ANADOLU ÜNİVERSİTESİ

FİNANSAL ZAMAN SERİLERİNİN FONKSİYONEL YAPISININ GENETİK ÖĞRENMEYLE BELİRLENMESİ

Özgür İCAN

(Doktora Tezi)

Eskişehir, 2013

FİNANSAL ZAMAN SERİLERİNİN FONKSİYONEL YAPISININ GENETİK ÖĞRENMEYLE BELİRLENMESİ

Özgür İCAN

DOKTORA TEZİ

İşletme Anabilim Dalı

Danışman: Prof. Dr. Hasan DURUCASU

Eskişehir

Anadolu Üniversitesi Sosyal Bilimler Enstitüsü

Temmuz, 2013

JÜRİ VE ENSTİTÜ ONAYI

Özgür İCAN'ın "Finansal Zaman Serilerinin Fonksiyonel Yapısının Genetik Öğrenmeyle Belirlenmesi" başlıklı tezi **19 Ağustos 2013** tarihinde, aşağıdaki jüri tarafından Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca, **İşletme (Sayısal Yöntemler)** Anabilim Dalında Doktora tezi olarak değerlendirilerek kabul edilmiştir.

İmza

Üye (Tez Danışmanı)	:	**Prof.Dr.Hasan DURUCASU**
Üye	:	**Prof.Dr.Emel ŞIKLAR**
Üye	:	**Prof.Dr.Muzaffer KAPANOĞLU**
Üye	:	**Doç.Dr.Metin COŞKUN**
Üye	:	**Doç.Dr.Nuray GİRGİNER**

Prof.Dr.B.Zafer ERDOĞAN
Anadolu Üniversitesi
Sosyal Bilimler Enstitüsü Müdürü

Doktora Tez Özü

FİNANSAL ZAMAN SERİLERİNİN FONKSİYONEL YAPISININ GENETİK ÖĞRENMEYLE BELİRLENMESİ

Özgür İCAN

İşletme Anabilim Dalı

Anadolu Üniversitesi Sosyal Bilimler Enstitüsü, Temmuz 2013

Danışman: Prof. Dr. Hasan DURUCASU

Menkul kıymet fiyatlarının piyasanın geçmiş verileri kullanılarak tahmin edilmeye çalışılması her zaman ilgi çekici ve bir o kadar da zor bir uygulama alanı olmuştur. Geçmiş piyasa verilerine dayalı olarak tahminleme yapmada başvurulan veri odaklı yaklaşımlardan biri de genetik programlamadır (GP). GP, finansal zaman serilerini oluşturan sürecin matematiksel formunu araştırmada sıklıkla kullanılmaktadır. Yanı sıra, GP piyasaya giriş ve çıkış sinyalleri üreten kural tabanlı sistemlerin geliştirilmesinde de bir örüntü tanıma tekniği olarak son derece umut verici sonuçlar elde etmektedir. Menkul kıymet fiyatını tahminleyen matematiksel formun ya da kurallar bütününün tamamen bilgisayar tarafından araştırılmasında çok büyük bir arama uzayıyla karşı karşıya kalınmaktadır. Bu bakımdan, mutlak bir çözümden ziyade sınırlı hesaplama gücüyle makul sürelerde yeterince iyi çözümler bulunabilmesi ancak GP gibi meta-sezgisel teknikler yardımıyla gerçekleştirilebilmektedir.

Bu çalışma kapsamında bu doğrultuda üç sistem geliştirilmiştir. Geliştirilen bu sistemlerden ilk ikisi sembolik regresyon uygulaması diğeri ise kural tabanlı sistem uygulaması niteliğindedir. İlk iki sistemde, basit terminal ve fonksiyon kümelerinden sembolik regresyon yardımıyla tahmin edici matematiksel modeller türetilmeye çalışılmaktadır. Ardından bu yaklaşımın güçlü ve zayıf yönleri okuyucuya sunulmuştur. Üçüncü uygulama kapsa-

mında geliştirilen sistem ise, hiçbir ön bilgiye sahip olmaksızın yalnızca tarihi verileri kullanarak piyasaya giriş ve çıkış zamanlamalarının belirlenip belirlenemeyeceğini araştırmaktadır. Belirli bir deney tasarımı uyarınca sistemin spekülasyon performansı değişik öğrenme ve test periyotları için sınanmıştır. Sonuç olarak, GP yardımıyla bu kural tabanlı sistemin 6 ay (120 işlem günü) ve 12 aydan (240 işlem gününden) oluşan test vadelerinde al-bekle stratejisinden her seferinde üstün stratejiler geliştirebildiği istatistiksel olarak ortaya konulmuştur. Ancak; 1 hafta (5 işlem günü), 1 ay (20 işlem günü) ve 3 ay (60 işlem günü) gibi görece kısa test periyotlarında böyle bir bulguya rastlanamamıştır. Deneyler açıkça göstermiştir ki; sistem, hiçbir ön bilgiye sahip olmadan, yalnızca tarihi günlük verilerden hesaplanan kimi teknik analiz göstergelerini içeren kural kombinasyonlarını keşfederek, orta-uzun vadede kârlı al-sat stratejileri geliştirebilmektedir.

