penceresi

Şekil 2.20 de 8 Yap-boz'u problemi için oluşturulmuş Çözüm nesnesi için açılmış Genel Sonuç Kriterleri penceresi görülmektedir.



Şekil 2.21 Seçim Kriterleri tanımlama penceresi

Şekil 2.21 de 8 Yap-boz'u problemi için oluşturulmuş Oynanan Taş Bağlı Küme nesnesi için açılmış Seçim Kriterleri penceresi görülmektedir.

Şekil 2.22 Etki tanımlama penceresi

Şekil 2.22 de 8 Yap-boz'u problemi için oluşturulmuş Taş nesnesi için açılmış Etkiler penceresi görülmektedir.

2.2.1.2 Veri tutarlılığının kontrol edilmesi (zorlanması)

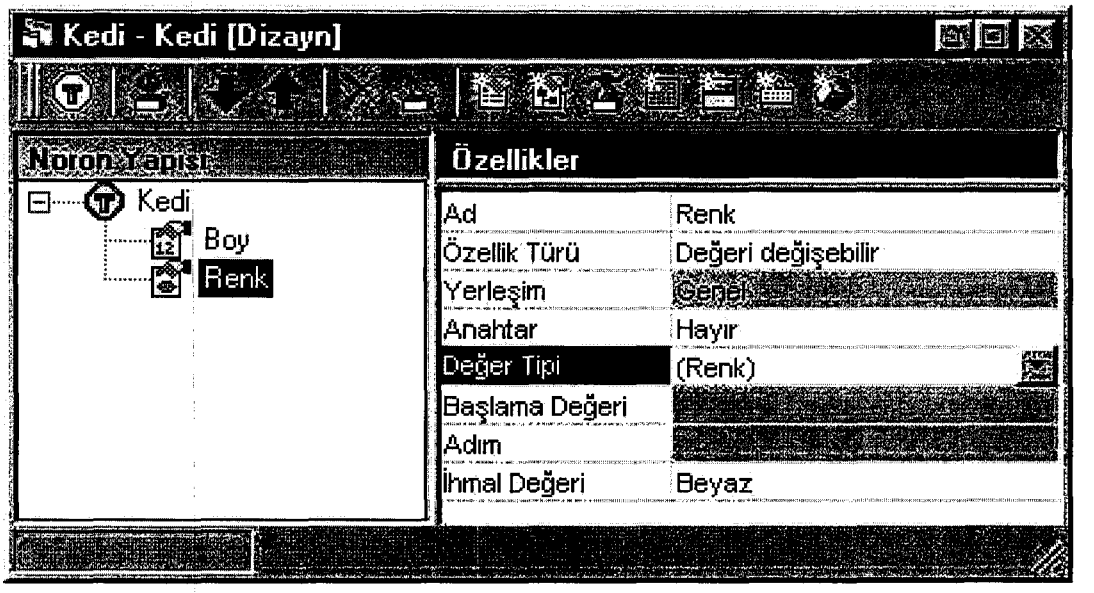
ABS içindeki tüm tanımlama arayüzlerinde mantıksal veri tutarlılığının sağlanması yer yer otomatik yer yer de kullanıcının hata mesajları ile uyarılması ile sağlanır. Nesnel tanımlamalar kendi içlerinde tutarlılık açısından otomatik olarak denetlenirken, nesnel tanımlamaların bir sonucu olarak oluşan yapıya, kriter ve etkiler olarak yapılan tanımlamaların uyum sağlaması gerekir. Baştan itibaren sonuna dek biçimsel ve anlamsal (syntax and semantic) bakımdan hatasız olarak tanımlanan bir problem için veri tutarlılığının elbette bir anlamı yoktur. Fakat kullanıcı daima hata, düzeltme, yeniden düzenleme, değişiklik yapma ve silme

yetisine sahiptir. Bu durumda tanımlamaların bir bölümü diğer bir bölümü ile uyum sağlamazsa doğal olarak ABS anlamlı bir arama algoritması oluşturamaz.

Veri tutarlılığının yeniden kontrol edilmesini gerektiren durumlar şunlardır :

- Yeni nesne tanımlanması
- Bir nesnenin özellik değerlerinde değişiklik yapılması
- Bir nesnenin silinmesi

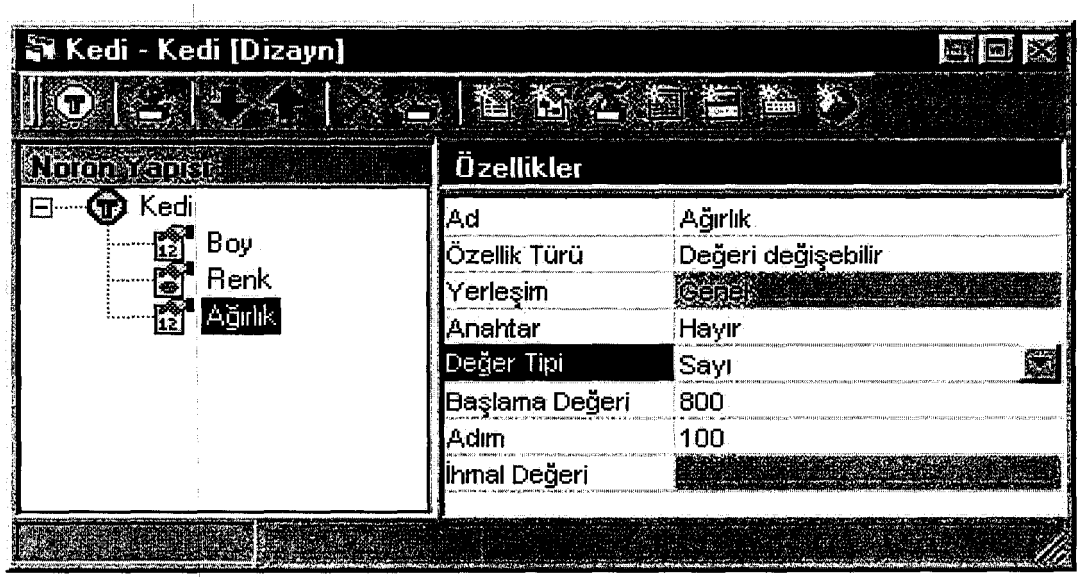
Yeni bir Tanım nesnesinin oluşturulması veri tutarlılığı açısından düzenleme işlemleri yapılmasını gerektirmez. Çünkü Tanım nesneleri başlangıçta bağımsız olarak oluşturulurlar. Küme nesneleri Tanım nesneleri ile birlikte otomatik olarak oluşturulduklarından onlar için de durum aynıdır. Bağlı Küme nesneleri ise ilk oluşturulduklarında hangi kümenin bağlandığı henüz tanımlanmadığından dolayı işlem gerektirmezler. Özellik nesneleri ise ilk tanımlandıklarından itibaren eğer varsa önceden türetilmiş Eleman nesneleri ile senkronize edilmelidirler. Bu amaçla Tanım nesnesinden türetilmiş tüm Eleman nesnelerinin Dinamik Eleman Özellikleri yapısına yeni Özellik nesnesinin gerektirdiği yapı ilave edilerek Özellik nesnesinde tanımlanan Başlangıç Değeri ve İhmal Değerine göre değer atamaları otomatik olarak yapılır. Aşağıda verilen örnekte bu otomatik düzenleme gösterilmiştir.



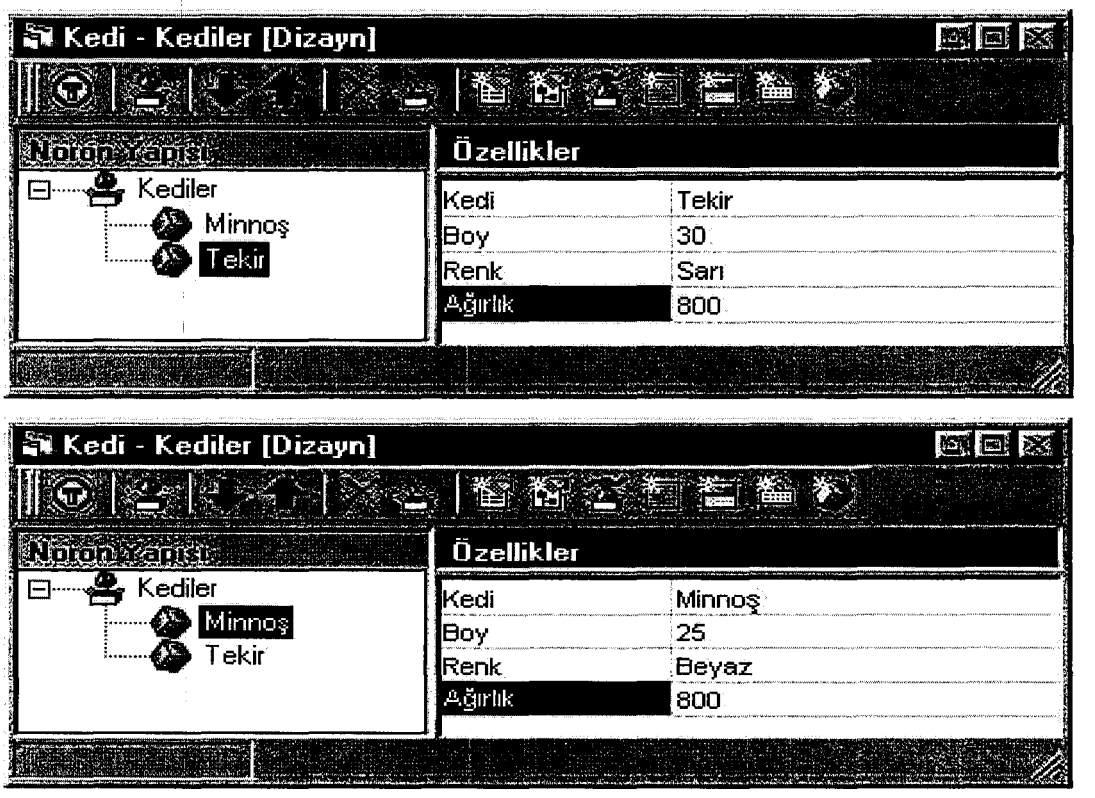
Şekil 2.23 Boy ve Renk Özellikleri ile "Kedi" Tanım nesnesi

Şekil 2.24 "Kedi" Tanım nesnesinden türetilmiş Eleman nesneleri

"Kedi" tanım nesnesine "Ağırlık" Özellik nesnesi ilave edildiğinde gerçek nesneler olan "Tekir" ve "Minnoş" Eleman nesnelerinin yapısına da "Ağırlık özelliği Başlama Değerinden veri alınarak değeri 800 olacak şekilde otomatik olarak yerleştirilir.



Şekil 2.25 "Kedi" Tanım nesnesine Ağırlık Özellik nesnesi ekleniyor



Şekil 2.26 Elemanlarda oluşan otomatik düzenleme

Yukarıdaki düzenlemeler yapılırken yeni oluşturulan her Özellik nesnesinin görülebilir olmayan sistem tanımlı "RefEleman" özelliği içine Küme nesnesindeki her Eleman nesnesinin ilgili eleman özelliğini gösteren işaretçiler yerleştirilir. Kısacası bir türetilmiş gerçek nesnenin Tanım nesnesi ile arasında sıkı bir bağ kurulur. Bu bağ yine türetilen her yeni Eleman nesnesi için de otomatik olarak yapılandırılır.

Anlaşılacağı üzere yeni nesnelerin tanımlanması aslında veri tutarlılığı açısından karmaşık işlemleri gerektirmezken tanımlı bir nesnenin özelliklerinin değiştirilmesi yada silinmesi sözkonusu olduğunda durum oldukça karmaşıklaşacaktır. Kriter ve etkiler aslında birer sistem tanımlı özellik olarak değerlendirildiğinden bunların tanımlanması nesnenin özellik değerlerinde değişiklik yapılması kapsamında açıklanacaktır.

Bir nesnenin özellik değerlerinde değişiklik yapılması problem için yapılan tüm tanımları derinden etkileyebilir. Çünkü nesnel tanımlamalar kendi içlerinde tutarlılığı gerektirirken kriter ve etki tanımlamaları da bu nesnel yapıya göre

doğrudur. Nesnel yapıdaki değişiklik aynı anda birçok kriter ve etki tanımını bir anda anlamsız, hatalı tanımlara dönüştürebilir.

Bir Bağlı Küme nesnesinin Bağlantı Kümesi özelliğindeki değerin değiştirilmesi doğrudan Tanım Zinciri'nin yapısını değiştireceğinden bu işlemin tüm problem tanımları içindeki etkilerinin değerlendirilmesi gerekmektedir. Kullanıcının böyle bir işlem ile ne yapmaya çalıştığı önceden tahmin edilemeyeceğinden yapı otomatik olarak yeni duruma adapte edilemez ve oluşan hataların düzeltilmesi hata mesajları ile uyarılarak yine kullanıcının kendisine bırakılır. Nesnel tanımlamaların kendi içlerindeki tutarlılığı açısından yapılan bir değişiklik mevcut verilerin tipleri ile uyumsuzluk yarattığında mevcut veri uyarı verilmeksizin silinir ve yerine uygun veri Özellik nesnesinin Başlangıç Değeri yada İhmal Değerinden alınarak eski uyumsuz veri yerine kullanılır[1]. Kullanıcıya uyarı verilmemesinin altıda yatan mantık ise eninde sonunda bu değişkliğin kullanıcı tarafından da yapılacak olmasıdır. Dolayısı ile bu işlem veri kaybı olarak nitelendirilemez. Unutulmamalıdır ki anlamsal tutarlılık tam olarak sağlanmadan ABS problemin otomatik çözüm aşamalarına geçmeyecektir.

ABS içinde tanımlanan tüm nesnelerin kriter ve etki ifadelerine(eğer bu ifadeler içinde kendi adları geçiyorsa) "RefFormül" sistem özelliği içindeki işaretçiler aracılığı ile bağlantıları kurulur. Bu bağlantılar sayesinde yapısında değişiklik oluşan her nesne bünyesinde adının geçtiği ifadeye ulaşır ve ifadenin yeniden kontrol edilmesi için kriter ve etki tanımlama arayüzüne mesaj gönderir. Arayüz de eğer anlamını yitirmişse ifadeyi ve ifadenin kriter yada etki tanımını "hatalı" olarak imler.

Kriter ve etki tanımlama arayüzü anlamsal bütünlüğü yazım hataları, ulaşılabilirlik ve tip uyumsuzlukları yönünden kontrol eder. Bu amaçla kriter yada etkinin tüm verileri öncelikle aşagıdaki yapı ile ifade edilir.

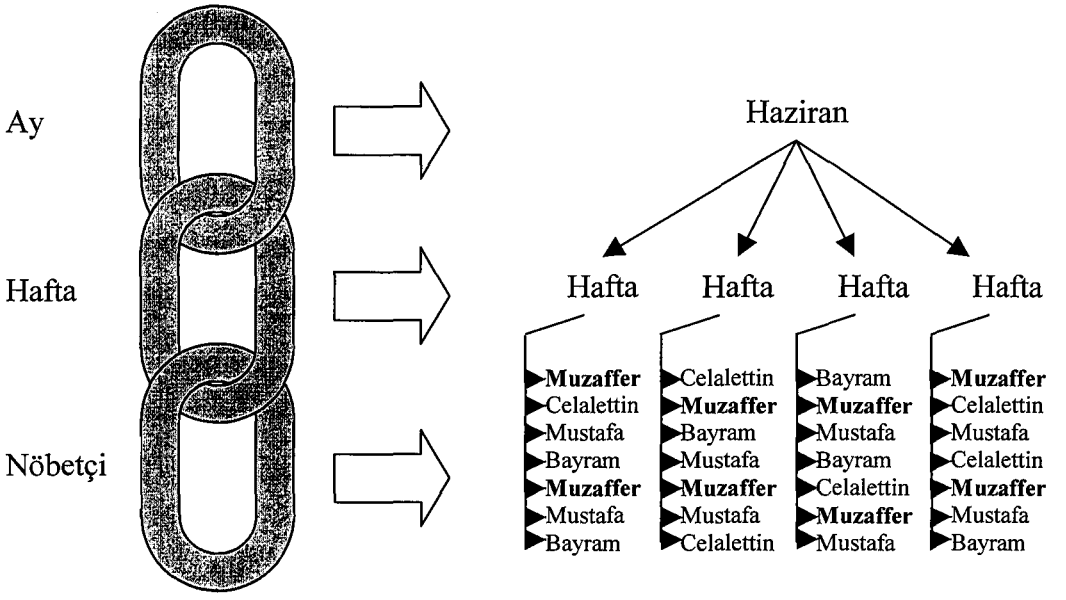
[1] A.B.S.'in çözüm getirmeye çalıştığı veri tutarlılığı problemine Microsoft Excel gibi tablolama programlarında da rastlanır. Başka hücrelerden aldığı verilerle bir formülü hesaplayan bir hücre veri aldığı hücrelerden birine uygun olmayan bir tipte veri yazıldığında hesaplama yapamaz ve hücrede sonuç değil hata mesajı görüntülenir[13].

```
Type formülT
   Ana_Text As String
   Amaç_Text As String
   Şart_Text As String
   Ana_Evaled As String
   Amaç_Evaled As String
   Şart_Evaled As String
   Ana_Hatasız As Boolean
   Amaç_Hatasız As Boolean
   Şart_Hatasız As Boolean
   Ana_Tip As Byte
   Amaç_Tip As Byte
   Şart_Tip As Byte
   Hatasız As Boolean
   Operatör As Long
   Önem As Long
   Açıklama As String
   Tür As Byte
   saltKüme As Long
End Type

Dim Formül As FormülT
```

Yukarıdaki yapı kriter ve etkiler için ortak kullanıldığından "Ana" Ana Parametre ve Etkilenecek Alan, "Amaç" ise Karşılaştırma Değeri ve Yeni Değer ifadeleri için kullanılır. Bazı durumlarda işlemin bir Küme nesnesinin tüm Eleman nesneleri üzerinde yapılacağını belirtilmesi amacıyla yalnızca bir Küme nesnesi olmak kaydıyla elemanlar yerine Küme adı kullanılabilir. Bu durumda bu Küme nesnesi "saltKüme" alanı kullanılarak işaret edilir. Bir kriter yada etki tanımının tüm verileri yukarıdaki formülT tanımından üretilmiş Formül değişkenine yerleştirilir ve kontrol işlemleri yapıldıktan sonra yine bu değişken içinde geri döner. Kontrol işlemi kullanıcının arayüz içinde bir başka kriter yada etki tanımına geçme komutu vermesiyle yada arayüzü kapatmasıyla tetiklenir. Hata yoksa komut uygulanır ancak hata varsa hata mesajı görüntülenir. Hata mesajının gelmesi durumunda kullanıcı hatayı düzeltmek yada tanımlamayı daha sonra düzeltmek üzere öylece bırakmak isteyebilir.

Kontrol yordamı "ifadeKontrol" adında bir alt yordamdır. "ifadeKontrol" anlamsal ve yazımsal kontrolu "eval" adında bir başka alt yordama yaptırırken tip uyumsuzluğu ve eksik tanımlamaların kontrolunu kendisi yapar. Dolayısıyla tip uyumsuzluğu ve eksik tanımlamalar ile ilgili hata mesajlarını bünyesinde barındırırken diğer uyumsuzluklar ile ilgili hata mesajlarını metnini "eval" alt yordamından alarak görüntüler.

Her ne kadar problemin genel görüntüsü Tanım Zinciri olarak ifade edilen yapı ile tanımlanıyor olsa da her Tanım nesnesinin kümeleri içinde türetilmiş birden fazla Eleman nesnesi olabileceğinden aslında yapı bir ağaç yapısıdır. Önceki bölümlerde belirtildiği gibi ağaç yapısı içinde gerçek nesneler tekil(unic) değildirler bu yüzden tanım ağacının büyüklüğü sonsuzdur. Çözüm de zaten bu sonsuz ağaç yapısının özelleşmiş, sonlu bir şekli olarak bulunur. Kriter ve etkiler bu ağaç yapısı içinde yer alan nesnelere tanımlandığından nesnenin yapı içinde bulunduğu yer itibarı ile, nerelere ulaşılıp nerelere ulaşılamayacağı bir tanımlamanın anlam yönünden kontrol edilmesinde temel çıkış noktasıdır. Ağaç yapısında bir daldan köke doğru bakıldığında tüm nesneler belirlidir. Yol takip edilerek seçim yapmak zorunluluğu ile karşılaşılmadan köke ulaşılabilir. Ancak kökten dallara doğru giden bir yolda daima seçimler söz konusudur.