Anahtar Kelimeler: evrimsel hesaplama, genetik programlama, sembolik regresyon, kural tabanlı sistemler, finansal zaman serileri, teknik analiz, etkin piyasa hipotezi

Abstract

DETERMINING THE FUNCTIONAL STRUCTURE OF FINANCIAL TIME SERIES BY MEANS OF GENETIC LEARNING

Özgür İCAN

Department of Business Administration

Anadolu University, Graduate School of Social Sciences, July 2013

Adviser: Prof. Dr. Hasan DURUCASU

Forecasting of security prices based on historical data has been a very popular and equally demanding application area. Genetic programming (GP) is one of the such data-driven techniques which has been employed in predicting security prices based on historical data. GP has been frequently used in searching the mathematical form of the process that generates financial time series. Besides; GP has been also reported to achieve very promising results as a pattern recognition technique in developing rule based systems which produce signals for entering and leaving the markets. When searching for the predictor mathematical form or for a rule pattern by the machine, an ill-defined optimization problem with a very large search space has to be faced with. Thus; instead of a deterministic one, a satisfying approximate solution within accepted time by limited computing power can only be found by metaheuristics such as GP.

To this aim, three systems have been developed throughout this work. The first two of these systems are symbolic regression applications and the other one is characterized as a rule-based system. The first two systems strive to generate predictor mathematical models by the help of symbolic regression from simple terminal and function sets. Then strong and weak sides of this approach is presented. The developed system within the scope of third application, searches whether in/out timing decisions for markets can solely be

determined depending on historical data without "à priori" knowledge. System speculation performance has been tested for different learning and test periods in the frame of a specific experiment design. As a result, it is statistically put forth that, this rule based system developed by the help of GP, could provide superior strategies compared to buy-and-wait strategy for testing periods of 6 months (120 trading days) and 12 months (240 trading days). In contrast, similar findings could not be reached for relatively shorter test periods like 1 week (5 trading days), 1 month (20 trading days) and 3 months (60 trading days). Experiments clearly show that; the system can provide profitable buy-sell strategies without "à priori" knowledge but only discovering rule combinations of some technical analysis indicators derived from daily historical prices.

Keywords: evolutionary computation, genetic programming, symbolic regression, rule-based systems, financial time series, technical analysis, efficient market hypothesis

..../..../2013

ETİK İLKE ve KURALLARA UYGUNLUK BEYANNAMESİ

Bu tez/proje çalışmasının bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumunda bilimsel etik ilke ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilmeyen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; bu çalışmanın Anadolu Üniversitesi tarafından kullanılan bilimsel intihal tespit programıyla tarandığını ve hiçbir şekilde intihal içermediğini beyan ederim.

Her hangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçlara razı olduğumu bildiririm.

Özgür İCAN

.....

Önsöz

Bilgisayarların problem çözmede kullanılmaya başlandığı ilk zamanlardan bu yana en çok merak edilenlerden biri, bilgisayarların ne zaman kendi kendilerini programlayabileceği olmuştur. Bilgisayarların kendi kendini programlayabilmesi, problem çözme sürecinin artık tamamen insan müdahalesi olmaksızın makinelerce yapılabileceği anlamına gelmektedir. Bu durum; modelleyicinin bilgisayara yalnızca problemi tanıtması, başka bir ifadeyle problemi modellemesi, çözüm adımlarını ise tamamen bilgisayara bırakması olarak da ifade edilebilir.

İnsanoğlu bilimsel ilerlemede henüz böyle bir aşamaya gelememiş olsa da, bu amaca yönelik girişimlere geçmişte rastlamak mümkündür. Bu girişimlerden belki de en büyüğü, Japonya Uluslararası Ticaret ve Sanayi Bakanlığı' nca 1982 yılında topyekün bir araştırma atağı olarak başlatılan V. Nesil Bilgisayar Sistemleri Projesi'dir. Çağının ilerisinde bir öngörüyle başlatılan bu proje, yapay zeka uygulamalarını etkin bir biçimde çalıştıracak ve çok sayıda işlemciyi kullanabilen paralel bilgisayar sistemleri ve yazılımları üretmeyi hedefleyen bir çabalar bütünü olarak anılmaktadır. Her ne kadar başarısı tartışılsa da, bu iyi niyetli çaba ve ABD'deki ve Avrupa'daki yapay zeka araştırmalarına ayrılan fonlar bu alanda birçok tekniğin gelişmesine sebep olmuştur. Genellikle matematiksel mantığa dayalı olarak mantık programlama paradigmasının egemen olduğu söz konusu araştırma dönemi maalesef büyük hayal kırıklıklarıyla sonuçlanmıştır.