Şekil 2.27 Tanım Zinciri'nin ağaç yapısına dönüşümü

Şekil 2.27 de görülen 8 adet Muzaffer nesnesi aslında aynıdır. Çünkü gerçek dünyada problemimize konu olan sadece bir tane Muzaffer vardır ve Nöbetçiler Kümesi içine de Nöbetçi Tanımından sadece bir tane türetilmiştir. Gerçekte aynı olmalarına rağmen şekildeki tüm Muzaffer nesneleri anlam yönünden birbirlerinden farklıdırlar. Durum Hafta nesnesi için de aynıdır. Bir adet olan Hafta

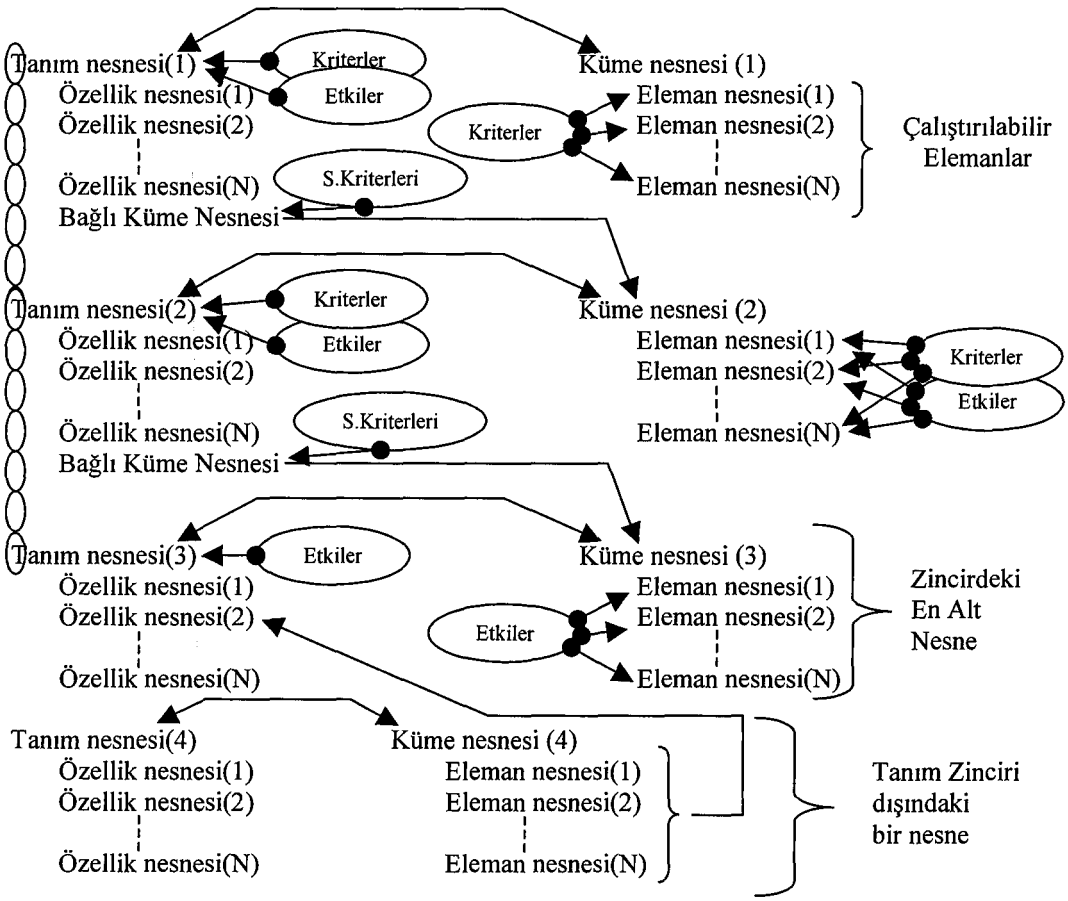
gerçek nesnesi planda 4 kez farklı anlamlarda kullanılmıştır. Hafta nesnesi için tanımlanmış bir "No" Özellik nesnesi olduğu varsayılırsa tanımlamalarda bu özellik için sadece tek bir değer girilebilirken planın oluşturulması esnasında bir etki tanımı ile şekildeki 4 hafta nesnesinin No özelliklerine 1 den 4 e kadar olan değerler atanabilir. Bu durumda "Muzaffer.Hafta.No" şeklindeki bir ifade ilk hafta içindeki bir Muzaffer için 1, ikinci hafta içindeki bir Muzaffer için de 2 değerini döndürecektir. Aynı ifadenin farklı değerler döndürmesi için bir parametreye daha ihtiyaç vardır ki bu parametre de ağaç içinde o an bulunulan konumdur. Kriter ve etki tanımlamalarında başlangıç noktası(o an bulunulan konum) her bir tanımlama için Tanım Zinciri'ndeki bir halka olarak belirlidir. Bu verilerden yola çıkılarak "eval" fonksiyonu içinde bazı Özellik nesnelerinin anlamsal olarak üst nesnelere taşınabilmesinin getirdiği karmaşıklık da dahil olmak üzere hiçbir karışıklığa izin verilmeden nesnel tanımlamanın anlamı algılanır ve kullanıcının hata yapıp yapmadığı kontrol edilir. Eğer hiçbir hata bulunamazsa hatasız tanımlama içindeki nesne adları işaretçilere dönüştürülerek saklanır. Tüm nesnelere adları ile değil bağlı liste elemanlarının indisleri ile ulaşıldığından proje içinde nesne adlarının değiştirilmesinin veri tutarlılığına bir etkisi olmaz.

2.2.1.3 Tanımlanmış bir problemin genel görünümü

ABS içinde bir problem;

- Yapılandırılmış sistem tanımlı nesneler
- Tanım Zinciri
- Bu nesneler içinde tanımlanmış kriter ve etkiler

öğelerinden oluşmaktadır. Aşağıdaki şekilde bu öğelerin ilişkileri üzerinde durularak bir problem tanımının genel olarak ABS içinde nasıl göründüğü açıklanmaya çalışılmıştır.



Şekil 2.28 Genel mantıksal görünüm

2.2.2 Yürütme bölümü

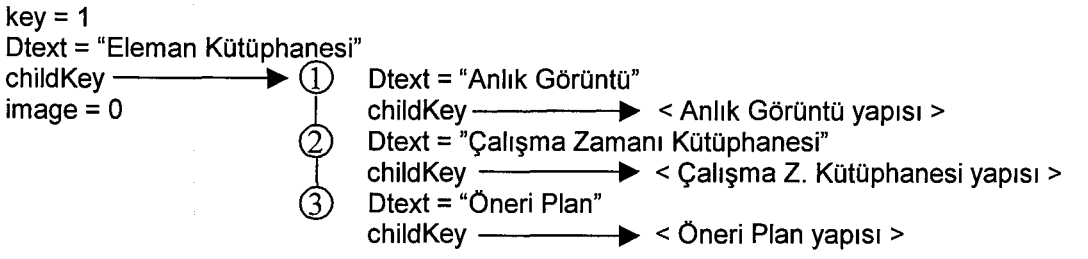
Yürütme bölümündeki işlemleri genel olarak özetlemek gerekirse : Tanımlama bölümünde tanımlanmış bir problem, tüm öğe ve tanımlamaları (eğer anlamsal tutarlılığa sahipse) ile Tanım Zinciri'nin en üst nesnesinin türetilen her Eleman nesnesine özel bir formatta yüklenir ve çözümü, yürütme bölümünde otomatik olarak ABS'in oluşturacağı bir arama algoritması ile yine bu Eleman nesneleri üzerinde aranır.

Yapılan tanımlamalardan problemi çözmek üzere özelleşmiş bir arama algoritmasının çıkartılması birkaç katmanlı bir ön hazırlık aşamasını gerektirmektedir. Bu amaçla öncelikle problem tanımının tüm anlamlı özelliklerini içeren bir Anlık Görüntü(snapshot) alınır. Daha sonra bu Anlık Görüntü'den Çalışma Zamanı Kütüphanesi oluşturulur. Anlık Görüntü, Çalışma Zamanı

Kütüphanesi ile birlikte Eleman nesnesinden ulaşılabileck şekilde kayıt ortamında dosyalanır. Daha sonra yürütme "Form_Düşün" adlı bir forma devredilir. Form_Düşün içinde Çalışma Zamanı Kütüphanesi'nden alınan veriler ile Arama Algoritmasının Hazırlık Veri Yapıları oluşturulur.

Bu aşamada arama algoritmasının yapılandırılması için gerekli tüm veriler uygun formatta hazırlanmıştır. Son olarak da yine Form_Düşün içindeki "aramaMotoru" alt yordamı çalıştırılır. "aramaMotoru" da işlemin durdurlması yada duraklatılması gerektiğine dair bir mesaj gelene dek peşpeşe "openNode" adlı bir başka alt yordamı çalıştırarak probleme tanımlamalara uygun bir çözüm arar. Bulunan çözüm de "Öneri Plan" olarak adlandırılır.

Anlık Görüntü, Çalışma Zamanı Kütüphanesi ve Öneri Plan(üçü birlikte bir "Eleman Kütüphanesi" olarak); ABS içinde tıpkı problem tanımının saklandığı gibi bağlı-liste formatında fakat tanımlamalara ait bağlı listeden yalıtılmış ikinci bir bağlı-liste yapısı olarak oluşturulur ve tanımlamalar dosyasından ayrı olarak ".jpl" uzantılı dosyalar halinde kayıt ortamına aktarılırlar. En üst Tanım nesnesine ait Eleman nesneleri(Tepe Eleman) sadece kendilerine ait jpl dosyasını gösteren bir işaretçi barındırırlar fazlasını değil. Çünkü ABS de bir proje büyük çoğunlukla yeni bir plan üretmek için açılacak, eskiden üzerinde çalışılmış Tepe Eleman nesnelerinin tümünün Eleman Kütüphane'lerinin yüklenmesi zaman alıcı, hafızayı fazla miktarda işgal eden gereksiz bir işlem olacaktır.



Şekil 2.29 Eleman Kütüphanesi bağlı-liste yapısı

2.2.2.1 Anlık Görüntü'nün alınması

ABS içinde Anlık Görüntü alma işleminin etik bir anlamı vardır. Planlama işlemi daima belirli kurallara göre yürütülür ve daha sonra da üretilen plan gerçek hayatta uygulanır. Planlama ve uygulama aşamalarında geçerli olan kurallara göre

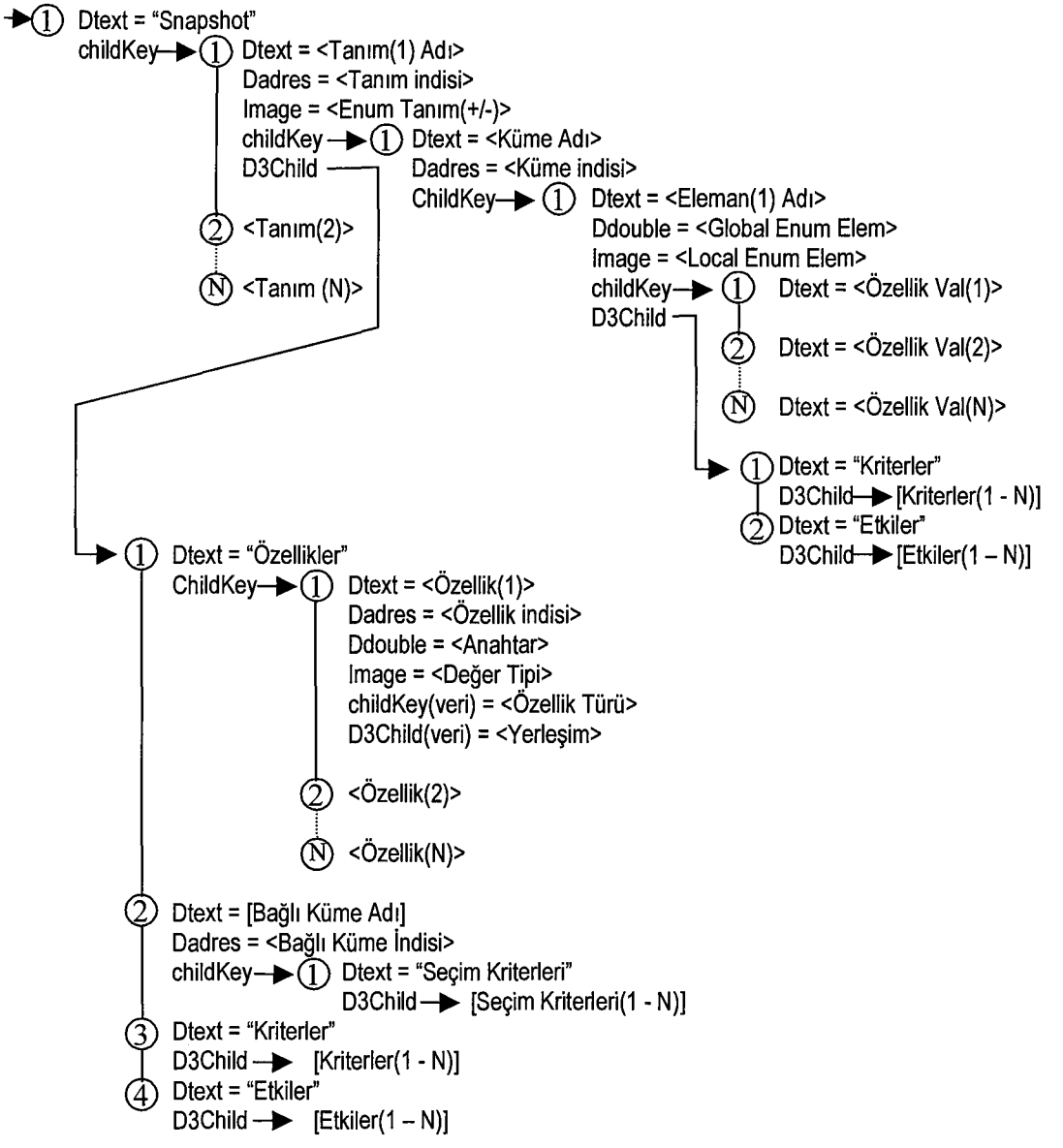
üretilmiş ve uygulaması çoktan bitmiş bir plan için, kurallarda daha sonra yapılmış değişikliklerden doğan uyumsuzluklardan planı yapan kişi sorumlu tutulamaz. Ancak kural değişikliklerinden sonra oluşmuş yeni kuralların geçerli olduğu bir dünyada, uygulanmış ve işi bitmiş dahi olsa, eski planların(eğer yeni kurallara göre değerlendirilirlerse) hatalı görünecekleri de bir gerçektir. Bu durumda eski planların derhal yok edilmeleri gerekir. Oysa geçmiş, geleceğin daha iyi olabilmesi için dersler içerdiğinden kıymetlidir ve bozulmadan saklanması gerekir. Bu durumda geçmişi o zamanki kurallar ile beraber saklamak ve gerektiğinde yine o kurallara göre irdelemek en makul yaklaşım tarzı olacaktır. İşte Anlık Görüntü alınmasının altında yatan etik budur.

Anlık Görüntü alma işlemi ABS içindeTanım Zinciri'nin en üst halkasını oluşturan bir Tanım nesnesinden kümesi içine bir Eleman nesnesi(Tepe Eleman) türetilmesi, veya daha önceden türetilmiş bir Tepe Eleman nesnesine "Düzenle" komutu verilmesi ile başlatılır. Her Tepe Eleman nesnesi türetildiği yada düzenlendiği anda mevcut olan problem tanımlarını içermeli, daha sonra tanımlamalar üzerinde yapılan değişikliklerden etkilenmemelidir. Kural değişiklikleri bazen planın üretilmiş fakat henüz gerçek dünyada uygulanmamış olduğu bir ara zamanda da gerçekleşebilir. Bu sebepten kullanıcıya, Tepe Eleman nesnesini, üretilmiş planın yok olması pahasına da olsa[1], yeni kurallara göre düzenleyebilme imkanı verilmiştir.

Anlık Görüntü'nün yapısı tanımlamalar bölümündeki yapıya büyük ölçüde benzerlikler gösterir. Fakat tanımlamalar bölümünde veri tutarlılığının sağlanmasına yardımcı olması için nesnelere iliştirilmiş işaretçiler ve işaretçi listeleri zaten görevlerini tanımlama bölümünde yapmış olduklarından Anlık Görüntü yapısı içine alınmaz. Anlık Görüntü yapısı içinde Tanım nesneleri Tanım Zinciri'ndeki sıraları göz önüne alınarak yerleştirilir, Tanım Zinciri'ne dahil olmayan Tanım nesneleri ise sıralamada en sona atılır. Tanımlama bölümünde kök nesneler olarak saklanan Küme nesneleri Anlık Görüntü yapısında kök değil Tanım nesnelerine bağlı yapılar olarak saklanır. Eleman Kütüphanesi Tepe Eleman nesnelerinden her biri için ayrı ayrı oluşturulan bir yapı olduğundan bu yapının bir

[1] Düzenleme işleminden sonra, mevcut planın yok edilmesi yerine, yeni kurallar ile uyumsuz olduğu noktaların kullanıcıya gösterilmesi daha uygun bir tarz olabilir. Ancak bu işlem gelecekte gerçekleştirilmek üzere şimdilik kapsam dışında tutulmuştur.

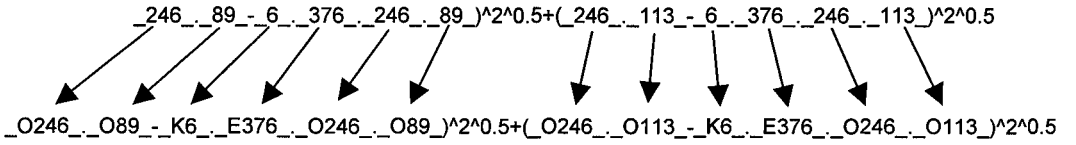
parçası olan Anlık Görüntü içinde Eleman kütüphanesinin sahibi olan Tepe Eleman nesnesinden başka diğer Tepe Eleman nesnelerinin verilerinin bulunması gereksizdir. Bu yüzden Anlık Görüntü yapısı içinde Tanım Zinciri'nin en üst halkasını oluşturan Tanım nesnesinin kümesinde daima tek Eleman nesnesi bulunur. Diğer nesneler ise tanımlama bölümündeki hiyerarşi aynen korunarak yerleştirilir.



Şekil 2.30 Anlık Görüntünün bağlı-liste yapısı

Anlık Görüntü yapısı içinde Çalışma Zamanı Kütüphanesi'nin oluşturulmasına yardımcı olması açısından Tanım nesneleri ve Eleman nesneleri "1" den başlayarak numaralandırılır. Numaralandırmada sadece Tanım nesneleri için geçerli olmak üzere Tanım Zinciri dışında kalan Tanım nesnelerinin numaraları negatif işaretli olarak saklanır. Her Tanım nesnesinin kümesinde birden fazla Eleman nesnesi bulunabileceğinden Eleman nesneleri Tanım nesnesi içinde yerel(local enum elem), Anlık Görüntü yapısı içinde ise genel(global enum elem) olmak üzere iki ayrı şekilde numaralandırılırlar.

Anlık Görüntü yapısı içinde uygulanan işlemlerden bir diğeri de Çalışma Zamanı Kütüphanesinin hazırlanmasına kolaylık sağlaması için kriter ve etki tanımlamalarının saklanma yapısına Eleman, Küme ve Özellik nesnelerini sırasıyla ifade eden "E", "K" ve "O" karakterlerinin ilave edilmesidir.



Anlık Görüntü alınırken uygulanan son işlem ise tüm Eleman nesnelerine uygulanması istenerek Küme nesnesinin adıyla(Salt Küme) yapılmış kriter tanımlarının Eleman nesnelerine dağıtılması işlemidir. Örneğin bu işlemde : "Kediler.Boy" gibi bir tanımlama Kediler.Tekir.Boy, Kediler.Minnoş.Boy, ...vb (Kediler kümesinin tüm Eleman nesneleri için birer tane) şeklinde sanki tüm elemanlar için ayrı ayrı tanımlanmış kriterlerden oluşan bir ifadeler gurubu gibi dağıtık hale getirilir.

Anlık Görüntü yapısına Önem derecesi sıfır olarak tanımlanmış kriterler ile Görünmez türde tanımlanmış Özellik nesneleri yürütmede hiç bir fonksiyonları olmayacağından alınmazlar.

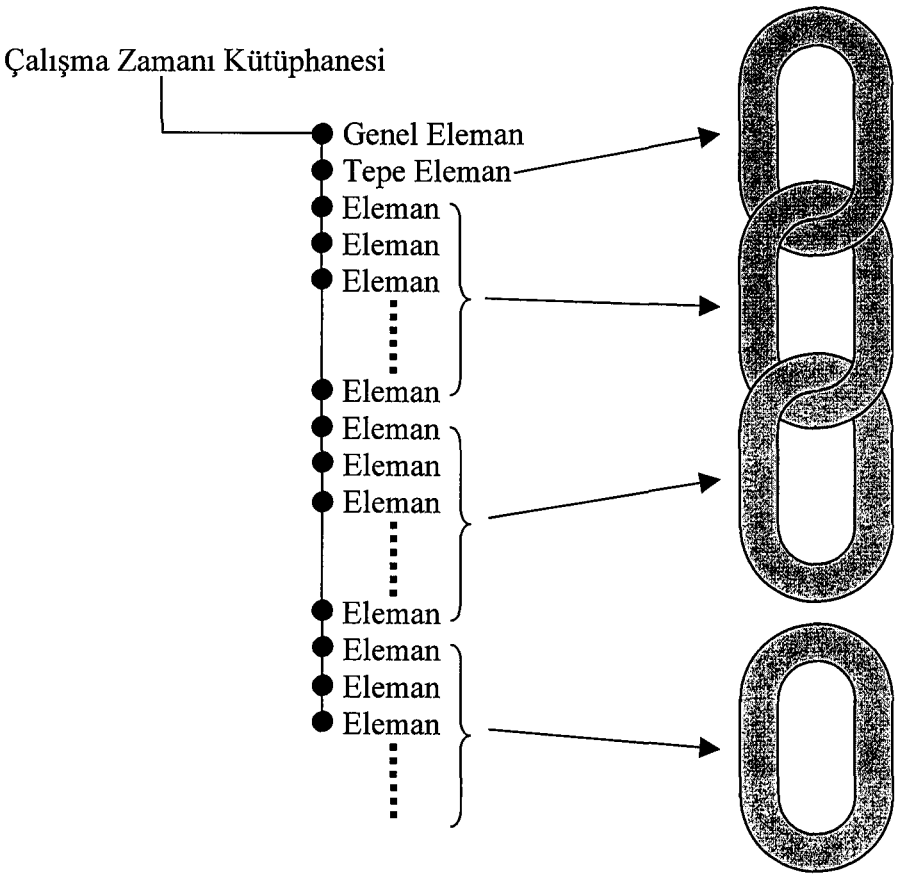
2.2.2.2 Çalışma Zamanı Kütüphanesi hazırlanması

Algoritmaya hazırlık aşamasının Anlık Görüntü'nün alınmasından sonraki ikinci adımı Çalışma Zamanı Kütüphanesi'nin oluşturulmasıdır. Çalışma Zamanı Kütüphanesi'nin oluşturulması aşamasında artık tek veri kaynağı Anlık Görüntü yapısıdır, tanımlama bölümündeki verilere hiç başvurulmaz.

Çalışma Zamanı Kütüphanesi arama algoritmasının performansını arttırmak ve hafıza harcamasını minimum düzeye indirgemek maksadı ile gereksiz verilerin filtre edildiği, veri ulaşım yollarının optimize edildiği, yerleşimli özelliklerin yerlerine yerleştirildiği ve algoritmanın tüm işlemleri hiçbir işlem tekrarı yapmaksızın, en sade biçimde gerçekleştirebilmesi için çalışma aşamalarında değişen verilerle değişmeyen verilerin düzenlendiği önemli bir ara aşamadır.

Çalışma Zamanı Kütüphanesi'nin yapısı Anlık Görüntünün aksine tanımlamalar bölümündeki yapıyı bire bir yansıtmaz. Çalışma Zamanı Kütüphanesi'nde Tanım nesneleri dışlanarak sadece türetilmiş gerçek nesneler olan Eleman nesneleri üzerine bir yapı oluşturulur.

Çalışma Zamanı Kütüphanesi'ndeki ilk eleman tanımlamalarda hiç tanımlanmamış hayali bir elemandır. Bu elemanın, çözüm ağacında en tepede duran ve yerleşimi "Genel" olarak tanımlanmış özellikleri içeren gizli bir eleman(Genel Eleman) olduğu varsayılır. Genel Eleman'ın kendine ait özellikleri, kriter ve etkileri yoktur. Sadece ağaçtaki alt nesnelerden gelen özellikleri vardır. İkinci eleman ise daima Otomatik Düşün komutunun verileceği, Eleman Kütüphanesi'nin de sahibi olan Tepe Eleman nesnesidir. Açıklanan bu iki eleman Çalışma Zamanı Kütüphanesi'nin değişmez yapısını oluşturur. Geriye kalan elemanlar ise öncelikle Tanım Zinciri içindeki ve daha sonra da dışındaki tüm Tanım nesnelerinin türetilmiş Eleman nesneleridir. Tanım nesneleri her ne kadar yapıda var olmasa da Genel Eleman ile Tanım Zinciri dışındaki Tanım nesnelerinin elemanları arasında kalan elemanlar sıra itibarı ile Tanım Zinciri'nin yapısını aynen yansıtır ve bu sıra algoritma için çok önemlidir. Çünkü Çalışma Zamanı Kütüphanesi'nde bağlantı noktalarının tanımlandığı Bağlı Küme nesneleri bulunmaz.



Şekil 2.31 Çalışma Zamanı Kütüphanesi genel görünümü

Çalışma Zamanı Kütüphanesi'nde her bir eleman için yapılan temel işlemler :

- Özellik değerlerinin düzenlenmesi
- Alt grupların yerleştirilmesi
- Kriter ve etki tanımlarının düzenlenmesi

işlemleridir.

Özellik değerlerinin düzenlenmesi yerleşimli özelliklerin yerlerine dağıtımını, değeri sabit ve değeri değişebilir özelliklerin birbirlerinden ayrılmasını, bu özelliklerden her biri için "ulaşım indisler dizisi" nin oluşturulmasını kapsar. Ulaşım indisler dizisi daha sonra kriter ve etki tanımlarının düzenlenmesi aşamasında kullanılacak bir düzenlemedir. Çalışma Zamanı Kütüphanesi'nde bir

eleman için iki tür özellik alanı oluşturulur. Bu özellik değeri alanları da kendi içlerinde iki guruba ayrılırlar.

- Sabit özellikler
  - Bir özellik olarak elemanın kendisi
  - Diğer sabit özellikler
- Değişebilir özellikler
  - Elemanın kendi değişebilir özellikleri
  - Alt nesnelerden gelen özelikler.

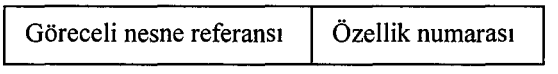
Çalışma Zamanı Kütüphanesi'nde sabit ve değişebilir değerli özellikler farklı dallarda yer alırlar. Hatırlanacağı üzere bir özelliğin değer tipi bir başka gerçek nesne olabilmektedir. Dolayısıyla bu durumun da ifade edilebilmesi için her elemanın ilk sabit özelliği bir özellik olarak kendisidir. Sabit değerli özelliklere tanımlama bölümünde zaten yerleşim tanımlanması engellendiğinden alt nesnelerden gelen yerleşimli özelliklerin de değeri değişebilir tipte olduğu garanti edilmiştir. Bu özellikler de değişebilir özellikler yapısı içinde gösterilirler. Elemanların yukarıda bahsedilen özellik değerlerinin yanına her bir özellik değeri için o özelliğin bir tanıtıcı imlemesi(indis dizesi) yerleştirilir. Kriter ve etki tanımlamalarının matematiksel notasyonların dışında kalan kısmında daima nesne ve özelliklerine çağırımlar bulunduğundan bu çağırımlar özelliklerin yanına yerleştirilen indis dizeleri ile eşleştirilerek çözümlenecektir.

Çalışma Zamanı Kütüphanesi'nde veri tekrarı olarak nitelenebilecek tek unsur elemanların üst nesnelere taşınmak üzere yapılandırılmış olan özelliklerinin hem üst nesnede hem de elemanın kendi yapısında bulunmasıdır. Bu işlemin yerleşimli özelliklere ulaşmak sözkonusu olduğunda kaçınılmaz olduğu değerlendirilmiş olup arama algoritması içinde ilave yer tutmaması için gerekli düzenlemeler yapılmıştır. Yerleşimli özellik yerleşiminin tanımlandığı nesne içinde verisini tutarken

bağlı bulunduğu nesne içinde sadeece gerçek yerini gösteren bir göreceli işaretçi[1] barındırır.

"Alt Gurup" olarak tanımlanan yapı Tanım Zinciri'nin belirli bir halkasına dahil olan bir elemanın alt halka elemanlarının tümünden haberdar olmasını sağlayan ve bu amaçla içine alt halkanın elemanlarının yerleştirildiği bir bağlı liste elemanıdır. Alt Gurup yapısı bir işaretçiler listesi değildir. Çalışma Zamanı Kütüphanesi'nde tüm elemanlar bir sıraya bağlı kalınarak numaralandırılmış olduklarından Alt Gurup yapısında sadece başlangıç ve bitiş elemanlarının numaralarının saklanması yeterlidir.

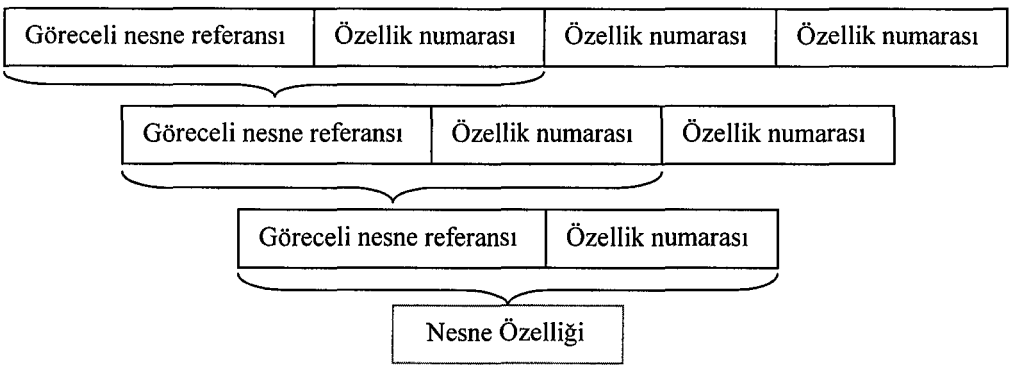
Çalışma Zamanı Kütüphanesi'nin son yapı dalını kriter ve etkiler oluştururlar. Kriter ve etki ifadelerinin matematiksel notasyonlar ile Tanım, Küme, Eleman ve Özellik nesnelerinin indislerini gösteren işaretçilerden oluştuğunu hatırlayınız. Bu noktaya gelinene dek ifadelerin bu yapısı daima korunmuştur. Ancak arama algoritması bir ağaç yapısı üzerinde çalışacağından bu belirli işaretçiler kullanılamaz. Çünkü ağaç yapısı içinde bir nesne aynı anda birçok yerde bulunabilir fakat aynı kalan kriter ve etki ifadelerinin işaretçileri(nesnelerin ağaç yapısı içinde bulundukları yere göre) anlamları değişse de hep aynı yeri gösterirler. Bu yüzden kriter ve etki ifadelerinin işaretçileri bu aşamada göreceli işaretçilere dönüştürülür.



Şekil 2.32 Göreceli İşaretçi genel yapısı

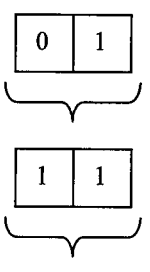
[1] Göreceli işaretçiler işaret ettikleri adresleri kendi bulundukları yere göre gösteren işaretçilerdir. Bu anlamda aynı değere sahip iki normal işaretçi hafızada nerede olurlarsa olsunlar aynı adresi işaret ederlerken, göreceli işaretçiler eğer farklı hafıza alanlarında iseler, aynı değeri taşısalar bile farklı adresleri gösterirler.[9] Göreceli işaretçiler bilinen işaretçilerin aksine negatif değerler taşıyabilmekte ancak bu anlamda akla gelen ilk fikirdeki gibi negatif değerler yönü değil, Tanım Zinciri içindeki belirli bir halkayı işaret ederler. A.B.S.'in arama algoritmasındaki adresleme işlemleri düzlemsel(lineer) bir hafıza alanı üzerinde değil bir ağaç yapısı üzerinde değerlendirilmektedir. Bu yapı üzerine göreceli işaretçilerin aldıkları pozitif değerler dallardan köke doğru olan yönde değerlendirilir. Kökten dallara doğru göreceli bir işaretleme belirsizlikten dolayı mümkün olmadığından negatif değerli göreceli işaretleme aslında göreceli değildir ve direk olarak bir gerçek nesneyi Tanım Zinciri'ndeki yeri itibarı ile gösterir. A.B.S. in arama algoritmasının yapısı bütünüyle bu göreceli işaretçilere dayandığından açıklamaların bundan sonraki kısmında "göreceli işaretçi" deyimi ile sıklıkla karşılaşılacaktır.

Bu noktada göreceli işaretçilerin yapısına bir miktar değinmenin faydalı olacağı değerlendirilmiştir. Göreceli işaretçiler iki temel veri tutan bir dizi formatındadır. Bu verilerden ilki göreceli nesne referansı diğeri ise özellik numarasıdır. Bu ikili beraberce bir nesnenin özelliklerinden birinin değerini yada bir başka göreceli nesne referansını işaret edebilir. Bu yüzden bir göreceli işaretçi birbiri içine kaskatlanmış daha uzun bir dizi olarak da tanımlanabilir.



Şekil 2.33 Göreceli İşaretçi fonksiyonel yapısı

Yine Ay-Hafta-Nöbetçi örneği üzerinde açıklamalar yapmak göreceli işaretcileri daha anlaşılır bir şekilde ifade edecektir. Şekil 2.27 de görülen ağaç yapısı içinde Nöbetçi nesnesinin üst nesne olan Hafta nesnesine taşınmış bir Haftalık Nöbet Sayısı özelliğinin de olduğunu hatırlayarak üçüncü Hafta nesnesinin altındaki bir Muzaffer nesnesi için Haftalık Nöbet Sayısına işaret eden bir göreceli işaretçi şöyle olacaktır :



Bu işaretçinin göreceli nesne referansı 0 olduğundan nesnenin kendisi işaret edilmekte ve özellik numarasından birinci özelliğinin değerine bakılmaktadır. Haftalık Nöbet Sayısı Hafta nesnesine yerleşimli bir özellik olduğundan değeri yine bir göreceli işaretçi olarak dönecektir. Dönen işaretçinin göreceli nesne referansı +1 dir ve bir üst nesneyi göstermektedir. Üst nesne ise Hafta(üçüncü) nesnesidir. Hafta nesnesinin birinci özellik değeri de "2" olarak geri döndürülür.

Göreceli işaretçiler, kriter yada etki ifadelerinden nesnelere çağırımlar yapan işaretçiler dizisi üzerinde bir takım işlemler yapıldıktan sonra oluşan yeni dizi yine Çalışma Zamanı Kütüphanesi'nin elemanlarının özellik değerleri içinde daha önceden hazırlanmış olan imlemeler arasında arama yapılarak oluşturulurlar.

Çalışma Zamanı Kütüphanesi'nde Küme ve Tanım nesneleri bulunmadığından kriter ve etki ifadelerinde dönüşümler yapmak gerekir. Dönüşüm :

Küme.Eleman.Özellik[.Özellik.Özellik.....]

şeklinde Küme nesnesi ile başlayan ifadeler için Küme işaretçisi silinerek,

Tanım.Özellik[.Özellik.Özellik.....]

şeklinde Tanım nesnesi ile başlayan ifadeler için Tanım nesnesi yerine o tanımdan türetilmiş ilk Eleman nesnesi konularak,

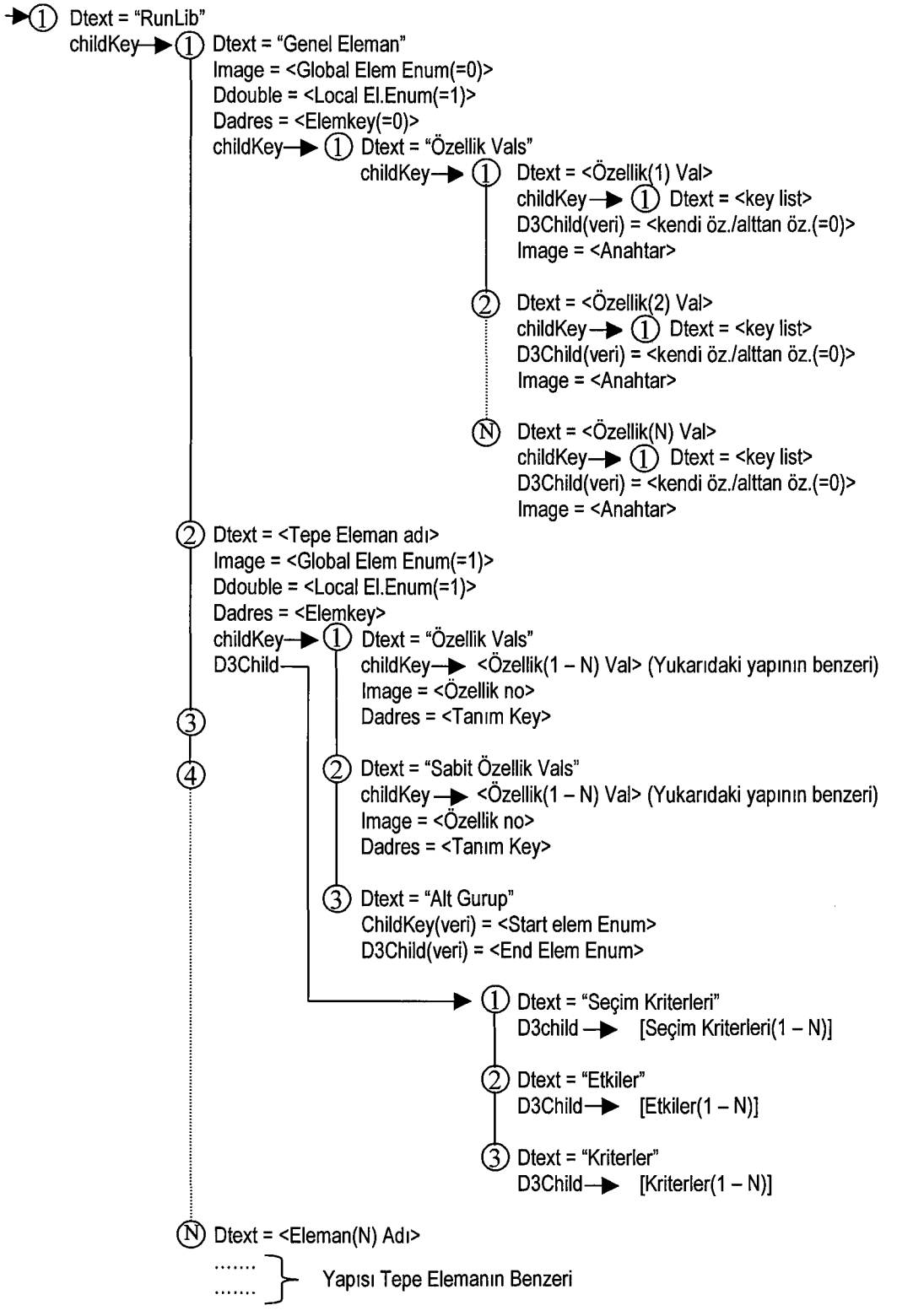
Özellik[.Özellik.Özellik.....]

şeklinde sadece Özellik nesnesi ile başlayan ifadeler için ise baş tarafına içinde bulunulan Eleman nesnesi konularak gerçekleştirilir ve bu dönüşüm sonucu anlam hiç bozulmadan tüm ifadeler ;

Eleman.Özellik[.Özellik.Özellik.....]

şeklinde Eleman nesnesi ile başlayan standart bir forma dönüştürülür. Çalışma Zamanı Kütüphanesi'ndeki elemanların da özellik imleri aynı formda olduğundan dönüşüm sonucu bulunan form bu imler arasında aranarak bulunan yerdeki eleman göreceli nesne referansı, özelliğin numarası ise özellik numarası olmak üzere göreceli işaretçi üretilir.

Kriter ve etki ifadeleri içinde elbette değeri sabit özelliklere de çağırımlar yapılmaktadır. Arama algoritması içine bu tip çağırımların aynen sokulması gereksiz işlemlerin defalarca yapılmasına sebep olacaktır. Bu sebeple benzer ifadeler bu aşamada değerleri hesaplanarak sadeleştirilirler. Örneğin bir kedinin boyu ve ağırlığı hayatı boyunca değişmesine rağmen, rengi daima aynı kalacaktır. Böyle bir örnekte kullanıcı "Kedi" Tanım nesnesinin "Renk" Özellik nesnesini "Değeri Sabit" olarak tanımlayacaktır. Bir kriter ifadesinde yer alan "Kediler.Minnoş.Renk" ifadesi eğer arama algoritmasına aynen sokulursa ifadenin geri dödüreceği değer olan "Beyaz" değeri ifade ile her karşılaşıldığında yeniden hesaplanacaktır. Oysa "Kediler.Minnoş.Renk" ifadesinin değerinin "Beyaz" olduğu ve değişmeyeceği garanti edilmiştir öyleyse ifade direk olarak "Beyaz" ifadesi ile yer değiştirilebilir.