Yaşanan olumsuzluklara rağmen edinilen tecrübeler göstermiştir ki; yapay zeka araştırmalarının yoğun olduğu yıllarda edinilen deneyimler yakın zamanda ortaya çıkan web teknolojilerinin ve hatta paralel donanım mimarilerinin çekirdeği olmuştur. Bu bağlamda, bir başarısızlıktan ziyade olgunlaşma süreci olarak değerlendirilebilecek bu süreç, birçok yapay zeka tekniğinin ortaya çıkmasına sebep olmuştur. Yapay zeka tekniklerinden bir grubu oluşturan evrimsel hesaplama teknikleri de ilk zamanlardan beri ivme kaybetme-

den gelişmeye devam etmiştir. Dünyanın en prestijli bilişim topluluklarından olan ACM[1] tarafından SIGEVO[2] adı altında ana bilişim ilgi alanlarından biri olarak tanınmıştır.

Uzman sistemlerde uzmanlık bilgisinin belli gösterimler yardımıyla makineye aktarılması mutlaka gerçek (insan) uzmana ihtiyaç duyan bir süreçtir. Öte yandan, uzmanlık bilgisinin makine tarafından otomatik olarak çıkarsanıp kural veya model tabanlarına aktarılabilmeleri için mutlaka bir öğrenme mekanizmasına ihtiyaç duyulmaktadır. Evrimsel hesaplama teknikleri bu öğrenme mekanizması işlevini yerine getirmektedir. Bu çalışmada başvurulan genetik programlama tekniği de diğer evrimsel algoritmalar gibi biyolojik evrimden esinlenen genetik öğrenmeyle bu boşluğu doldurmaktadır.

Finansal piyasalarda, söz konusu tekniklerin kullanımının çok eskilere dayanmadığı bilinmektedir. Bu bağlamda, çalışmanın Türkiye'de bu alandaki uygulama eksikliğini gidermeye yönelik bir çaba olduğunu belirtmek gerekir. Menkul kıymet piyasalarında geçmiş fiyatlardan hareketle gelecek dönem fiyatlarının tahmin edilebilmesi konusunda yapılan başkaca çalışmalarla izlenilen yaklaşım arasındaki temel farklılık, makinenin herhangi bir ön bilgiye sahip olmaksızın gerçek uzmanları ikame etmeye çalışmasıdır. Başka bir ifadeyle; bu çalışma, insan uzmanların bilgisine yalnızca ham geçmiş verilerden erişilip erişilemeyeceğini belirlemeye yönelik bilimsel bir çabadır. Söz konusu uzmanlık bilgisini ikame etme; bir veri yığınından matematiksel model çıkarsama olabileceği gibi; sistemdeki örüntüleri yakalamaya yönelik bir kural tabanı uygulaması da olabilir. Bu çalışmada her iki uygulama tipinin de nasıl gerçekleştirilebileceği detaylı bir biçimde açıklanmış ve yalnızca geçmiş fiyat bilgilerini kullanan kural tabanlı bir genetik programlama uygulaması aracılığıyla makinenin piyasaya üstün gelip gelemeyeceği araştırılmıştır. Böylelikle, Türk finansal piyasalarının etkinliği de sınanmış olmaktadır.

İstanbul Menkul Kıymetler Borsası (İMKB)'nın adı, 5 Nisan 2013 tarihinde yeni yapılandırma uyarınca Borsa İstanbul (BIST) olarak değiştirilmiştir. Ancak, çalışmanın tüm aşamaları bu tarihten önce tamamlandığından çalışma metniyle diğer kısımlar arasındaki adlandırma bütünlüğünün korunması amacıyla İMKB adı ve kısaltması korunmuştur.

[1] Association for Computer Machinery

[2] Special Interest Group for Genetic and Evolutionary Computation

İçindekiler

Sayfa

Tablo Listesi

Sayfa

Şekil Listesi

Sayfa

1. Giriş

Sermaye ve para piyasaları dünya üzerindeki sosyal sistemler arasında sürekli ve büyük hacimli veri üretmeleri bakımından özel bir öneme sahiptir. Öyle ki; veri üretme sıklığı bakımından fiziksel sistemlere belki de en yakın beşeri sistem menkul kıymet piyasalarıdır. Saatlik, dakikalık ve hatta daha kısa periyotlarda veri akışı yaratan finansal sistemlerin oluşturduğu zaman serileri olan finansal zaman serileri, sık frekanslı olmalarının yanı sıra gerçek zamanlı olarak özel bir ekonomik değere sahiptir. Bu ekonomik değer, finansal zaman serilerinin saklanmasını ve ilgili çıkar gruplarına aktarılması işini doğurmuştur. Büyük miktarlardaki finansal verinin saklanması ve iletilmesi de ancak veritabanı teknolojosi ve bilişim altyapısının ilerlemesi sayesinde mümkün olmuştur. Böylesine büyük hacimli verilerin uzunca bir dönem erişilebilir biçimde saklanabilmesi, finansal zaman serilerinde ampirik model geliştirmede araştırmacıların motivasyonunu daha da arttırmakta ve yeni araştırma olanakları sunmaktadır.

Finansal zaman serilerinin modellenmesinde son zamanlarda istatistiksel tekniklerin yanı sıra *otomatik öğrenme (machine learning)* tekniklerine de başvurulduğu gözlemlenmektedir. Daha çok *veri güdümlü (data driven)* olarak nitelendirilen bu tekniklerin kullanılmaya başlanmasındaki itici güç gelişen donanım ve yazılım teknolojileridir. Bu gelişime paralel olarak, uygulamacılar da daha talepkâr problemlere bir çözüm yaklaşımı olarak otomatik öğrenme tekniklerini kullanmaya başlamışlardır.

Sanayi devriminden günümüze kadar geçirilen süreçte uygulamalı disiplinler temel bilimlerle aralarındaki farkı giderek kapatacak bir zemin kazanmıştır. Bu süreçte insanlığın geçirdiği iki dünya savaşı ve bunların ardından gelen kutuplaşma (soğuk savaş) dönemi günümüz uygulamalı bilimlerini olduğu gibi, bunların vazgeçilmez aracı olan bilgisayar teknolojisini üst seviyelere taşımıştır. Transistörün icadıyla bilgisayarların işlem kapasitelerindeki artışı, bilgi saklamada manyetik disk ortamlarının ortaya çıkması takip etmiştir. Bunlara eklenen bilgisayar ağı teknolojisindeki gelişmeler, teorik olarak ele alınan ama

pratikte baş edilemez gibi görünen problemlerin matematiksel ve bilgisayar modelleriyle incelenebilmesine olanak sağlamıştır. Günümüzde; işlemci teknolojisinde işlemci çekirdeklerindeki *bütünleşik paralel mimariler (multicore)*; ağ teknolojilerinde ise *bilgisayar kümeleri (clusters)* ve *dağıtık ağ hesaplaması (distributed- grid computing)* gibi paralel hesaplama yaklaşımları uygulamacılara yeni olanaklar sunmaya devam etmektedir.

Hennesy ve Patterson, bilgisayar mimarisindeki sözü edilen gelişmeleri dört ana başlık altında toplamışlardır (Hennessy ve Patterson, 2006:14).

> Dramatik bir hızda ilerleyen dört (bilgisayar mimarisi) uygulama teknolojisi, modern bilgisayar mimarisi uygulamaları için kritik öneme sahiptir:
>
> 1. *Entegre devre mantığı teknolojisi:* Transistör yoğunluğu yılda 35 % dolaylarında artmaktadır ki; bu 4 yıldan biraz fazla bir sürede 4 katına çıkması demektir. Entegre devre *(die)* boyutlarındaki artış biraz daha az tahmin edilebilir olmakla beraber her sene 10-20 % dolaylarındadır. Bileşik etki; bir çip üzerindeki transistör sayısının, yaklaşık olarak senelik 40-55 % oranında artışıdır.
> 2. *Semikondüktör DRAM (dynamic random access memory):* Kapasite her yıl kabaca 40 % oranında artmaktadır ki; böylece 2 senede ikiye katlamaktadır.
> 3. *Manyetik disk teknolojisi:* Öncesinde yoğunluk yıllık 30 % oranında artmaktayken bu oran 1990 sonrasında 60% 'lara çıkmış; 1996 'dan 2004 'e kadar ise 100 % oranında gerçekleşmiştir. Devamında 2004 sonrasında yeniden 30 % 'lara düşen bu oranın izlediği inişli çıkışlı gelişim profiline rağmen sabit diskler, belleklere (DRAM) nazaran bit başına 50 - 100 kez daha ucuzdur.
> 4. *Ağ teknolojisi:* Hem ağ bağlayıcılarının *(switch)* hem de iletim sisteminin performansına bağlı olan ağ performansı da bu teknolojilere paralel olarak ilerlemektedir.