Şekil 2.34 Çalışma Zamanı Kütüphanesinin bağlı-liste yapısı

2.2.2.3 Arama algoritmasına hazırlık veri yapıları

Arama algoritması içinde yapının bağlı-liste şeklinde olması (performans açısından) beklenmemelidir. Çalışma Zamanı Kütüphanesi'nden arama algoritması yapısına geçilirken mevcut veriler artık kayıt ortamına aktarılmayıp sadece RAM hafıza içinde işleneceğinden performansı da arttırmak maksadıyla bilinen veri yapılarına dönüştürülürler. Bu dönüşüm işlemi kolay sayılabilecek bir işlemdir çünkü Çalışma Zamanı Kütüphanesi'nin yapısı oluşturulurken bu aşama düşünülmüş, veriler birbirine benzeyen yapılara dönüştürülmüşlerdir[1].

Hazırlık veri yapılarının ilki Çalışma Zamanı Kütüphanesi'ndeki eleman nesnelerine karşılık gelen "DElemanT" dir. Arama algoritmasında elemanlar bir ağaç yapısı oluşturacaklarından yapı, kök tarafında bağlı bulundukları üst eleman nesnesine bir işaretçi olan "Parent" alanı barındırır. Elemanların dallara doğru olan yönde de bağlandıkları diğer elemanlar vardır. Bunlar her birine bir işaretçi tutan "childs" dizisi ile ve alt gurupta o an için eklenmeyi bekleyen sıradaki ilk eleman nesnesi de "adayChild" alanı ile işaret edilir. Elemanların değişebilir tipteki özelliklerinin değerleri yine bu yapı içindeki "özVals" dizi alanı ile eleman yapısı içinde saklanır.

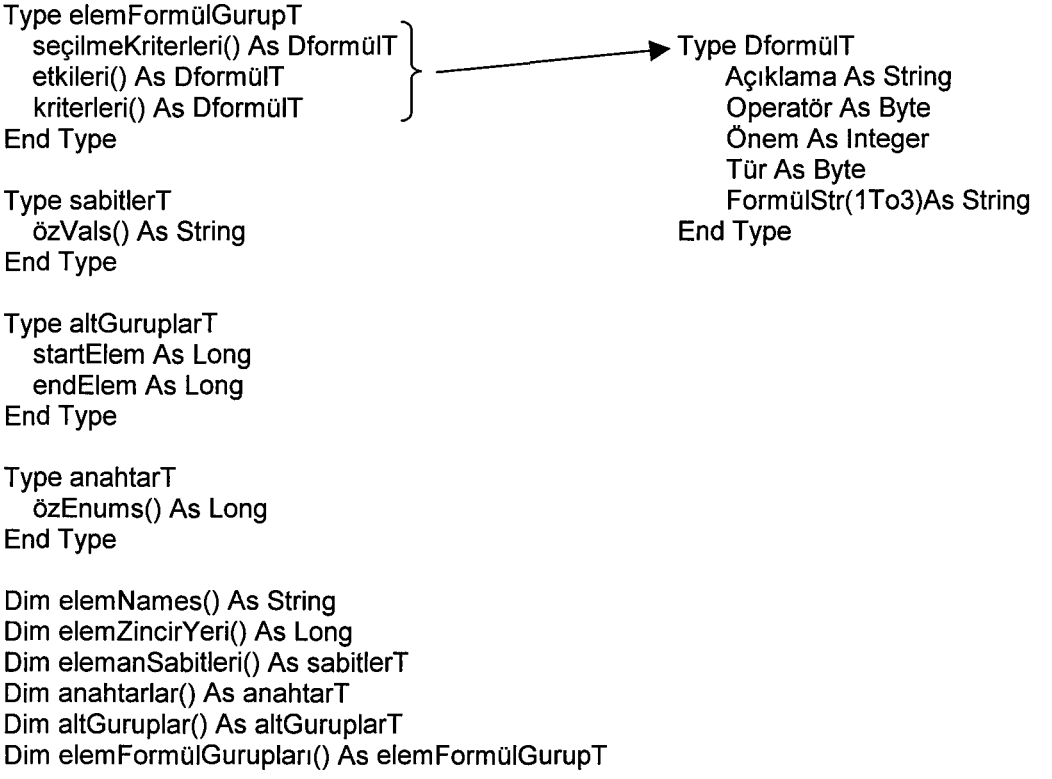
```
Type DElemanT
   orijin As Long
   Parent As Long
   özVals() As String
   childs() As Long
   adayChild As Long
End Type
```

Bu yapı arama algoritması içinde iki farklı yerde kullanılmaktadır. Bunlardan birincisi "iskeletElemanlar" dizi değişkenidir.

```
Dim iskeletElemanlar() As DElemanT
```

Şeklinde tanımlanan iskelet elemanlar dizisi içine Çalışma Zamanı Kütüphanesi ndeki tüm elemanlar, daha sonra durum yapısı içinde buradan kopyalanarak oluşturulmak üzere aktarılır. Çalışma Zamanı Kütüphanesi'nde bir eleman için tanımlanan bilgilerin hepsinin DElemanT yapısı içinde olmadığına dikkat

edilmelidir. DElemanT yapısının, arama ağacı içinde bilfiil bulunacak bir yapı olduğundan sade ve az hafıza harcayan bir yapı olmasına dikkat edilmiştir. Çalışma Zamanı Kütüphanesi'nde eleman için tanımlanan bilgilerin geri kalanı olan eleman adları, sabit özellik değerleri, anahtar özellikler, alt guruplar ve kriter/etki ifadeleri "orijin" alanı ile DElemanT yapısına ilişkilendirilmiş aşağıdaki yapılar ve değişkenleri ile saklanırlar.



Bu yapıların tümü dizi şeklindedir çünkü ilişkide oldukları elemanlar da dizi şeklindedir.

DElemanT yapısının bulunduğu ikinci yer ise çözüm aşamasında problemin durumlarını taşıyan "durumT" yapısıdır. Bu yapı da "durumlar" dizi değişkeninde kullanılır. Her durum içinde çözüm durumu olup olmadığı, tanımlanmış kriterlere göre ne derecede iyi bir durum olduğu, Anahtar özelliklerden oluşan bir özel tanımlayıcı ve durum elemanları bilgilerini taşır.

---

[1] Çalışma Zamanı Kütüphanesi'nin sadece yapı açısından birbirine benzeyen eleman nesnelerinden oluşturulmuş olması.

```
Type durumT
   naÇözüm As Boolean
   puan As Double
   aşılamazPuan As Double
   elemanlar() As DElemanT
   tanımlayıcı As String
End Type

Dim durumlar() As durumT
```

Arama algoritmasında işlev gören diğer değişkenler ise durum tekrarlarının kontrol edilmesi, açılan her durum budağının oluşturduğu açılacak yeni budakların tutulması, o ana kadar ulaşılan en iyi çözüm durumunun saklanması ve silinen durumların hafızada yer kaplamaması için, üzerine yazılmak üzere kayıt edildiği aşağıdaki yapılardır

```
Dim açılanDurumlar() As Long
Dim açılacakDurumlar() As Long
Dim enİyiÇözüm As durumT
Dim freeDurumKeys() As Long
```

2.2.2.4 Arama algoritması ve işleyişi

Arama algoritmasının bir ağaç yapısı oluşturduğundan daha önceki bölümlerde bahsedilmiştir. Çözüm durumları bu ağaç yapısı içinde aranır. Algoritmanın başlamasından önce bir dizi ön işlem yürütülür. Bu işlemler :

- Temel değişkenlerin oluşturulup sıfırlanması :

```
ReDim açılacakDurumlar(0)
ReDim açılanDurumlar(0)
ReDim freeDurumKeys(0)
enİyiÇözüm.naÇözüm = True,
```

- Çalışma Zamanı Kütüphanesi verilerinin arama algoritması hazırlık veri yapılarına aktarılması :

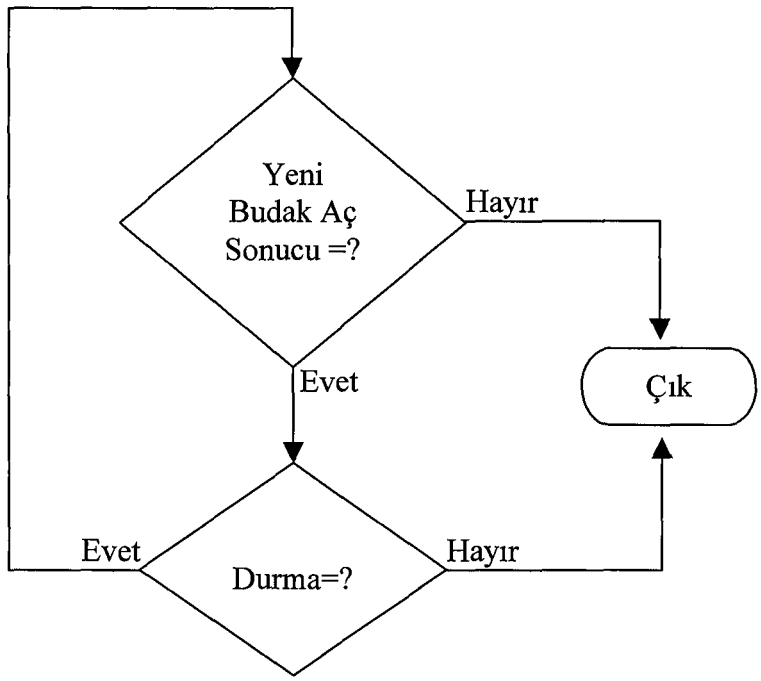
```
eleman_iskeletlerini_oluştur,
```

- Ağaç yapısındaki ilk(kök) durum olan "Başlangıç Durumu" nun oluşturulması :

```
başlangıçDurumuOluştur,
```

- Tanımlama bölümü ve Eleman Kütüphanesi'ndeki verileri taşıyan bağlı liste yapıları silinerek hafızanın boşaltılmasıdır.

Başlangıç durumunun başarı ile oluşturulmasından sonra artık arama algoritması çalıştırılabilir.



Şekil 2.35 Aramada döngü algoritması

```
Do While openNode
    DoEvents
    If Not durma Then Exit Do
Loop
```

Çalıştırma işlemi aslında yeni budakların açıldığı sonsuz bir döngüdür. Sadece yeni açılan bir budağın bir optimal çözüm durumu döndürmesi, açılacak hiçbir budak kalmaması yada kullanıcının düşünme işlemine bir komut ile ara vermesi ile döngüden çıkış şartları oluşur. Döngü içinde ise yeni budakları açan alt yordam "openNode" yordamıdır.

```
Private Function openNode() As Boolean
  Dim açılacakDurum As Long, newDurumKey As Long, cnt As Long, elemKey As Long
  Dim açılacakDurumSonYeri As Long, adCnt As Long

  If açılacakDurumlar(0) = 0 Then durma = False: Exit Function' açılacak durum kalmadıysa durdurur.
  openNode = True
  açılacakDurum = açılacakDurumlar(1)
  For elemKey = 1 To UBound(durumlar(açılacakDurum).elemanlar)
    Do
      cnt = durumlar(açılacakDurum).elemanlar(elemKey).adayChild
      If cnt < 1 Then Exit Do
      If cnt = altGuruplar(durumlar(açılacakDurum).elemanlar(elemKey).orijin).endElem Then
        durumlar(açılacakDurum).elemanlar(elemKey).adayChild = 0
      Else
        durumlar(açılacakDurum).elemanlar(elemKey).adayChild = cnt + 1
      End If

      newDurumKey = yeniDurum(açılacakDurum, elemKey, cnt)

      If newDurumKey Then
        ReDim Preserve geçPuanSırası(UBound(geçPuanSırası) + 1)
        geçPuanSırası(UBound(geçPuanSırası)) = açılacakDurum & " => " & newDurumKey & "    " &
              durumlar(açılacakDurum).puan & "  " & durumlar(newDurumKey).puan

        If (durumlar(newDurumKey).aşılamazPuan >= enİyiÇözüm.aşılamazPuan) Or _
          enİyiÇözüm.naÇözüm Then
          yerleştir newDurumKey 'açılacaklar içinde uygun yerine
          Exit Function
        Else
          cnt = UBound(açılanDurumlar) + 1
          ReDim Preserve açılanDurumlar(cnt)
          açılanDurumlar(cnt) = newDurumKey
        End If
      End If
    Loop
  Next elemKey
  'açılacakDurum'un hiçbir elemanı açılamamışsa bu durum açılacaklardan silinir
  açılacakDurumSil 1
End Function
```

Arama algoritması içindeki görünümleri itibarı ile; elemanlar her durumT yapısı içinde doğrusal olmayan bir ağaç yapısı halinde saklanırken, durumlar, verilerinin her biri bir durumT yapısı olan iki farklı alanda doğrusal bir yapıda saklanırlar. Bunlardan biri "açılanDurumlar" diğeri ise "açılacakDurumlar" dizi değişkenidir. "açılanDurumlar" dizi değişkeni içinde durumlar sadece tekrarlayan durumların yönetimi amacıyla saklanmakta olup bu durumlar iç yapıları itibarı ile artık yeni budaklar oluşturma yeteneğini kaybetmiş(tüm budakları açılmış)

durumlardır. "açılacakDurumlar" dizi değişkeninde ise hala içinde yeni budakların açılabileceği durumlar saklanır. "açılacakDurumlar", "açılanDurumlar"dan farklı olarak durumT yapısı içindeki "puan" alanının değerine göre durumlarını daima büyükten küçüğe doğru sıralanmış olarak tutar. Bu sıralama bilgilendirilmiş arama (Informed Search) metotları için büyük önem arzeder. Çünkü bilgilendirilmiş arama metodlarında, bir sonraki aşamada sezgisel fonksiyonlar sonucu optimal çözüme en yakın olduğu değerlendirilen durum öncelikli olarak işleme tabi tutulmalıdır. İşte "openNode" alt yordamı da aramanın tipi ne olursa olsun daima "açılacakDurumlar" dizi değişkeni içindeki ilk durum ile işlem yapar. Aramanın bilgilendirilmemiş(Uninformed Search) olması durumunda ise "puan" değeri tüm durumlar için aynı olacağından "açılacakDurumlar" dizi değişkeni içinde sıralama yapılamayacak, arama doğal olarak enine aramaya(Breadth First Search) dönüşecektir.

Bir durum üzerinde işlem yapılırken dikkati çeken nokta aynı durum içinde açılabilecek çok sayıda budak olabileceğidir. Bu budaklar durum yapısı içindeki her elemanın alt gurubu içindeki elemanların sayılarının toplamı kadardır.

$$\sum_{E=1}^{N} \text{Alt Gurup eleman sayısı}$$

Budak açma işlemine sanki yukarıdaki formülde sayısı hesaplanan tüm budaklar açılacakmış gibi Tepe Eleman'dan başlanır. Döngü bu şekilde kurulmuştur fakat sadece bir tek budak açılıp "openNode" yordamından çıkılır. Çünkü açılan her budak yeni bir durum oluşturur ve bu yeni durum "açılacakDurumlar" dizi değişkeni içinde uygun yerine yerleştirilir. Yeni durum puan açısından geldiği durumdan daha iyi bir sonuç verebileceğinden gelinen durumun işlemine kalınan yer işareti konularak ara verilir ve artık yeni durum üzerinde işleme devam edilir.

Daha önce de belirtildiği gibi budak açma işlemi aslında mevcut durumdan yeni bir durum oluşturma işlemidir. Bu işlem esnasında durumda bir değişiklik olacağı açıktır. Yeni durum kıymetlendirilmesi açısından üç kategoride değerlendirilir :

- Yeni durum tanımlı kurallara uymamaktadır.
- Yeni durum tanımlı kurallara uyar fakat çözüm değildir.
- Yeni durum bir çözüm durumudur.

İlk seçenekte olduğu gibi yeni durum kriterlerce tanımlanmış problem kurallarına uymayan bir durum ise hemen silinir. İkinci seçenek olarak eğer durum problemin kurallarına uyuyor fakat bir çözüm durumu niteliğini taşımıyor ise etki işlemleri yapıldıktan sonra puanı ve aşılamaz puanı hesaplanır ve puanına göre "açılacakDurumlar" dizisi içinde uygun yerine yerleştirilir. Son seçenekte olduğu gibi eğer yeni durum bir çözüm durumu ise "enİyiÇözüm" değişkenine aktarılır ve arama işlemine son verilip verilmeyeceği değerlendirilir. Arama işlemine son verilmesi şartı daima açılacak durum kalmamasıdır. Bir çözüm durumu üretildiğinde açılacak durumlar içinden aşılamaz puanı, bulunan çözüm durumundan daha kötü olanlar silinir. Silme işleminden sonra da yeni durumlarda aşılamaz puanın mevcut çözüm durumuna eşit yada daha iyi olması şartı ile aranmaya devam edilir. Silme işleminden sonra eğer "açılacakDurumlar" dizisi tamamen boşalmışsa daha iyi bir çözüm bulunamayacağı değerlendirilerek arama algoritması sonlandırılır. Eğer hala "açılacakDurumlar" dizisi içinde durumlar varsa mevcut çözümden daha iyi bir çözümün bulunabileceği umudu ile arama işlemine devam edilir. Arama işleminin herhangi bir noktasında kullanıcı tarafından verilen bir komut da arama işlemini duraklatabilir.

Şimdiye değin bu bölümde yapılan açıklamalardan anlaşılacağı üzere bir durumun problem kurallarına uygunluğu, çözüm olup olmadığı, puan ve aşılamaz puanının hesaplanması önemli bir aşamadır ve kriter ifadelerinin hesaplanması ile yapılır. Kriter ifadelerinin tanımlanmasında kullanılan "Önem Derecesi" ve "Kriter Türü" bir durumun problem kurallarına uyup uymadığının bir göstergesi olarak değerlendirilir. Kriter türü "Anlık", önem derecesi "Zorunlu" ve ifade sonucu olumlu değer döndürmüyorsa durum problem kurallarına uymuyor demektir[1]. Bu uygunsuzluk öncelikli olarak hesaplanan Seçim Kriterleri'nden kaynaklanıyor ise etki ifadeleri hiç hesaplanmaz. Seçim Kriterleri'nden başarı ile geçen bir durumun

[1] Örneğin : Satranç oyununda Taş nesnesinin alt gurubu satranç tahtasındaki tüm Kare'lerdir. Oysa Şah taşı sadece bir komşu kareye(eğer bu kare tehdit altında değilse) hareket edebilir. Şah taşının iki kare ilerdeki bir kareye hareket ettirildiği bir durum kurallara uymayacağından birdaha değerlendirilmemek üzere silinir.

etkileri hesaplanır ve uygulanır.(Uygunsuzluk etkilerden sonra da ortaya çıkabilir.) Son olarak kriterleri hesaplanan durumun artık tüm kıymetlendirme verileri ortaya çıkmıştır. Tekrarlayan bir durum olup olmadığı test edildikten sonra sıraya koyulmak üzere "openNode" yordamına geri döndürülebilir. Tüm bu işlemler "openNode" yordamı içinden çağrılan "yeniDurum" alt yordamı aracılığı ile gerçekleştirilir.