Hesaplama teknolojilerindeki bütün bu gelişmeler; daha önceden uygulamacıların dikkatini çekmemiş veya pratik olmayacağı düşüncesiyle geriye atılmış, büyük ölçekli problem durumlarını bilgisayar ortamında modellenip çözülebilir hale gelmektedir.

Gerçek hayattaki uygulamalarda karşılaşılan birçok mühendislik ve işletmecilik probleminin analitik yöntemlerle modellenip çözülmesi, bilgisayar teknolojilerindeki söz konusu gelişmelere rağmen zorluğunu korumaktadır. Mutlak ve kesin sonucu bulmanın analitik olarak imkansız olduğu, nümerik hesaplama yöntemleriyle ise çok fazla hesaplama

kaynağına ihtiyaç duyulan problemlerde, uygulamacılar yaklaşık çözümleri hızlı bir biçimde elde edebilme imkanını sunan *meta-sezgisel (metaheuristics)* çözüm yöntemlerini çokça kullanır olmuşlardır. Meta-sezgisel yöntemlerin ne olduğuna değinilmeden önce *sezgisel yöntemleri (heuristics)* tanımlamakta fayda bulunmaktadır.

Sezgisel yöntemler, *alan bilgisinin (domain spesific knowledge)* problem durumlarında kullanılmasıyla hızlı ve yaklaşık çözümler bulmaya çalışan yöntemlerdir. Bunlar, genellikle alan uzmanının *pratik kurallarından (rule of thumb)* genelleştirilmiş algoritmalardır.

Meta-sezgisel terimi ise, genellikle çeşitli stokastik optimizasyon yöntemlerine verilen genel bir isimdir. Bazı kaynaklarca bu teknikler üzerine yapışmış, pek de açıklayıcı olmayan, hatta yanlış anlamalara yol açabilecek bir terim olarak da nitelendirilmektedir (Luke, 2009:7). Yunanca'dan İngilizce'ye geçmiş *"meta"* kelimesi bir ön ek olup , önüne geldiği kelimeye tamlama anlamı katmakta; yalnız başına ise "sonrası, ötesi, ileri hali" anlamlarına gelmektedir [1]. Başka bir muhtemel anlamı da önüne geldiği kelimenin yinelenmesiyle oluşan isim tamlamasıdır. Bu durumda "metaheuristics" terimi "sezgisellerin sezgiseli" anlamına gelmektedir ki; bu ifadenin meta-sezgiselleri iyi tanımladığı düşünülmemektedir.

Meta-sezgisel teknikler genel olarak tanımlanırsa; karmaşık arama uzaylarında hızlı ve yaklaşık global çözümler verebilme özelliğine sahip stokastik optimizasyon teknikleridir. Genel olarak meta-sezgisel tekniklerde doğal bir olgunun metafor olarak kullanımına rastlanmaktadır. Örneğin *tavlama benzetimi (simulated annealing)* eriyik halinde bulunan katı madde atomlarının kararlı hale gelirken geçirdiği süreçten esinlenilerek geliştirilmiştir. Benzer biçimde *karınca kolonisi (ant colony)* tekniğiyle popülerlik kazanmış olan *sürü zekası (swarm intelligence)* teknikleri genel olarak karıncalar ve arılar gibi "sosyal böceklerin" doğada verdikleri ölüm kalım savaşında, birbirleriyle uyumlu bir biçimde hareket ederek elde ettikleri güçlü davranış kalıplarından esinlenen tekniklerdir. Bu teknikler, söz konusu metaforun doğal özelliklerinin birebir olmasa da kaba bir karşılığını bünyelerinde barındırır. Örneğin; *sürü zekası* tekniklerinin doğal olarak paralelleştirmeye açık olmaları, doğada sosyal böcek kolonilerinde çok sayıda birey arasında iş bölüşümü ve iletişi-

[1] `http://www.etymonline.com/index.php?search=meta&searchmode=none`
(Erişim tarihi: 08.12.2009)

min esas olmasıyla ilişkilendirilebilmektedir. Benzer şekilde *tavlama benzetimi* algoritmasının erken sonlandırılmasının kötü sonuçlar vermesi; çabuk soğuyan metal eriyiklerin atom dizilimlerinin rastgele olması gibi doğal fenomendeki bir olgunun izdüşümüdür.

Bahse konu olan metaforlar arasında belki de en güçlüsü *evrimdir*. Canlılar doğada *doğal seçilim* yoluyla yüz milyonlarca yıl içinde kazandıkları özelliklerle aşırı dinamik ortamlara uyum sağlayabilmişler ve bu yolla nesillerini sürdürebilmişlerdir. Birçok canlı, evrim yoluyla karmaşık hayatta kalma mekanizmaları geliştirmiştir. Bu canlılar bu sayede değişen doğa koşullarına uyum sağlayarak nesillerini sürdürmeyi başarmıştır. Doğadaki hayatta kalma savaşında başarılı olan canlıların hayatta kalmasını, zayıf olanların ise elenmesini sağlayan *doğal seçilim* mekanizmasıdır. Böylelikle; hayatta kalanların başarılı olmalarında etkili olan özelliklerin genetik miras olarak gelecek kuşaklara bırakılması kuşakların ortalama başarılarını daha iyiye götürecektir. Bu adaptasyon süreci evrimin bir optimizasyon mekanizması gibi taklit edilebileceği fikirlerinin önünü açmıştır. Biyolojik evrimin işleyişinden esinlenen bütün meta-sezgiseller, *evrimsel hesaplama* adı altında incelenmektedir.

1.1. Evrimsel Hesaplama ve Genetik Programlama

Evrimsel hesaplama, doğadaki biyolojik evrimden esinlenen bir çok optimizasyon algoritmasının altında sınıflandırıldığı şemsiye bir kavramdır. Bu kavramının ortaya çıkışı incelendiğinde, Alman araştırmacı Ingo Rechenberg'in 1971 tarihli doktora tezi olan, *"Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution"*, adlı Almanca eserinde ortaya attığı ve biyolojik evrimin ilk kez bilgisayar ortamında taklit edilmesiyle, karmaşık teknik problemlerin çözülmesini sağlayan *evrimsel stratejiler* sayesinde olmuştur. Rechenberg ve arkadaşları bu ilk uygulamaları hava araçlarının aerodinamik tasarımı için kullanmıştır. Rechenberg ile hemen aynı dönemlerde, John Holland tamamen bağımsız olarak yaptığı çalışmaları, 1975 yılında yayınladığı *"Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence"* adlı eserinde toplamış, konunun bilim dünyasında bilinirliğinin artmasında büyük rol oynamıştır. Holland bu eserde, evrimin birçok problemin çözümü için sanal ortamda gerçekleştirilebileceğini, konunun matematiksel temellerini ortaya koyarak teorileştirmiştir. Holland teorik çalışmalarını, *şema teorisi (schemata the-*

ory) altında toplamıştır . Holland'ın bu çabaları daha sonradan evrimsel hesaplamanın en popüler tekniği haline gelecek olan *genetik algoritmaların* (GA) da mucidi olarak anılmasına yol açmıştır (Holland, 1992).

Genetik algoritmaların, mühendislik uygulamalarında asıl popülerlik kazanması ise Goldberg'in 1989 yılında yayınladığı *"Genetic Algorithms in Search, Optimization and Machine Learning"* kitabından sonra olmuştur. Goldberg *temel yapıtaşı hipotezi (building block hypothesis)* ile genetik algoritmaların nasıl iyi çözümler yakalayabildiklerini ortaya koymuştur (Goldberg, 1989).

Genetik Programlama (GP) ise ilk olarak GA uygulamalarında kullanılan sabit uzunluktaki ikili veya sembolik kromozom kodlamalarının yerine, LISP[2] programlama dilindeki sembolik ifadelere benzer "program ağaçları" kullanılabileceği düşüncesiyle, John Koza tarafından gündeme getirilmiştir. Koza'nın *"Genetic Programming: On the Programming of Computers by Means of Natural Selection"* kitabı konu hakkında yazılmış ilk kitaptır. Bu kitapta Koza, çok değişik alanlardan mühendislik problemlerinin hepsinin, aslında belirli prosedürlerden oluşan bilgisayar programları olarak görülebileceğini[3] ve bu yüzden ortak bir yöntemle çözülebileceğini önermiştir.

1.1.1. Genetik programlama kaynaklarına genel bakış

Günümüzde konunun değişik alanlardaki uygulamalarıyla ilgili geniş bir kaynakça oluşmasına rağmen, daha yakın zamana kadar lisans seviyesinde bir ders kitabı yayınlanmamıştır. Konuya olan yoğun ilginin devam etmesi ve birçok alt dala ayrılması sayesinde, konuya yeni başlayanlar için giriş düzeyinde kaynak sıkıntısı giderilmiştir. Kronolojik olarak bakıldığında bu kaynaklar (Koza, 1992), (Banzhaf; Nordin; Keller ve Francone, 1997), (Langdon ve Poli, 2002) ve (Poli; Langdon ve McPhee, 2008) olarak sıralanabilir.