"yeniDurum" alt yordamı içinde öncelikle üst durumun bir kopyası alınır. Bu kopya durumun temel değişkenlerinin(elemanlar hariç) değerleri sıfırlanır. Daha sonra açılacak budak alt gurupta sırada bekleyen ilk eleman olarak "iskeletElemanlar" yapısından aynen kopyalanarak ağaca eklenir. Ağaçtaki yeri ile ilgili ilişkileri de kurulduktan sonra sırasıyla Seçilme Kriterleri, Etkileri ve Kriterleri uygulanır. Seçilme Kriterleri ve Kriterlerin uygulanması aşamasında kurallara uygunsuzluk durumu değerlendirilip gerekiyorsa yeni durum silinerek alt yordamdan çıkılır. Kurallara uygun bir durumun daha önce üretilmiş diğer durumlardan ağaç yapısı ve anahtar özelliklerin değerleri bakımından farklı olması gerekir. Aksi takdirde arama algoritması aynı yere dönen çemberler çizmek yada geldiği bir duruma geri dönmek sureti ile çıkmaza girebilir. Bu işlem için de kontrolu yapan "checkBefore" alt yordamı kullanılır. Herşey normal ise "yeniDurum" alt yordamı yeni durumu geri döndürecektir.

```
Private Function yeniDurum(ParentDurumKey As Long, ScopeElemKey As Long, newChildKey As Long) As
                                                                                    Long
  Dim newDurumKey As Long, newElemKey As Long, newChRef As Long, cnt As Long

  newDurumKey = freeDurumKeyBul
  durumlar(newDurumKey) = durumlar(ParentDurumKey)
  durumlar(newDurumKey).aşılamazPuan = 0
  durumlar(newDurumKey).puan = 0
  durumlar(newDurumKey).naÇözüm = False
  durumlar(newDurumKey).tanımlayıcı = ""
  With durumlar(newDurumKey)
    For cnt = 1 To UBound(.elemanlar)
      .elemanlar(cnt).adayChild = altGuruplar(.elemanlar(cnt).orijin).startElem
    Next cnt
    newElemKey = UBound(.elemanlar) + 1
    ReDim Preserve .elemanlar(newElemKey)
    .elemanlar(newElemKey) = iskeletElemanlar(newChildKey)
    .elemanlar(newElemKey).Parent = ScopeElemKey
    newChRef = UBound(.elemanlar(ScopeElemKey).childs) + 1
```

```
      ReDim Preserve .elemanlar(ScopeElemKey).childs(0 To newChRef)
      .elemanlar(ScopeElemKey).childs(newChRef) = newElemKey
      If seçilmeKriterleriniUygula(durumlar(newDurumKey), newElemKey) Then
        GoTo sil
      Else
        etkileriniUygula durumlar(newDurumKey), newElemKey
        If durumKriterleriUygula(durumlar(newDurumKey)) Then GoTo sil
      End If
    End With
    anahtarYaz durumlar(newDurumKey)
    If checkBefore(durumlar(newDurumKey).tanımlayıcı) Then
      GoTo sil
    End If
    yeniDurum = newDurumKey
    Exit Function
sil:
    durumSil newDurumKey
End Function
```

Kriter ve etkilerin uygulanmasında işlemleri farklı yordamlarda yapılmasına karşın işlem genelde aynıdır. Uygulanacak ifade "elemFormülGurupları" yapısından ilgili eleman için alınır, şart ifadesine öncelik verilerek ifade içindeki göreceli çağırımların değerleri durumun ağaç yapısındaki elemanlardan çeklilip yerlerine yerleştirilir, oluşan ifade "evaluate" fonksiyonu aracılığı ile hesaplanır ve sonuçlar operatöre göre değerlendirildikten sonra önem değerinin pozitif yada negatif olma durumları, zorunluluk durumu da değerlendirilerek parçalanır. Parçalar türün "Nihai" yada "Anlık" olmasına göre puan ve aşılamaz puan şeklinde durum içindeki yerlerine yazılır. Etkiler için ise hesaplanan ifade etkilenecek alana aktarılır. Tüm bu işlemlerin yapılmasının ana şartı şart ifadesinin "Evet" değeri döndürmesidir.

Herhangi bir sebeple arama işlemi duraklatılmışsa yine değerlendirilmesi gereken üç seçenek vardır :

- Henüz bir çözüm durumu ile karşılaşılmamamıştır.("enİyiÇözüm" boş)

Bu durum ile arama işlemi kullanıcı tarafından duraklatıldığında yada çözümün imkansız olduğu durumlarda karşılaşılır, kullanıcı isterse duraklattığı arama işlemine kaldığı yerden devam eder. Yada çözüm olmasa da mevcut en iyi durumu görüntülemek için arama işlemini kalıcı olarak durdurabilir.

- Mevcut bir çözüm vardır, arama devam ederken duraklatılmıştır.

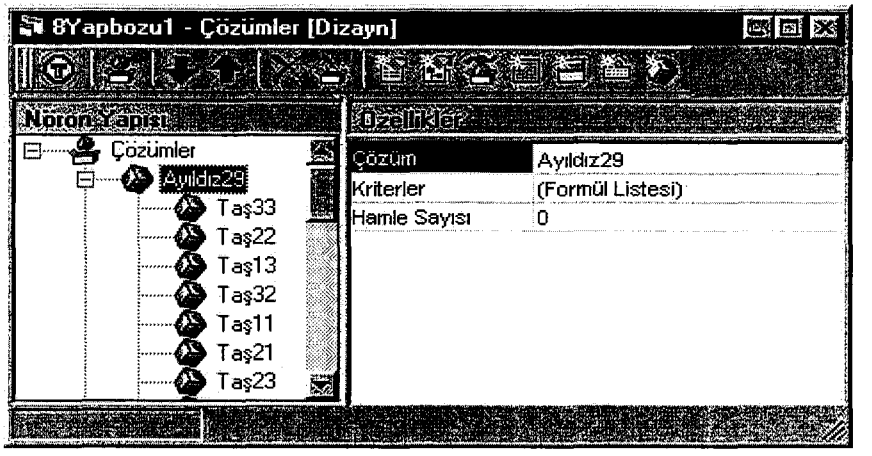
Bu durumda da yine kullanıcı devreye girmiştir. Mevcut çözüm en iyi çözüm olmayabilir. Kullanıcı isterse arama işlemine kaldığı yerden devam eder. Yada mevcut çözümü en iyi çözüm kabul ederek arama işlemini kalıcı olarak durdurabilir.

- Arama işlemi "açılacakDurumlar" dizisi boşaldığı için durmuştur.

Mevcut en iyi(optimal) çözüm bulunmuştur. Arama işlemine artık devam edilemez.

2.2.3 Sonuç gösterim bölümü

Arama algoritması herhangi bir nedenle tamamen durdurulduğunda mevcut çözüm durumu yada çözüm durumu yoksa en iyi durum, Eleman Kütüphanesi kayıt ortamından okunduktan sonra "sonuçYaz" alt yordamı aracılığı ile Çalışma Zamanı Kütüphanesi'nden sonra gelen ve Eleman Kütüphanesinin son nesnesi olan Öneri Plan yapısı içine yazılır ve Eleman Kütüphanesi tekrar kayıt ortamına aktarılır. Tanımlama bölümü ara yüzlerine tekrar dönüldüğünde ise "Otomatik Düşün" komutunun verildiği Tepe Eleman nesnesinin altında üretilen öneri plan uygulanmaya hazır olarak görüntülenir.



Şekil 2.36 Otomatik üretilmiş planı ile bir Eleman nesnesi

3 ÖRNEK UYGULAMA

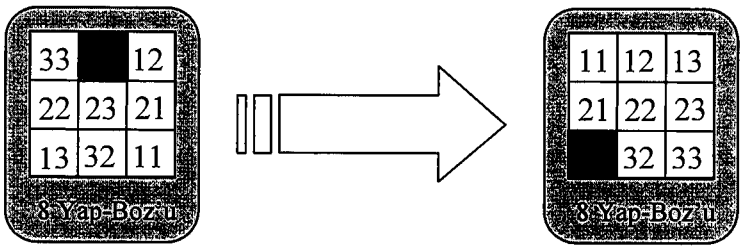
ABS'in ne olduğu, ne iş yaptığı ve nasıl çalıştığının en güzel bir örnek üzerinde açıklanabileceği değerlendirildiğinden bu bölümde seçilen bir planlama problemi kod seviyesine hiç inilmeden tamamen ABS'in sunduğu imkanlarla, sezgisel metotları ile birlikte tanımlanacak ve çözdürülecektir. Bu bölümde sadece tanımlamalar ile çözüm süreci değerlendirilecek olup üretilen plan üzerindeki irdelemeler sonuç bölümünde yapılacaktır.

3.1 Örnek Seçimi

Bölüm 1.1 de "Planlama Problemlerine Genel Bakış" adı altında bir ABS'in ilgi alanına giren problemlere kısaca değinilmiştir. Bu bölümde ise kapsama uygun ve aynı zamanda karşılaştırmaların yapılabilmesi amacıyla Yapay Zeka araştırmalarında üzerinde daha önce durulmuş bilinen bir oyuncak problemi örnek olarak alınmıştır.

3.2 8 Yap-Boz'u

8 Yap-Boz'u hemen hemen herkesçe bilinen ve Yapay Zeka araştırmalarında sıklıkla kullanılan, sezgisel metotların etkinliğinin belirgin bir şekilde gözlemlenebildiği ilgi çekici bir oyuncak problemidir. Problem yukarı-aşağı ve sağa-sola doğru kaydırılarak hareket ettirilebilen 8 adet taşın 3*3 lük bir muhafaza üzerine bir boş yer kalacak şekilde yerleştirilmesi ile oluşmuştur. Boşluğun yanındaki herhangi bir taşın boşluğa doğru kaydırılarak hareket ettirilmesi ile oynanan oyun karışık vaziyette duran taşların olmaları gereken yerlere getirilmeleri ile son bulur. Bir gerçek dünya problemi olarak 8 Yap-Boz'u insan tarafından kolay algılanması yada daha ilgi çekici olması için taşları, uygun yerlerinde olmaları durumunda düzgün görünen bir resim yada şekil ile boyanmış, kurallarını fiziksel yapısında gizleyen bir oyuncaktır. Ancak buradaki gösterimde taşlar üzerinde bir şekil yada resim değil, boyca(Y) ve ence(X) eksenlerde taşların olmaları gereken konumları temsil eden sayılar kullanılmıştır.



Başlangıç Durumu Çözüm Durumu

Şekil 3.1 8 Yap-Boz'u

8 Yap-Boz'unun kurallarına kısaca değinecek olursak :

- Taşların başlangıç konumlarından oluşan bir durum vardır.
- Sadece yanında boşluk olan taşlar boşluğa doğru kaydırılabilir.
- Tüm taşların orijinal konumlarında olması durumu çözüm durumudur.

8 Yap-Boz'unun bir planlama problemi olup olmadığı sorusuna kesinlikle "evet" yanıtı verilebilir. Çünkü başlangıç durumundan çözüm durumuna ulaşmak için hangi taşların oynanması gerektiğini içeren bir tablo aslında bir plandır. Bu planın etkinliği ise çözüme ulaşılıncaya kadar toplam kaç taşın hareket ettirildiği ile ölçülür(Path Cost). Her başlangıç durumu için en az taş hareket ettirilerek çözüme ulaşan bir plan vardır ki bu plan ABS içinde En Zeki Plan olarak adlandırılır.

3.3 8 Yap-Boz'unun Zorluk Derecesi Hakkında

8 Yap-Boz'unu orta düzeyde zeki bir insan birkaç dakikada çözebilmektedir. Bunu, çözümde oynanan taşları içeren bir plan olarak ele alırsak En Zeki Planı üretmek çok zeki insanlar için bile zor yutulur bir lokmadır. Çoğu kez insan bulduğu çözümün "En Zeki" olup olmadığını bile bilemeyecektir.

"8 Yap-Boz'u ilginç olabilecek tam uygun düzeyde bir zorluk derecesine sahiptir. Başlangıç durumuna göre değişmekle beraber, tipik bir çözüm yaklaşık 20 hamlede gerçekleşir. Problemin dallanma faktörü yaklaşık 3 tür(boşluk ortada olduğunda 4, kenarlardan birinde olduğunda 3, köşede olduğunda ise 2 taş hareket ettirilebilir). Bunun anlamı 20 hamle derinliğe ulaşan ayrıntılı bir arama için durum sayısının $3^{20} = 3.5 \times 10^9$ *durum gibi gözükeceğidir. Bu sayıyı tekrarlayan durumların izini takip ederek önemli ölçüde azaltabiliriz çünkü 9 taşın*

sadece 9! = 362800 değişik aranjmanı vardır. Bu halen büyük bir durum sayısıdır ve sonraki işimiz iyi bir sezgisel fonksiyon bulmak olmalıdır." (Çeviri : Artificial Intelligence A Modern Approach, Stuart J. Russel ve Peter Norvig)

8 Yap-Boz'u ile ilgili olarak daha önce de yapılmış araştırmalar göstermektedir ki; sadece bir çözüm bulmak değil de en az hamle ile çözüm bulmak söz konusu olduğunda problem insan için oldukça zor sayılabileceği gibi bir bilgisayar sistemi için de dikkate değer bir zorluk derecesi ortaya koymaktadır.

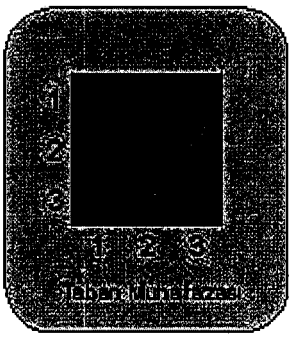
3.4 Problemin Tanımlanması

ABS içinde bir problem tanımlanırken bir tanımlama sırası gerekmektedir. Kriter ve etki tanımlamaları problem nesneleri içine tanımlanacağından ve nesnel tanımlamalara göre anlam kazanacağından nesnel tanımlamalar öncelikli olarak yapılmalıdır. Her aşamada yapılan tanımlar üzerinde değişiklik yapmak mümkündür. Ancak bu tanımlamalar yapıldıktan sonra nesnel yapıda yapılacak değişikliklerde sistem kullanıcının ne yapmak istediği tahmin edilemeyeceğinden bir takım tanımlamalar yine kullanıcı tarafından düzeltilmesi beklenerek hatalı olarak imlenebilir.

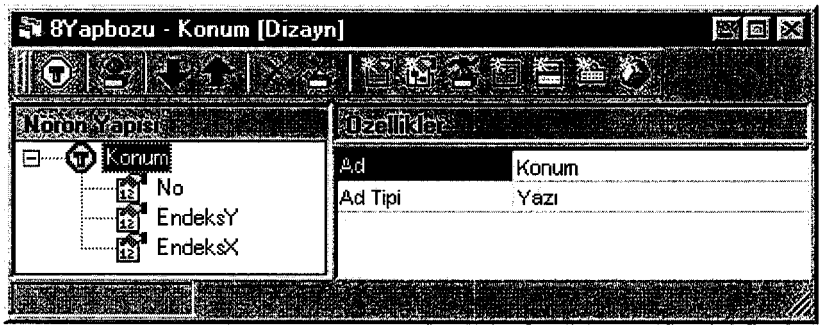
8 Yap-Boz'u nesnel tanımlama aşamasında Konum, Taş ve Çözüm olmak üzere toplam 3 adet Tanım nesnesi, Çözüm ve Taş nesnelerinden oluşan bir Tanım Zinciri ve Tanım Zinciri nesnelerine tanımlanacak kriter ve etki ifadelerini gerektirir. ABS'in kullanıcı arayüzü bu tanımlamaların kolaylıkla yapılabilmesi için özel olarak tasarlanmıştır.

3.4.1 Konumların tanımlanması ve türetilmesi

Tanımlanacak ilk Tanım nesnesi taşların üzerine yerleştirildiği taş yerleri olan, genellikle plastikten imal edilmiş taban muhafazasıdır. Taban muhafazasında taşların içinde bulunduğu alan 3x3 taş ebatlarında olup tanımlamalarda kullanılacak matematiksel işlemler açısından boyca(Y) ve ence(X) 1 den 3 e kadar tam sayılar ile numaralandırılmıştır. Bu numaralama iki boyutlu bir diziyi andırır. Ayrıca yapılan bir diğer numaralandırma ise sol üst köşeden sağ alt köşeye doğru satır satır giden 1 den 9 a kadar olan genel numaralandırmadır.

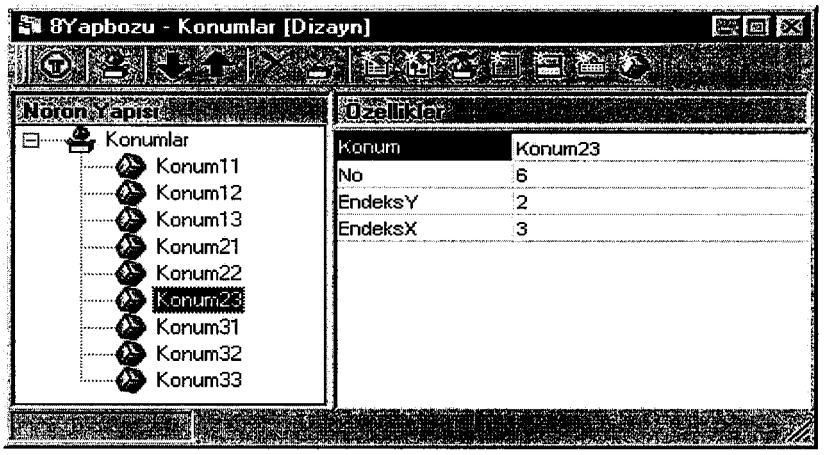


Şekil 3.2 Konumların bulunduğu taban muhafazası



"Konum" Tanım nesnesinde "No", "EndeksY" ve "EndeksX" sayı tipli değeri sabit Özellik nesneleri bulunur.

Şekil 3.3 Tanımlanan "Konum" Tanım nesnesi

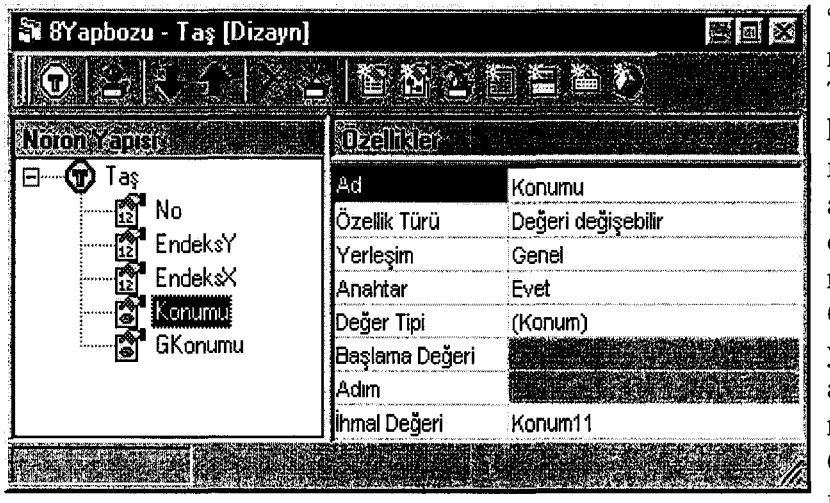


"Konum" Tanım nesnesinden 9 adet gerçek Konum nesnesi türetilir adları ve özellik değerleri girilir.

Şekil 3.4 Türetilen gerçek "Konum" nesneleri

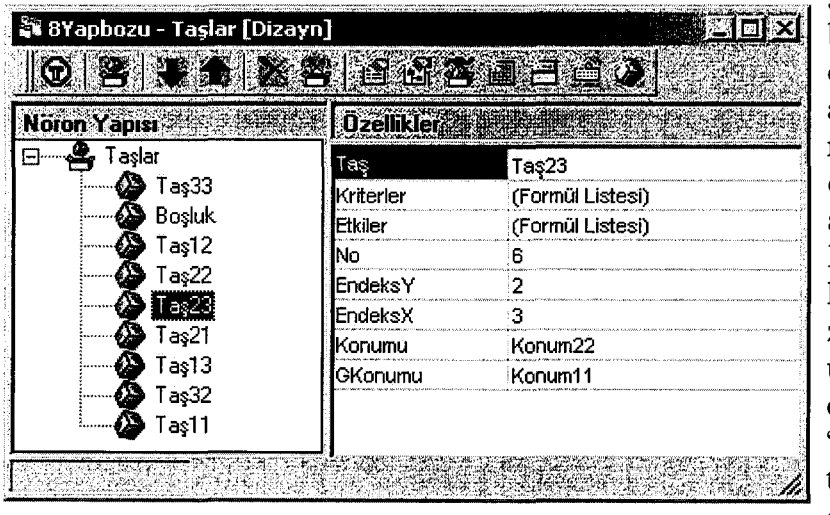
3.4.2 Taşların tanımlanması ve türetilmesi

Toplam 8 adet olan taşlar kare şeklinde genellikle plastikten imal edilmiş parçalardır. Taşlar konumlar ile benzer şekilde numaralandırılmışlardır.



Şekil 3.5 Tanımlanan "Taş" Tanım nesnesi

"Taş" Tanım nesnesine "Konum" Tanım nesnesindeki üç Özellik nesnesinin yanı sıra anahtar durumunda olan "Konumu" nesne değer tipli Özellik nesnesi ve yine "Gkonumu" adlı bir başka nesne değer tipli Özellik nesnesi tanımlanır.



Şekil 3.6 Türetilen gerçek "Taş" nesneleri

"Taş" Tanım nesne sinden "Boşluk" da bir taş nesnesi olacak şekilde 9 adet gerçek "Taş" nesnesi türetilir ve özellik değerleri atanır. Bu atamalarda taşların konumlarının bozulmuş bir yap-boz u tanımladığına dikkat ediniz. "GKonumu" bir taşıma değişkeni olarak kullanılacaktır.

3.4.3 Çözümün tanımlanması ve Tanım Zinciri'nin oluşturulması

Şimdiye dek yapılan nesnel tanımlamalar ile ABS'e problemin fiziksel yapısı aktarılmıştır. Problemin çözümünün genel yapısının nasıl olacağı ise "Çözüm" nesnesi ile tanımlanacaktır. "Çözüm" nesnesi Tanım Zinciri'nin iki halkasından üstte olanıdır ve "Çözüm" nesnesini "Taş" nesneleri ile ilişkilendirecek olan projedeki tek Bağlı Küme nesnesine ev sahipliği yapar. Doğal dille anlatım bu ilişkiyi daha kolay açıklamaktadır. "8 Yap-Boz'unun ***çözümü*** bir dizi ***taş*** oynanarak bulunur."