Son yıllarda GP'nın daha spesifik alt dallarına değinen kitaplar da yayınlanmaya başlamıştır. Örneğin GP program ağaçlarının değerlendirilmelerindeki zorluklar göz önüne

[2] Açılımı "**LIS**t **P**rocessing" olan LISP programlama dili tek bir dilden çok bir programlama dilleri ailesinin adıdır. FORTRAN ' dan sonra dünyanın en eski ikinci yüksek seviyeli programlama dili olan LISP, John McCarthy tarafından 1958 yılında geliştirilmiştir. Günümüzde yaşayan lehçeleri arasında Common Lisp, Scheme ve Emacs Lisp sayılabilir.

[3] Koza bu önerisine "program indirgeme" adını vermiştir.

alınarak geliştirilmiş "doğrusal" gösterimler, "doğrusal GP" başlığı altında toplanmıştır[4]. Bu konuda, (Brameier ve Banzhaf, 2006) gibi genel kaynaklara rastlamak mümkündür.

1.2. İktisadi ve Finansal Uygulamalarda Yapay Zeka Tekniklerinin Yeri

Geleneksel iktisadi görüşün temel özelliği bireylerin "rasyonel" olarak karar verdikleri varsayımıdır. Bu görüşün rasyonelllik tanımı içerisinde, bireylerin vardıkları kararlara nasıl eriştiklerini veya hangi usavurum (reasoning) süreçlerinden geçerek bu seçimleri yaptıklarını açıklayan herhangi bir bileşen yoktur. Geleneksel görüş sadece seçenekler arasında eğer bir özelliği en iyileyen bir seçim varsa, bu seçeneğin seçildiğini var saymaktadır. Bu rasyonellik modeli "kapalı kutu" veya bir başka ifadeyle usavurumsuz bir rasyonellik olarak nitelendirilebilmektedir. Bu durum, geleneksel iktisadi görüşün, birey ve organizasyonların karar vermelerinin ardında yer alan karmaşık ve özgün usavurma süreçlerini görmezden gelerek analiz için son derece basitleştirilmiş modeller kullanabilmesini sağlamaktadır. Bu sadeleştirmenin ekonomik teorinin gelişiminde büyük rol oynadığı, hatta gündelik hayatta pratik birçok izdüşümü olduğu kabul edilse de; tam da bu basitleştirme yüzünden iktisadi fenomenlerin doğasında yer alan karmaşıklık ve bu karmaşıklığı doğuran belki de çok basit olarak nitelendirilebilecek kurallar analiz perspektifinin dışında bırakılmaktadır (Dixon, 2001).

Karar teorisinin temellerini atmış olan, Von Neumann gibi öncüllerce de kabul edilen, mikro iktisadın çekirdeğindeki insan arketipinin tam bilgi altında rasyonel kararlar alabilen bireyler olduğu varsayımıdır. Hatta karar teorisinin temeli sayılabilecek fayda teorisinde bu durum fayda fonksiyonlarıyla somutlaşmaktadır (Von Neumann ve Morgenstern, 1945). İzleyen yıllarda ortaya çıkan oyun teorisi de aynı varsayımlar üzerine inşa edilmiş ancak farklı olarak çatışma ortamında bireylerin ve organizasyonların karar verme süreçlerini incelemektedir.

Klasik iktisadın temel varsayımlarından olan "homo economicus"; bir başka deyişle iktisadi kararlar verirken rasyonel davrandığı varsayılan ve bireysel faydasını en büyüklemeye çalışan insan arketipi, iktisadi analizin temeline oturtulmuştur. Bu terimi ortaya

[4]Doğrusal genetik programlamanın doğrusal programlamayla herhangi bir ilişkisi yoktur. İlki GP gösterimlerinin ağaç yerine, hiyerarşik olmayan doğrusal dizilimlerde gösterilmesini öneren bir GP yaklaşımıyken, ikincisi doğrusal en iyileme tekniklerine verilen genel isimdir.

atan ünlü 19. yüzyıl iktisatçısı John Stuart Mill olsa da (Persky, 1995), bu düşünce Mill'in öncülleri olan 18. yüzyıl iktisatçıları Adam Smith ve David Ricardo'nun fikirlerinde rasyonel ve yalnızca kendi çıkarlarıyla ilgilenen bireyler olarak tarif edilmektedir (Smith, 1863). Klasik görüş, 19. yüzyıl sonlarında matematiksel modeller kullanan bir grup iktisatçı tarafından daha da ileriye götürülmüştür. İktisatta matematiksel modellemeyi savunan ve kendi çalışmalarında kullanan bu iktisatçılar arasında Francis Edgeworth, William Stanley Jevons, Vilfredo Pareto ve Léon Walras gibi ünlü iktisatçılar sayılmaktadır. Nihayet Lionel Charles Robbins 20. yüzyılda 'Rasyonel Kararlar Teorisi' adı altında bu kavramı teorileştirmiş ve kendinden sonra gelen neo-klasik iktisatçılara da bu yönde bir etkide bulunmuştur.