"Çözüm" Tanım nesnesinin "Hamle Sayısı" Özellik nesnesi ve "Taşlar" ile bağlantıyı sağlayan "Oynanan Taş" Bağlı Küme nesnesi mevcuttur.

Şekil 3.7 Tanım nesnesine bağlanan Bağlı Küme nesnesi

Şimdiye değin yapılan tanımlamalar ile 8 Yap-Boz'u probleminin nesnel tanımlamaları bitirilmiştir. Sonraki aşama problem kuralları, akış değişikliklerinin durum yapısına etkileri ve çözüm için gerekli sezgisel metotların kriter ve etki ifadeleri aracılığı ile sisteme tanımlanmasıdır.

3.4.4 Taş Seçim Kriterlerinin tanımlanması

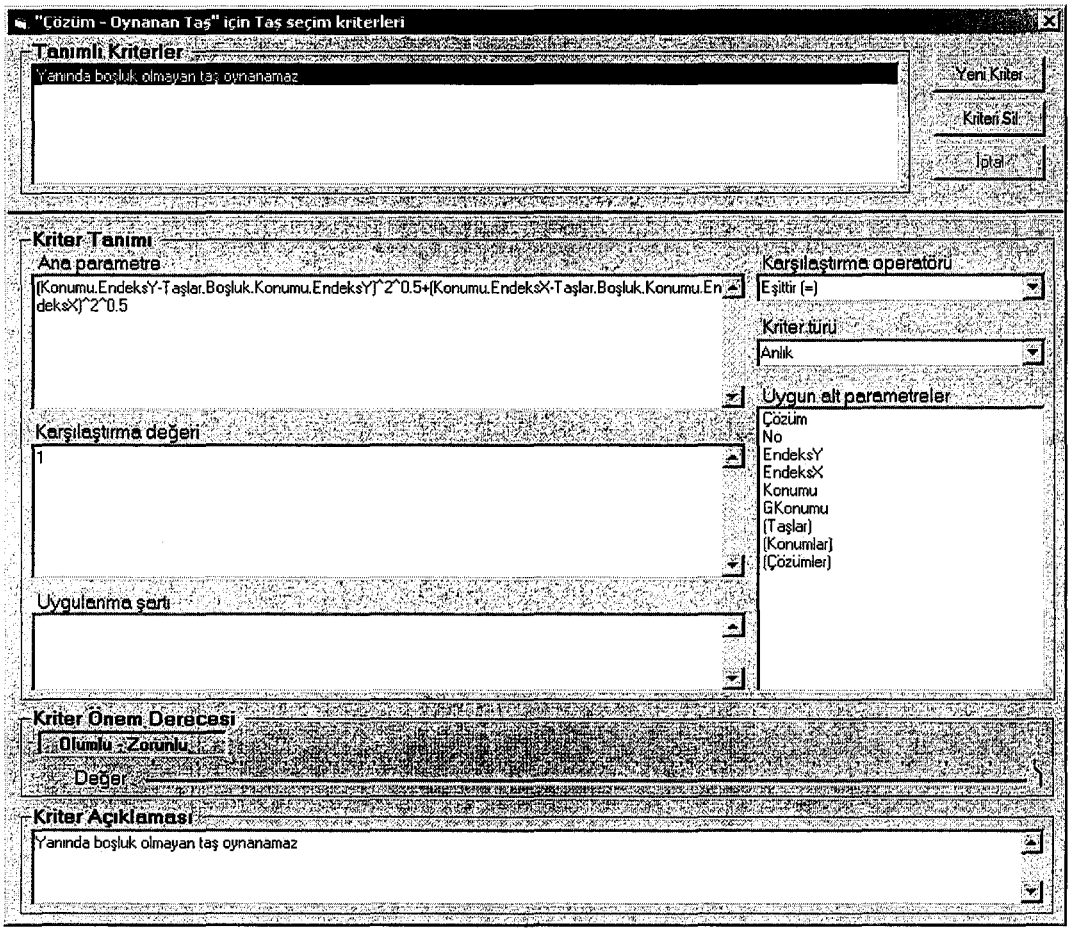
8 Yap-Boz'u çözülmeye çalışılırken geçerli olan fiziksek kurallar bu aşamada tanımlanacaktır. Oyuncağın fiziksel yapısı göz önüne alındığında "hangi taşlar hareket ettirilebilir" sorusunun cevabı bu kuralı açıklar. Elbette bir yanında boşluk olmayan taş hareket ettirilemez. İşte "Oynanan Taş" Bağlı Küme nesnesinin anlık türdeki zorunlu Seçilme Kriteri de bu olacaktır.

Yapılacak işlem sadece yanında boşluk olan taşları diğer taşlardan ayırt edebilecek matematiksel bir ifadedir. Herhangi bir taşın "Boşluk" taşına olan fiziksel mesafesi 1 birim(taş boyu) olduğunda bu taş "Boşluk" taşı ile yan yanadır. Köşeden komşuluk kapsama girmez çünkü köşeden komşu olan taş 1.41 birim uzaklıktadır. Bu yaklaşımın ABS içindeki ifadesi şöyle olacaktır : Bir taş ile "Boşluk" taşı arasındaki X ve Y eksenleri farklarının mutlak degerleri toplamı 1'e eşit olmalıdır.

Oynanacak taş için ;

(Konumu.EndeksY - Taşlar.Boşluk.Konumu.EndeksY) ^2 ^0.5 +
(Konumu.EndeksX - Taşlar.Boşluk.Konumu.EndeksX) ^2 ^0.5 = 1

Bu taş seçme kriteri hiçbir zaman aşılamayacak bir kural olarak tanımlanacağından türü "Anlık", Önem Derecesi ise "Olumlu-Zorunlu" olmalıdır.



Şekil 3.8 Taş Seçim Kriterleri

3.4.5 Taş Etkilerinin tanımlanması

8 Yap-Boz'unun çözüm aşamalarında bir taş hareket ettirildiğinde ne olur ? Yukarıdaki Seçim Kriteri zaten sadece "Boşluk" taşının yanında olan taşların hareket ettirilebileceğini tanımlamıştır. Dolayısı ile hareket ettirilen bir taş mevcut duruma "Boşluk" taşı ile yer değiştirmesi şeklinde bir etki yapacaktır. Ayrıca hareket ettirilen her taşı bir hamle olarak değerlendirecek olursak Çözüm nesnesi

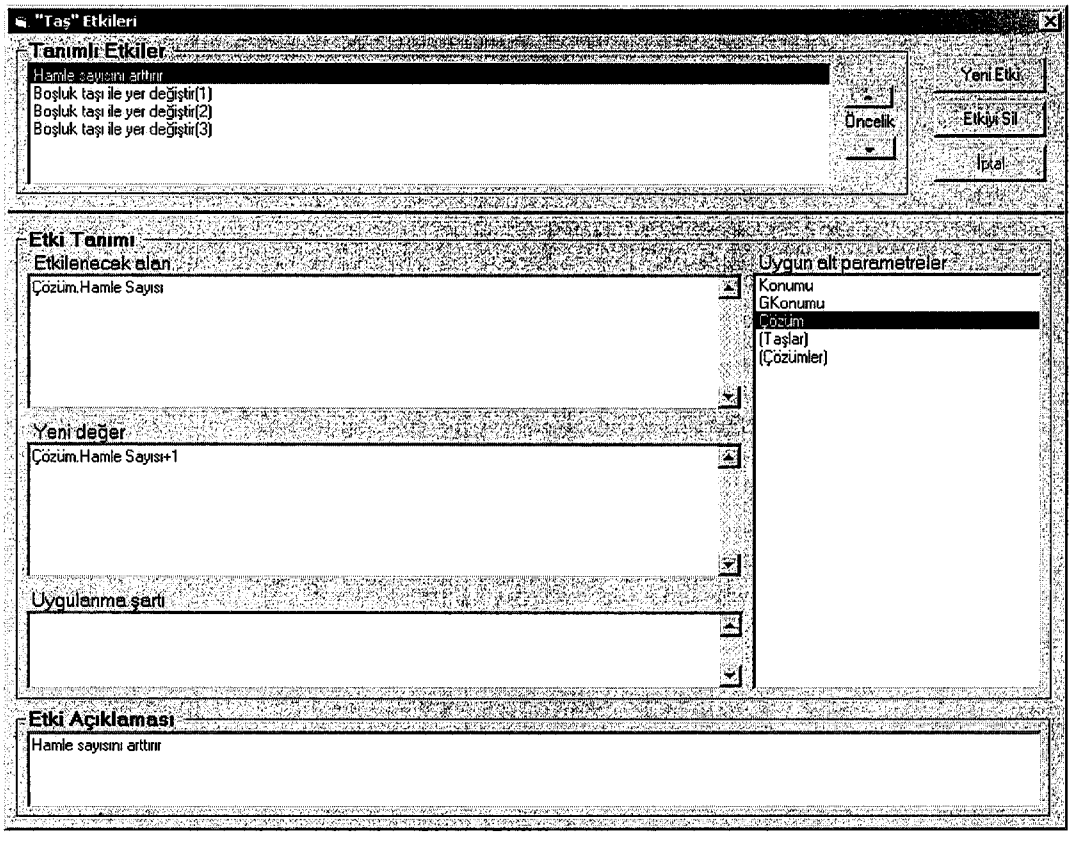
içindeki "Hamle Sayısı" özelliğinin değerinin de hareket ettirilen her taş için bir artması da tanımlanması gereken bir etkidir. Hamle sayısı bir ifade ile arttırılabilir.

Çözüm.Hamle Sayısı = Çözüm.Hamle Sayısı + 1

Ancak boşluk taşı ile yer değiştirme işlemi bir değiştokuş(swap) işlemi olduğundan 3 adım ve bir ek değişken gerektirir. Bu ek değişken Taş nesnesi yapısındaki "GKonumu" Özellik nesnesi olacaktır.

Taş nesnesi için

- GKonumu = Konumu
- Konumu = Taşlar.Boşluk.Konumu
- Taşlar.Boşluk.Konumu = GKonumu



Şekil 3.9 Taş Etkileri

3.4.6 Çözüm Kriterlerinin tanımlanması

Problemin çözülebilmesi için yapılacak son tanımlamalar Çözüm nesnesinin kriterleridir. Bu alanda 3 adet kriter tanımlaması yapılacaktır.

- Çözüm durumunun özelliklerinin tanımlanması
- Hamle Sayısının minimuma zorlanması
- Çözüme götüren sezgisel metodun tanımlanması

Çözüm durumunun tanımlanması oldukça kolaydır. Her taşın "No" özelliğinin değeri eğer konumunun "No" özelliğinin değerine eşitse tüm taşlar yerinde ve problem çözülmüş demektir.

Taşlar.Konumu.No = Taşlar.No

Bu kriter zorunlu bir kriter olmalıdır. Ancak ara aşamalarda sağlanmaması sonuçta sağlanamayacağı anlamına gelmez. Dolayısıyla kriter türü "Nihai" olacaktır.

Hamle sayısının minimuma zorlanması ise şu ifade ile sağlanır :

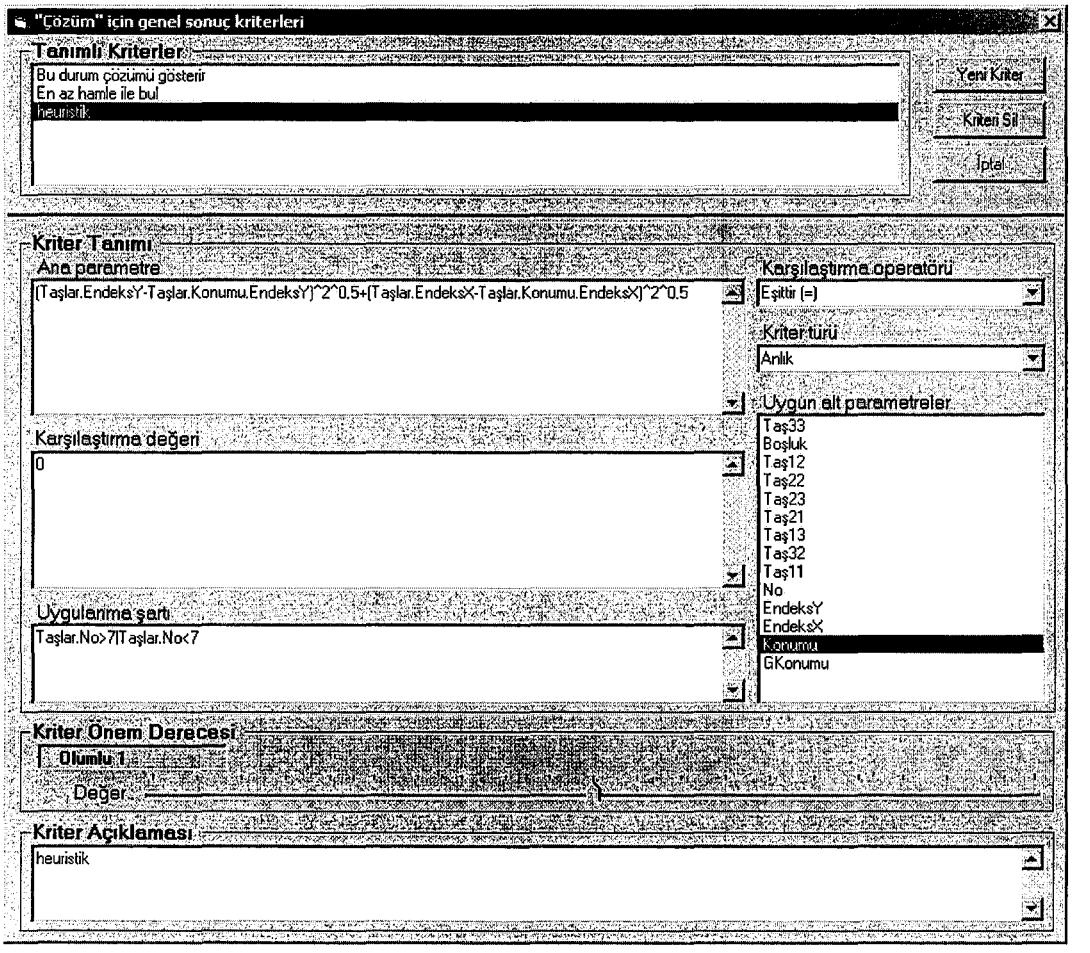
Çözüm.Hamle Sayısı = 0, Önem derecesi = 1, Tür = "Anlık"

Sıra sezgisel fonksiyona gelmiştir. 8 Yap-Boz'u için bilinen iyi sezgisel metotlardan biri taşların olmaları gereken konumlara olan blok uzaklıklarının[1] çözüme olan yakınlığın bir belirtisi olarak değerlendirilmesidir. Bu metot şu şekilde ifade edilebilir :

(Taşlar.EndeksY - Taşlar.Konumu.EndeksY) ^2 ^0.5 +
(Taşlar.EndeksX - Taşlar.Konumu.EndeksX) ^2 ^0.5 = 0

Türü yine anlık olan bu kriterin Önem derecesi 1 ("min Hamle Sayısı" kriteri ile aynı) dir. Kriterin "Şart İfadesi" alanındaki ifade ise "Boşluk" taşını ifade eden 7 nolu taşın bu işlemlerden muaf tutulacağı anlamına gelmektedir.

[1] Blok uzaklığı kavramı yap-boz'da taşların köşegenlere doğru değil de sadece sağa-sola ve yukarı-aşağı hareket edebiliyor olmalarından kaynaklanan ve yatay mesafe ile dikey mesafenin aritmetiksel olarak toplanması ile elde edilen mesafedir.

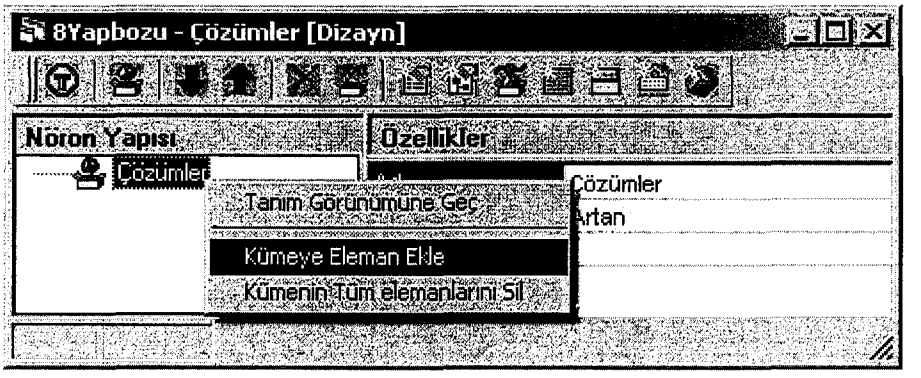


Şekil 3.10 Çözüm Kriterleri

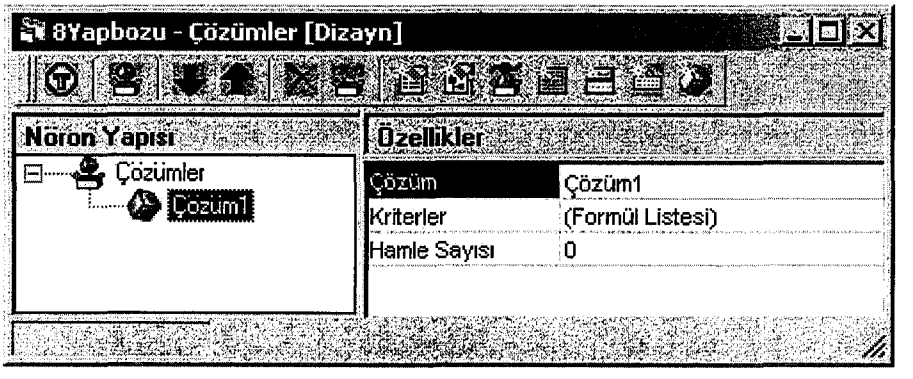
Artık problem tümüyle tanımlanmıştır ve "Çözüm1" gerçek nesnesi türetilebilir.

3.4.7 "Çözüm1" gerçek nesnesinin türetilmesi

Yapılması gereken tek iş "Çözümler" Küme nesnesine "Kümeye Eleman Ekle" komutunun verilmesidir. Bu komut işletilebilir Eleman nesnesini üretmekle kalmayacak, bu eleman için Eleman Kütüphanesi'ni de oluşturacaktır.



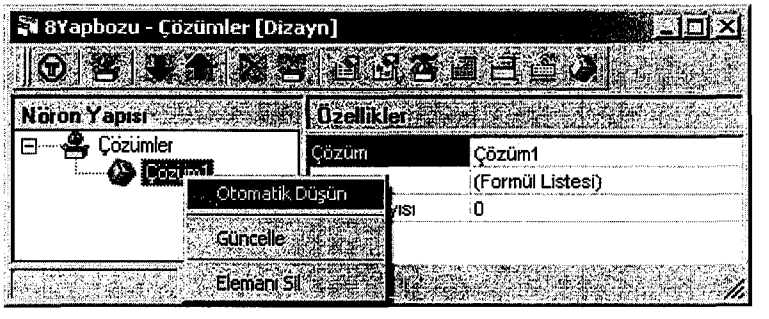
Şekil 3.11 Eleman türetilmesi



Şekil 3.12 Türetilen "Çözüm1" Eleman nesnesi

3.5 Problemin Çözdürülmesi

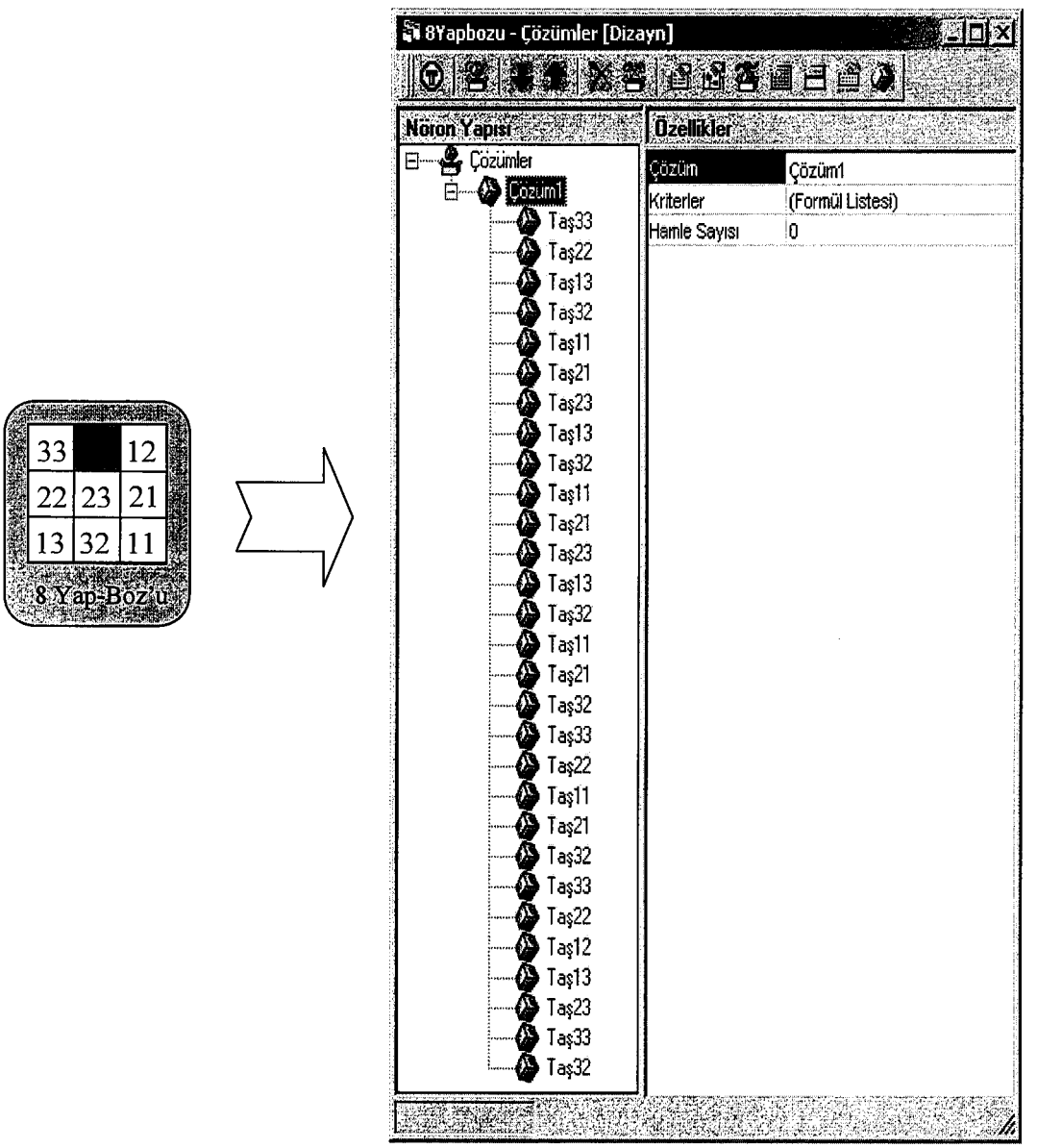
"Çözüm1" gerçek nesnesine "Otomatik Düşün" komutu verilir.



Şekil 3.13 Arama işleminin başlatılması

3.6 Çözümün (Öneri Planın) görüntülenmesi

Düşünme işlemi bittiğinde arayüz otomatik olarak tanımlamalar bölümüne geri döner ancak üretilen plan "Çözüm1" nesnesi altında görüntülenir.



Şekil 3.14 Üretilen Öneri Plan

Şekil 3.14 de "Çözüm1" nesnesi altında görülen taş dizisi sıra ile oynandığında problemin çözüldüğü görülecektir. Bu bir Öneri Plan dır ve tanımlama bölümünde bozulmuş olarak verilen 8 Yap-Boz'u probleminin mümkün olan en zeki çözümünü içerir.

4 TARTIŞMA, SONUÇ VE ÖNERİLER

4.1 ABS'in Bilinen Arama Algoritmalarını Kullanması

ABS in arama algoritmasının kullanacağı strateji tamamen yapılan tanımlamalara bağlı olacaktır. Bu anlamda yukarıda açıklanan 8 Yap-Boz'u problemi referans alınarak tanımlamalarda yapılan ufak değişikliklerle ABS in bilinen bilgilendirilmiş ve bilgilendirilmemiş arama stratejilerini nasıl kullandığı üzerinde durulacaktır.

Bilgilendirilmiş veya bilgilendirilmemiş olsun genel olarak bir arama stratejisinin nasıl işleyeceği yeni açılan budakların ilerleyen aşamalarda hangi sıra ile işleme tabi tutulacaklarını belirleyen bir kuyruk(queue) yapısı tayin eder. ABS ise kuyruk yapısı yerine esnek bir sıralama metodu kullanır. Sıralamaya referans teşkil eden verilerin(puan) niteliği aslında bu sıralama yapısını sıralı bir veri kaynağı olarak kullanabileceği gibi bir kuyruk yada yığın(stack) gibi de kullanabilmektedir. Bu sayede ABS kullanıcının kriter ifadelerini tanımlama tarzından bir strateji ortaya çıkarır.

Arama stratejileri 4 temel kriter ile değerlendirilirler :

- **Tamlık** : Strateji eğer var ise mutlaka çözümü bulur.
- **Optimallik** : Strateji birden fazla çözüm var ise en iyisini bulur.
- **Zaman Gereksinimi** : Bir çözümün bulunmasının ne kadar zaman alacağı.
- **Yer Gereksinimi** : Bir çözümün bulunmasının ne kadar hafıza kullanacağı.

Tamlık ve optimallik kavramları strateji sonucu bulunan çözüm ve çözümün niteliği ile ilgili olarak doğmuş kavramlardır ve bir strateji için tamdır yada tam değildir veya optimaldir yada optimal değildir denilebilir. Zaman ve yer gereksinimi kavramları ise hesaplama sonucu matematiksel bir ifade ile ifade edilebilirler. Arama ağacı içinde yeni budakların açılmasına dayanan algoritmada bir budağın açılmasının zaman ve hafıza yönünden bir maliyeti vardır. Bu maliyet budağın hafızada kapladığı yer, açılması için geçen süre ve toplam kaç budağın açılması gerektiği hesaplanarak bulunabilir. Stratejiler ise genellikle kaç budağın

açılacağını belirlediğinden zaman ve yer gereksinimi budak sayısıyla ilişkilendirilecek ve beraberce değerlendirilecektir[12].

4.1.1 Bilgilendirilmemiş(Uninformed / Blind) arama stratejileri

"Bilgilendirilmemiş Arama" deyimi mevcut durumdan çözüm durumuna kadar olan süreçte kaç budak açılacağına yada çözüm durumunun uzaklığına dair hiçbir verinin olmadığını ifade eder. Bu durumda arama çözüm durumuna ulaşılana dek mümkün olan her olasılığın denenmesi şeklinde yapılacaktır. Bu noktada dallanma faktörü ve çözümün bulunduğu durumun derinliği en kötü durum için :

b : Dallanma Faktörü
d : Çözüm derinliği

olmak üzere ;

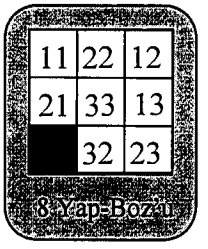
$1 + b + b^2 + b^3 + \ldots\ldots + b^d \Longrightarrow O(b^d)$

adet budak şeklinde kendini gösterecektir. Bu olabilecek en kötü durumdaki en fazla budak sayısıdır. Dallanma faktörü 10 olan bir problem için 1000 budak/saniye hızla işlem yapan bir bilgisayarın 10 hamle derinliğinde olduğu varsayılan bir çözüme ulaşması en kötü durumda 128 gün(yaklaşık 4 ay ve bir hafta) sürer, 100 byte/budak lık bir hafıza ihtiyacına göre de 1 terabyte hafıza gerektirirdi. Arama stratejisi bu sayıyı daha aza indirgemek amacıyla değişik yöntemler kullanır.

4.1.1.1 Önce Enine Arama(Breadth-First Search)

Adından da anlaşılacağı üzere enine genişleyen bir ağaçta yapılan bir aramadır. Bu stratejide önce kök budağı, daha sonra kökten çıkan dalların budakları, daha sonra da onların budakları açılarak devam edilir. Genel olarak d derinliğindeki bir budağın açılması için d -1 ve daha yukarıdaki budakların tümünün açılmış olması gerekir. Önce Enine Arama en genel sistematik arama stratejisi olup derinliğe inişte açılan budakların maliyetlerinin eşit olduğu problemlerde optimal, kullanıldığı tüm problemlerde tam(complete)dır.

ABS içindeki uygulaması ise 8Yap-Boz'u problemi referans alındığında Çözüm nesnesinden "min Hamle Sayısı" ve sezgisel fonksiyon(heuristic) kriterleri devre dışı bırakıldığında uygulanacak stratejiye karşılık gelir. Bu durumda 8 Yap-Boz'u 16 ncı derinliğe ulaşmak için 8600 kadar budak açmıştır[1] ve muhtemelen 29 hamle derinlikteki çözümü bulmak için pratik zaman ve hafıza yeterli olmayacaktır. Bu nedenle en iyi çözümün 8 hamle derinlikte olduğu bilinen bir durum başlangıç durumu olarak alınmıştır. Diğer bilgilendirilmemiş arama stratejileri değerlendirilirken de referans alınacak olan bu bozulmuş durum Şekil 4.1 de gösterilmiştir.



Şekil 4.1 Optimal çözümün 8 hamlede bulunabildiği başlangıç durumu

Şekil 4.1'de gösterilen başlangıç durumu için ABS yaklaşık 270 budak açarak sonuca ulaşmıştır. Dallanma faktörü yaklaşık olarak 3 olan 8 Yap-Boz'u probleminde eğer 8 hamle derinlik için bir hesap yapılacak olursa :

$$1+3^2+3^3+3^4+3^5+3^6+3^7+3^8 = 9838 \text{ budak}$$

rakamı bulunacaktır. Çözümün 8. derinliğin başlarında bulunmuş olabileceği düşünülürse bu rakamın 3273 budağı mutlaka açılmış olmalıdır. Ancak ABS'in tekrar eden durumları yönetmesi sonucu sayı büyük ölçüde azalmıştır. Bulunan çözüm planına göre sırasıyla oynanması gereken taşlar :

[1] A.B.S. de tekrarlayan durumlar yönetildiğinden bu rakam O(b) şeklinde görülmez.

Taş32
Taş33
Taş22
Taş12
Taş13
Taş23
Taş33
Taş32

Şeklindedir ve optimal olduğu bilinen A* stratejisi ile de aynı çözüm planı üretilmiştir.

4.1.1.2 Düzenli Maliyet Araması(Uniform Cost Search)

Önce enine aramanın en sığ(derinlik açısından köke en yakın) çözümü bulduğu yukarıda açıklanmıştır. Düzenli Maliyet Araması aramada açılan budakların maliyetlerinin birbirinden farklı olması durumunda uygulanan Önce Enine Arama stratejisinin düzenlenmiş bir halidir. Aslında Önce Enine Arama için; budak açma maliyetleri hep aynı olduğundan herhangi bir derinlik için toplam maliyetin g(n) derinliğe(derinlik(n)) eşit olduğu özel bir Düzenli Maliyet Araması'dır da denilebilir. Daima maliyeti en düşük olan budak öncelikli olarak açılır. Belirli şartlar sağlandığında karşılaşılan ilk çözüm durumunun en iyi çözüm olacağı garantilenmiştir. Çünkü maliyeti daha düşük bir çözüm varsa onun daha önce açılmış olması gerekecektir.

8 Yap-Boz'u problemi üzerinde bu strateji için örnek vermek pek de mümkün değildir. Çünkü hamle maliyetleri daima eşittir. Ancak kriter tanımlamaları ile ABS'in Düzenli Maliyet Araması yapması fevkalade sağlanabilir.

4.1.1.3 Önce Derine Arama(Depth-First Search)

Daima çözümü öncelikle derinlikte arayan bir stratejidir. Artık yeni budak açmanın mümkün olmadığı en derin budak da açıldıktan sonra daha sığ budakların açılmasına geçilir. Bunun diğer bir açıklaması daima açılan bir budağın oluşturduğu ilk dal üzerine yeni bir budak açmak ve böylece en dibe kadar devam etmek şeklindedir. Zaman ve yer harcaması bakımından önce enine aramaya nazaran çözüme ulaşıldığı anda açılmış toplam budak sayısının çok daha az olma

ihtimali mevcut olmakla beraber en kötü durum için aynıdır. Fakat ne yazık ki Önce Derine Arama stratejisi derinliğin bir sınırı olmayan problemlerde çıkmaza girerek sonuç üretemeyebileceğinden tam kabul edilemeyeceği gibi, ulaşılan ilk çözümün en iyi olamama ihtimali de mevcut olduğundan optimal de değildir.

ABS içinde Önce Derine Arama stratejisi açılan en son budağın açılacaklar sıralamasında ilk sırayı alması için "max Hamle Sayısı" gibi bir kriter oluşturularak sağlanabilir. Bu kriter devreye sokulduğunda son açılan budak daima sonraki işlemde ilk açılacak budak olacaktır. Optimal çözümün 29 hamle derinliğinde olduğu başlangıç durumu için yapılan aramada tekrarlayan durumlardan dolayı ulaşılan ilk dip noktası yaklaşık 750 hamle derinliğinde olmakta bu bölgeden yukarı doğru çıkılırken karşılaşılan tekrarlı durumların sayısının çok fazla olmasından dolayı birim zamanda başarıyla açılan budak sayısı azalmakta, ilk çözümün bulunmasının ne kadar zaman alacağı bilinmemektedir. Optimal çözümün 8 hamle derinliğinde olduğu diğer başlangıç durumu için yapılan aramada ise durum değişmemiş arama 1020 nci derinliğe ulaşılıp hala çözüm bulunamadığından durdurulmuştur. Kaldı ki Önce Derine Arama stratejisinin derinliklerde kaybolma gibi bir dezavantajı olduğu da bilinmektedir.

4.1.1.4 Derinlik Limitli Arama(Depth-Limited Search)

Derinlik Limitli Arama, Önce Derine Arama stratejisinin bilinen derinlikte kaybolma probleminin aşılması amacıyla üretilmiş bir stratejidir. Arama belirli bir derinliğe ulaşıldığında daha fazla derine inmeden yüzeye doğru yönlendirilir. Bu stratejinin çözümün en fazla hangi derinlikte olabileceğinin bilindiği problem türlerinde etkili olduğu bilinmektedir. Derinlik Limitli Arama stratejisi verilen limitin çözümün bulunduğu derinlikten fazla veya eşit olduğu durumlarda tam, aksi durumlarda tam değildir. Yine stratejide optimallik yoktur.

ABS'de Derinlik Limitli Arama "Hamle Sayısı" özelliğine zorunlu bir küçüklük kriteri tanımlanarak sağlanabilir. Optimal çözümün 8 hamle derinliğinde olduğu başlangıç durumu için yapılan aramada bu limit 7 olarak verildiğinde çözümün bulunamadığı görülecektir. Limit değeri 8'e yükseltildiğinde ise çözüm optimal Önce Enine Arama ile bulunan çözümün aynısıdır. Fakat bu optimalliğin bir yanılsama olduğu ortadadır. Çünkü araştırıldığında bu planın aslında 8 hamle derinliğindeki ağaçtaki tek çözüm olduğu görülmüştür.

4.1.1.5 Ötelemeli Derinlik Araması(Iterative Deeping Search)

Derinlik Limitli Arama stratejisinin $O(b^{limit})$ budak olan avantajlı zaman ve yer gereksiniminden çözüm derinliğinin bilinmediği problemlerde de yararlanılabilmesi amacıyla geliştirilmiş olan Ötelemeli Derinlik Araması stratejisi derinlik limitinin kademeli olarak arttırıldığı bir Derinlik Limitli Arama stratejisinden ibarettir. Ötelemeli Derinlik Araması stratejisi aynı zamanda ötelemeli olarak arttırılan limit değerinden dolayı Önce Derine Arama ve Önce Enine Arama stratejilerinin avantajlarını birleştirerek tam ve optimal bir arama stratejisi ortaya koyar. Ötelemeli Derinlik Araması stratejisi limitin her artışında aramaya yeni baştan başladığından birçok budak yeniden açılacaktır. Bu durum ilk bakışta bir dezavantaj gibi görünse de üssel olarak büyüyen bir ağaç yapısında budakların neredeyse tümü alt dallardadır ve bu dezavantaj önemini yitirir.

Limit değerinin her ötelemede 1 birim arttırılması Önce Enine Arama stratejisi ile Ötelemeli Derinlik Araması stratejisinin arasındaki fakın anlaşılmasına engel olabilir. Ancak limitin her ötelemede 1 birimden daha fazla arttırılabiliyor olması konuya açıklık getirecektir.

Ötelemeli Derinlik Araması stratejisinin ABS'de uygulanması biraz ilginçtir. Limit bir Özellik nesnesi olarak "Çözüm" Tanım nesnesine eklenebilir. Ancak mevcut limite ulaşıldığında çözüm bulunamamışsa çözümün bulunamayacağı değerlendirilip arama durdurulacaktır. İlk bakışta "Limit" özelliğinin değerini arttıracak olan etkinin hangi nesneye tanımlanacağı tahmin edilemeyebilir. Fakat ABS'in ilginç sayılabilecek bir özelliği sayesinde Tepe Eleman nesnesine tanımlanan etkiler "açılacakDurumlar" dizisi boşalıp hiçbir çözüm bulunamadığı durumlarda uygulanarak arama yeni baştan başlatılır. Tepe Eleman nesnesi kök durumunda da zaten var olduğundan hiçbir zaman bir başka nesne tarafından seçilmeyecek, dolayısıyla yukarda açıklanan şartlar hariç etkileri hiçbir zaman uygulanmayacaktır. İşte normal şartlarda etki tanımlanması anlamsız gibi görünen Tepe Eleman nesnesine etki tanımlanabilmesinin altında yatan mantık budur.

Yukarıda açıklanan işlemler yapılıp çalıştırıldığında ABS yine stratejinin gereği olan optimum çözüme 8 nci derinlikte ulaşmıştır.

4.1.1.6 Çift Yönlü Arama(Bidirectional Search)

Çift Yönlü Arama stratejisinin altında yatan teori aynı zamanda arama işlemine başlangıç durumundan ileriye ve çözüm durumundan geriye doğru başlamak ve bu iki ayrı arama ağacının dallarının kesiştiği ilk noktada başlangıçtan çözüme giden bir yol yakalamak şeklindedir. Detaylı bir inceleme yapıldığında her iki çıkış noktasından da toplam yolun yarısı kat edilerek çözüme ulaşıldığı görülür bu ise $O(b^d)$ yerine $O(2b^{d/2})$ gibi bir budak gereksinimi demektir ki iki ifadenin alabileceği sonuçlar itibarı ile aralarındaki fark Çift Yönlü Arama lehine muazzamdır. Her iki yönde de tam ve optimal olduğu bilinen Önce Enine Arama stratejisi kullanıldığında Çift Yönlü Arama da tam ve optimal olacaktır. İlk bakışta harika bir teori olarak görünen Çift Yönlü Arama stratejisi maalesef sadece belirli problem türlerinde kullanılabilmektedir.

Uyumluluğun sağlanmasının programlama açısından çok zor olmayacağı değerlendirilmekle beraber ABS'in mevcut ilkel hali Çift Yönlü Arama stratejisini desteklememektedir.

4.1.2 Bilgilendirilmiş(Informed / Heuristic) Arama Stratejileri

Bilgilendirilmemiş arama stratejileri üzerinde yukarıda yapılan incelemelerde görülmüştür ki ABS hemen hemen bilinen tüm bilgilendirilmemiş arama stratejilerini başarı ile uygulayabilmesine rağmen kombinasyon patlamasının önüne geçebilmek adına stratejilerin içerdiği dezavantajları da beraberinde taşımaktadır. Optimal çözümün 29 hamle olduğu başlangıç durumu için hiçbir bilgilendirilmemiş arama stratejisi kabul edilebilir bir süre içinde çözüm üretememiş, araştırmanın sonuçlarının alınabilmesi için Optimal çözümün 8 hamle derinlikte olduğu bilinen bir başka başlangıç durumu referans alınmıştır.

Zifiri karanlık, bomboş ve büyükçe bir odada bir sigara paketini bulmaya çalışmak ile yanmakta olan bir sigara izmaritini bulmaya çalışmak arasındaki farkı bir düşününüz. Yanan sigara yerini belli edeceğinden hemen bulunacaktır. Burada yanan sigaranın yaydığı ışık probleme özel bir bilgidir ve küçük olmasına karşın tüm odayı arşınlamaktan kurtaracaktır. İşte bilgilendirilmiş arama stratejilerinin altında yatan teori de budur. Eğer ağaç yapısı içinde çözümün bulunduğu yere dair

ufacık bir ip ucumuz olsaydı durum uzayındaki birçok budağı gereksiz yere açmaktan kurtulabilirdik.

4.1.2.1 Önce En İyi Arama(Best-First Search)

İyi tanımlanmış bir problemin çözümü üzerinde çalışırken genel bir arama stratejisi kullanılması ve sadece probleme özel bilginin bir sonra hangi budağın açılacağı kararının verildiği kuyruk fonksiyonuna yerleştirilmesi Önce En iyi Arama stratejisinin en genel ifadesidir.

Bir sonra açılacak budağın gerçekten en iyi olduğunu kesin olarak bilmek aslında bir arama uygulamaktan ziyade çözüme doğrudan götüren bir yöntem kullanmak anlamında olacaktır. Oysa Yapay Zeka problemlerinde en iyi budağın hangisi olduğunu net bir şekilde ifade eden bir fonksiyon sadece hayalden ibarettir. Eğer böyle bir fonksiyon bulunabilseydi problem zaten deterministik olur, Yapay Zeka araştırmalarının ilgi alanı dışında kalırdı. Burada bahsedilen probleme özel bilgi sadece bir ip ucu mahiyetindedir, çözümün yerini yada ona olan uzaklığı çok genel bir açıda gösterir. Diğer bir deyişle bulanıktır ve en iyi budağı değil iyi olduğu değerlendirilen bir gurup budağı işaret eder. Arama da bu budaklar öncelikli tutularak yapılır.

Bilgilendirilmiş arama stratejilerinde Önce En İyi Arama stratejisi kendi içinde sınıflandırılan bir arama stratejisidir. Bundan sonra bu sınıflandırmadan doğan alt strateji türlerine değinilecektir.

Tahmin edilen toplam masrafın minimize edilmesi(Greedy Search)

Önce En İyi Arama stratejilerinden en basiti çözüme olan toplam maliyetin minimize edilmesidir. Bu çözüme en yakın olduğu sezgisel fonksiyon(probleme özel bilgi / Heuristic) ile tahmin edilen budağın öncelikli olarak açılması anlamına gelmektedir. Birçok problemde çözüm durumuna olan uzaklık kesin olarak bilinmemekle beraber tahmin edilebilmektedir. İşte bu tahmini yapan fonksiyona sezgisel fonksiyon(heuristic function) denir ve genellikle "h" sembolü ile temsil edilir.

Tahmin edilen toplam masrafın minimize edilmesi(GS) arama stratejisi Önce Derine Arama stratejisinde olduğu gibi çözüme doğru bir çizgi halinde arama yaparken sadece bir çıkmaz son ile karşılaştığında geriye döndüğünden Önce

Derine Arama stratejisinin dezavantajlarını taşır. Optimal değildir. Tekrarlayan durumlar yönetilmediğinde sonu olmayan bir yolda kaybolacaktır. Bu yüzden tam olmadığı değerlendirilmektedir. Ancak ABS içinde tekrarlayan durumlar stratejiden bağımsız olarak daima kontrol edildiğinden GS'in ABS içindeki uygulamalarında strateji tamdır.

ABS içinde h fonksiyonları rahatlıkla tanımlanabilmektedir. Bunun bir örneği "Örnek Uygulama" bölümünde 8 Yap-Boz'u problemi için "sezgisel metot" adı altında verilmiş olan taşların orijinal konumlarına olan toplam blok uzaklıklarının çözüme olan uzaklığa dair bir ip ucu olarak değerlendirilmesidir. Aslında yerlerinde olmayan taşların sayısı da bir h fonksiyonudur.

h fonksiyonlarının kalitesi önce açılmasını önerdikleri budakların sayısının azlığı ile değerlendirilir. Örnekte kullanılan h fonksiyonu "yerlerinde olmayan taşların sayısı" fonksiyonuna nazaran çok daha az budağı işaret edeceğinden daha etkin bir h fonksiyonudur.

8 Yap-Boz'u problemi üzerinde GS'in uygulaması "Çözüm nesnesinin kriterlerinden "min Hamle Sayısı" kriteri iptal edildiğinde oluşur. Bu durumda sadece h fonksiyonu devrede kalacak ve arama bu h(n) ile yapılacaktır.

Optimal çözümün 29 hamle derinliğinde olduğu bilinen bir başlangıç durumu için aynı bilgisayar kullanılarak yapılan denemelerde Optimal çözüme ulaşan strateji 1 dk 56 sn içinde 29 hamlelik bir çözüm üretirken GS ile ilk çözüme 4 sn gibi kısa bir sürede ulaşılmış fakat çözüm 55 hamlelik bir çözüm önerisi olmuştur.

Tahmin edilen toplam yolun minimize edilmesi(A* Search)

GS tahmin edilen toplam masrafı minimize eden h(n) fonksiyonunu kullanarak arama ağacını büyük ölçüde budamıştır. Fakat ne yazık ki bulunan çözüm optimal değildir. Diğer taraftan Düzenli Maliyet Araması yolu minimize eden[1] g(n) fonksiyonu ile optimal ve tam, fakat etkin olmayan bir strateji ortaya koymaktadır. Eğer bu iki strateji, avantajlarını beraberce kullanabilmek amacıyla birleştirilebilseydi doğrusu oldukça iyi olurdu. Sevindirici olan bunun basitçe g(n) ve h(n) fonksiyonlarının aritmetik toplamı ile sağlanabiliyor olmasıdır.

[1] 8 Yap-Boz'u probleminde Düzenli Maliyet Araması problemin özelliğinden, birim yol maliyetinin tüm budaklar için eşit olmasından dolayı Önce Enine Arama stratejisine karşılık gelmektedir.

$f(n) = g(n) + h(n)$

Bu sayede en etkin, optimal ve tam bilgilendirilmiş arama stratejilerinden biri olan A* stratejisi ortaya çıkmaktadır. Tek koşul ise h(n) fonksiyonunun toplam maliyeti olabileceğinden daha az hesaplamamasıdır. 8 Yap-Boz'u probleminde kullanılan h fonksiyonu incelenirse bulunacak blok mesafesi fiziksel olarak daha kısa olamaz. Dolayısıyla kullandığımız h(n) asla maliyeti olabileceğinden daha az hesaplamayacaktır.

"Örnek Uygulama" bölümünde gösterilen ve örnek olarak kullanılan başlangıç durumu için optimal çözümün 29 hamle derinliğinde olduğu bu strateji sayesinde bilinmektedir. Kaldı ki "Örnek Uygulama" bölümünde kullanılan strateji de A* stratejisi olduğundan bu bölümde ABS uygulamasına değinilmeyecektir.

A* stratejisinin geliştirilmesi çalışmaları sonucu yer gereksinimini azaltmak maksadı ile IDA* (Ötelemeli Derinlikli A*) ve SMA*(Basitleştirilmiş Hafıza Bağımlı A*) gibi stratejiler de geliştirilmiştir.[12] Bu stratejilerin uygulamaları ABS'de mümkün olmakla beraber sağladıkları yer gerksinimi avantajlarından ABSin tekrar eden durumları yönetme stratejisinin belirli olmasından dolayı yararlanılamamaktadır. Bu durum ayrıca bir araştırma konusu olabilir. Yer gereksinimini azaltmaya yönelik olarak geliştirilmiş bu arama stratejileri asıl ilgi alanının aşırı bilgilendirilebilme ile kombinasyon patlamasının engellenmesi olan ABSlerdeki uygulamalarında A* stratejisine olan benzerliklerinden dolayı araştırma kapsamı dışında tutulmuşlardır.

4.2 ABS'in Çalışma Performansı

Gerçekte çalışma performansını etkileyen en önemli faktör tanımlamaların arama algoritmasına taşınırken ne dercede derlendiğidir. Bir derleme işlemi en alt düzeyde donanım seviyesinde fiziksel olarak yapılabileceği gibi daha üst seviyelerde mikroişlemcinin mikrokodları düzeyinde, işletim sisteminin sunduğu hazır geliştirme kodları düzeyinde yada J-Plan'da olduğu gibi uygulama geliştirme

ortamının sunduğu kolaylıklar düzeyinde olabilir. Bu düzeyin tabana ne dercede yakın olduğu performansın da belirgin bir göstergesi olacaktır.

Tanımlamaların derlenme düzeyi ABS için geliştirilmeyi bekleyen en önemli yönüdür. Bu tezde sadece teori savunulmakla beraber yine de hazırlanan yazılımın yeterince bilgilendirildiğinde kabul edilebilir sürelerde birçok problem için çözümü etkinlikle bulabileceği değerlendirilmektedir. Örneğin 2 Ghz AMD XP mikroişlemcisi, 133 Mhz FSB'I, 256 MB DDR RAM'I ile bir test donanımı 8 Yap-Boz'u probleminin 29 hamle derinliğindeki optimal çözümünü A* stratejisi ile 1 dk 56 sn gibi tatminkar sayılabilecek bir sürede bulabilmiştir. Optimallikten ödün vermek kaydıyla çözümler saniyeler içinde bulunabilmektedir.

Uygulama geliştirme ortamının da yazılan kodu derlemesinde kodun hızlı çalışması açısından etkinliği değerlendirilmesi gereken bir başka konudur. Visual Basic derleyicisinin derlediği bir kodun C++ derleyicisi ile derlenmiş bir koddan yaklaşık 5 kat daha yavaş çalıştığı bilinmektedir.

4.3 Geliştirilebilirlik

4.3.1 Kullanıcı ile etkileşim

ABS in ilk örnegi olan J-Plan problem tanımlamalarında 2. Bölümde detayları açıklanan yapıyı kullanmaktadır. Bu yapı için oluşturulmuş arayüzler ise 3. Bölümde örnek uygulama açıklanırken çeşitli yerlerde şekillerle gösterilmiştir. Genel olarak nesnel bir yalkaşım ile problem tanımlamanın kullanıcıya rahatlık sağlayacağı değerlendirilmiştir. Ancak bazı bölümlerde teorinin bir yazılıma dönüştürülmesinin zor, zaman gerektiren ve tümüyle önceden planlanmasının neredeyse imkansız bir iş olması, daha da önemlisi beyin fırtınası ortamının yaratılabildiği, kodlama yükünün paylaşılabileceği bir gurup çalışması gerektirmesi nedeni ile başlangıç aşamasında yapılan hatalar yamalarla kapatılarak bugünkü noktaya gelinmiştir. Bu durum bazı arayüzlerdeki stillerin tartışmaya açık olması durumunu doğurabilmektedir. Örneğin Tanım Zinciri'nin oluşturulması esnasında Bağlı Küme nesneleri kullanmak yerine işlemin görsel olarak da yapılabileceği iddia edilebilir.

Diğer taraftan tamamen görsel olmayan, kod yaklaşımı da mümkündür. Kod yaklaşımı 8 Yap-Boz'u probleminin tanımlanması üzeride şu şekilde örneklenebilir.

```
Proje 8YapBozu
    Proje.YeniTanım("Konum", "Konumlar")
        Konum.YeniÖzellik("No", Sabit, Sayı)
        Konum.YeniÖzellik("EndeksY",  Sabit, Sayı)
        Konum.YeniÖzellik("EndeksX",  Sabit, Sayı)
        Konumlar.YeniEleman("Konum11")
            Konumlar.Konum11.No = 1
            Konumlar.Konum11.EndeksY = 1
            Konumlar.Konum11.EndeksX = 1
        Konumlar.YeniEleman("Konum12")
            Konumlar.Konum12.No = 2
            Konumlar.Konum12.EndeksY = 1
            Konumlar.Konum12.EndeksX = 2
        .........
        .........
    Proje.YeniTanım("Taş", "Taşlar")
        Taş.YeniÖzellik(.....)
        ........
        ........
        Taşlar.YeniEleman("Taş11")
        ........
        ........
    Proje.YeniTanım("Çözüm", "Çözümler")
        Çözüm.YeniÖzellik("Hamle Sayısı", Değişebilir, Sayı)
        Çözüm.KümeBağla("Oynanan Taş", Taşlar)

    Taş.YeniEtki("Hamle Sayısını Arttırır",Çözüm.Hamle Sayısı ++)
    Oynanan Taş.YeniKriter("Yanında boşluk olmalı", .....)
    Çözüm.YeniKriter("Çözüm durumu", .....)
    Çözüm.YeniKriter("min Hamle Sayısı", .....)
    Çözüm.YeniKriter("Heuristic", .....)

    Çözümler.YeniEleman("Ayıldız")
Son.

>Ayıldız.OtoDüşün
```

Bir diğer husus Üretilen Plan'ın gösterimidir. Bu işlem halihazırda tanımlamalar ışığında bir ağaç görünümü şeklindedir. Oysa çözüm yazıcıdan çıktısı alınabilir bir form halinde de olabilirdi. Bu durumda çıkış formunun şeklinin ve veri alanlarının konumlarının da tanımlanması gerekecektir. Biraz daha yaratıcılığı zorlarsak aslında tüm tanımlamalar ve veri girişleri bir form üzerinde yapılabilir.

4.3.2 Elastikiyet

ABS için yeterince elastiki bir yapı ön görülmüş olmasına karşın hala elastikiyetin arttırılması mümkündür. Örneğin ABS içinde mutlak değer alan bir işlem tanımlı değildir. Bu yüzden gerektiğinde ifadenin karesinin kare kökü alınmaktadır. Oysa gelişmiş programlama dillerinde olduğu gibi yeni operatör, fonksiyon hatta alt yordam tanımlama imkanları kullanıcının seçenekleri arasında sunulabilir.

Bir diğer konu ise gerçek nesnelerin parametre olduğu aritmetiksel işlemler yapılamamaktadır. Örnegin "Konum11 + 1" ifadesi hatalı olarak değerlendirilecektir. ABS aritmetik işlemlerin "Çözüm1.Hamle Sayısı + 1" ifadesindeki gibi Özellik nesneleri ile yapılmasına müsade eder. Oysa gerekli düzenlemeler yapılarak "Konum11 + 1" ifadesinin "Konum12" sonucunu döndürmesi sağlanabilir. Bu geliştirme olmazsa olmaz değildir çünkü "Konum" Tanım nesnesine "Sonraki Konum" adlı nesne değer tipli bir Özellik nesnesi eklenerek de istenilen yapılabilir fakat yukarıdaki yöntem problem tanımlamalarında elastikiyeti büyük ölçüde arttıracaktır.

4.3.3 Ağ içinde dağıtık kullanım

Ağ içinde ABS kullanımının geliştirilmeyi bekleyen iki farklı yönü vardır. Mevcut sistem tek bir bilgisayar üzerinde çalışması için geliştirilmiş bir sistemdir. Fakat gerçekte problemin ağ yapısı içinde dağıtık olarak da tanımlanabilmesi sistem etkinliğini arttıracaktır. Tanımlamalar dışında gerçek nesneler de ağ içinde dağıtık olarak saklanabilir. Özellik değerlerinin değişik departmanlarca girişinin sağlandığı ana planlama işleminin ise bir merkezden bu bilgilerin toplanarak yapıldığı bir dağıtık sistemin yaratılabileceği ön görülebilir.

Ağ içindeki kullanımın diğer bir yönü işe arama algoritmasının yükünün ağ komşularına dağıtılabileceğidir. Bu durumda ana kontrol yine bir merkezde olacak, Tepe Elemanın açılan ilk budakları, tekrarlamaların kendi içinde yönetileceği fakat tanımlama kriterleri itibarı ile aynı prensiple çalışacak ağdaki diğer bilgisayarlarda çalışan ABSlere paylaştırılarak arama algoritması yükü hafifletilebilir. Bu yöntem burada ana hatları açıklanan teorinin üzerinde daha detaylı çalışmaların yapılmasını gerektirecektir.

4.3.4 Performans geliştirmesine donanımsal yaklaşım

Bölümün başlarında ABS'in çalışma performansı ile ilgili yapılan açıklamalardan da anlaşılabileceği gibi ABS'in yazılımı geliştirilerek şu andaki halinden binlerce kat hızlı çalışabilecek yeni bir sistem yapılandırılabilir. Burada kullanılan "binlerce kat" ifadesi abartılı bulunmamalıdır. Çünkü mevcut yazılım kriter ve etki değerlendirmelerinde karakter katarı işlemleri ile değerlendirme yapmaktadır. Bu yolun ise kolay anlaşılabilir fakat çok dolambaçlı ve yavaş bir yol olduğu bilinmektedir. Ayrıca kod incelenirse işaretçilerin gerçekte dolaylı olarak kullanıldıkları görülecektir. Oysa doğrudan gerçek işaretçilerin kullanımı çoğu kod parçasında mikroişlemcinin adresleme komutlarına bire bir karşılık gelmektedir.

Bir diğer taraftan yeni durumun puanına göre kuyrukta uygun yerine yerleştirilmesi bu yerin altında kalan verilerin hafızada ötelenmesini gerektiren zaman alıcı bir işlemdir. Bu işlemin hızlı yapılabilmesi için Windows API'nin "CopyMemory" rutini doğrudan kullanılmıştır. Her ne kadar hızlı çalıştıkları iddia edilse de API rutinleri genel amaçlar için hazırlanmışlardır ve özel kullanım amacının dışında gereksiz genelleme işlemleri de yaptıkları bilinir.

Tekrarlayan durumların kontrolü ise başlı başına bir işlem ve hafıza canavarıdır. Öyle ki durumlar uzayında yaratılan durumların tümü üzerlerinde sadece bu işlem için gerekli bilgiler bırakılacak şekilde küçültüldükleri halde bile sayıca çok fazla olduklarından hafızayı oldukça fazla işgal etmekte, yeni durumun ise tekrarlayıp tekrarlamadığı tüm bu durumlar uzayı içinde değerlendirilmektedir. Bu ise her yeni budak için yoğun işlemleri gerektirir.

Bir PC'de hafızanın uygulamalar arasında paylaşımını yapmak işletim sisteminin temel görevlerinden biridir.[14] Genel kullanıma yönelik işletim sistemi hafıza tahsis rutinlerinin özel bir işleme yönelik olarak düzenlenmesi durumunda gerçekte çok daha hızlı çalışabilecekleri değerlendirilmektedir.

Yukarıda açıklanan performans problemlerinin en etkin aşılma yönteminin ABSin bir PC bilgisayarda değil de ABS'lere yönelik olarak özel üretildiği donanımlarda kullanılması olacağı önerilmektedir. Bu donanım özel bir işlemci, özel bir hafıza ve veri yolu tasarımını gerektirebilir. Bu özel donanımda bir yeni budağın tekrarlama kontrolünün tek bir mikroişlemci komutu ile sağlanabileceği, durum kıymetlendirmesinin yine bir dizi mikroişlemci komutu ile sağlanabileceği, kuyrukta yeni durumun uygun yerine yerleştirilmesinin de yine bir mikroişlemci komutu ile sağlanabileceği önerilmektedir. Özel donanımın çok işlemcili ve ultra büyük hafızaya sahip hızlı bir donanım olması etkinliği arttıracak bir diğer nitelik olacaktır.

KAYNAKLAR

[1] BELLMAN, R. E., *An Introduction to Artificial Intelligence: Can Computers Think?*, Body & Fraser Publishing Company, San Francisco, (1978).

[2] CREVIER, D., *Artificial Intelligence: The Tumultuous History of the Search for AI. Basic Books*, New York (1991).

[3] BRUDNO, A. L., *Bounds and valuations for shortening the scanning of variations,* Problems of Cybernetics, **10**:225-241 (1963).

[4] BRATMAN, M. E., *Intention, Plans and Practical Reason*, Harvard University Pres, Cambridge, Massachusetts (1987).

[5] BRATMAN, M.E., *Planning and the Stability of Intention* (1992).

[6] BARWISE, J., *Everyday Reasoning and Logical Inference,* Behavioral and Brain Sciences (1993).

[7] PENROSE, R., *The Emperior's New Mind: concerning computers, minds, and the laws of physics*, Behavioral and Brain Sciences (1990).

[8] SACERDOTI, E. D., *A Structure for Plans and Behavior* (1977).
SACERDOTI, E. D., *Nonlinear Nature of Plans.(Conference textbooks)*, (1975).

[9] CORMEN, T.H., LEISERSON, C.E., RIVEST, R.L., ve STEIN, C., *Introduction to Algorithms*, The MIT Press Cambridge Massachusetts (2001).

[10] YANIK M., *Microsoft Visual Basic For Windows 95/98 NT: Visual Basic ile görsel programlama*, Beta Basım Yayım Dağıtım A.Ş., İstanbul (2000).

[11] APPLEBY, D., ve VANDEKOPPLE, J.J., *Programming Languages Paradigm and Practice*, McGraw-Hill Programming languages series (1997).

[12] RUSSEL, S.J., ve NORVIG, P., *Artificial Intelligence A Modern Approach*, Prentice Hall Series, New Jersey, USA (1995).

[13] DODGE, M., ve STINSON, C., *Running Microsoft Excel 2000*, Microsoft Press, Redmond, WA (1993).

[14] SAATÇİ, A., *Bilgisayar İşletim Sistemleri*, Meteksan yayınları, Ankara (1993).