"Rasyonel insan" görüşü ana akım iktisatçılar tarafından benimsense de birçok iktisatçı ve düşünür bu varsayıma karşı çıkmıştır.Bu varsayımın temelinde yatan fayda maksimizasyonu ve dolayısıyla bireysel fayda fonksiyonu kavramları eleştirilmiştir. Bunların başında Thorstein Veblen, John Maynard Keynes ve Herbert Simon gibi çok önemli iktisatçılar gelmektedir. Avusturya ekolü olarak anılan iktisatçılar da bu görüşe karşı çıkmışlardır. Karşıt görüşte olan bu düşünürlerin kimilerince değişik argümanlar öne sürülse de "homo economicus" arketipinin temel sorunu her türlü iktisadi öngörüyü başarıyla yapabildiği ve bu öngörüleri gerçekleştirirken her türlü bilgiye eksiksiz bir biçimde erişebildiği varsayımıdır. Bir başka ifadeyle, gerçek hayattaki belirsizlikten soyutlanmış ve gerçekçi olmayan varsayımlarla güçlendirilmiş bir arketiptir. Buna göre, bu insan arketipinin dayandığı temel varsayım olan "kusursuz rasyonellik" yerine "sınırlı rasyonellik" esastır. Bir başka ifadeyle, klasik iktisadın temel dayanağı olan rasyonel karar vericiler yerine kısmen rasyonel kararlar alabilen karar vericilerin tercihleri iktisadi fenomenlerin ortaya çıkmasında belirleyicidir. Bu akımı başlatan düşünürler arasında Herbert Simon'un ayrı bir yeri vardır. Öyle ki; Simon, yapay zekanın öncülerinden sayılırken aynı zamanda konunun iktisat ve ilgili alanlarda uygulanabilirliğini öncü çalışmalarıyla ortaya koymuştur (S.-H. Chen, 2005).

Herbert Simon ve ardıllarınca savunulan görüşe göre; iktisadi araştırmada araştırmacılara kolay gelen ancak tamamen gerçekçi olmayan varsayımlar yerine, karar vericilerin gerçekte nasıl karar verdikleri incelenerek işe başlanmalıdır. Bu bağlamda birçok bilim dalı ve disiplin, iktisadi modellere ortak bir zemin oluşturmalıdır. Bu bilim dallarının ba-

şında Simon'un kendisinin birçok çalışma yaptığı bilişsel psikoloji, bilgisayar bilimleri ve karar bilimleri gelmektedir. Simon'un öncülü olduğu bu görüş, iktisatta *sınırlı rasyonellik (bounded rationality)* ve *tatmin edici davranış (satisfying behaviour)* kavramlarının ortaya atılıp genellikle yapay zeka tekniklerine başvuran bilgisayar modelleriyle deneyselleştirilmesine imkan sağlamıştır (Andresen, 2001).

1.2.1. **Yapay zekanın kısa bir tarihçesi**

Yapay zeka (*Artificial Intelligence - AI*) çalışmalarının başlangıcı 1943 yılına kadar götürülmektedir. Bu tarihte Warren McCulloch ve Walter Pitts tarafından yapılan çalışmayla yapay sinir ağları *Artificial Neural Networks - ANN* disiplininin temelleri atılmıştır. Donald Hebb, 1949 yılında yaptığı çalışmayla McCulloch ve Pitts'in önerdiği mimarideki ANN'nın uygun bir biçimde tanımlandıklarında "öğrenme" yeteneklerine kavuşabileceğini öngörüsünü desteklemiştir. Hebb, süreç içerisinde halen etkili bir öğrenme modeli olarak gösterilen *Hebb modelini* geliştirmiştir (Russell ve Norvig, 2003:16).

Birbirinden bağımsız araştırmacılar sayesinde yürütülen çalışmalar ilk defa Alan Turing tarafından günümüzdeki AI sınıflandırmasına benzer bir şekilde gruplandırılmıştır. Otomatik öğrenme, evrimsel hesaplama ve pekiştirmeli öğrenme gibi konular ilk defa Turing tarafından yapay zeka vizyonu içerisinde ele alınmıştır. Dahası Turing'in geliştirdiği test, bir makinanın veya bilgisayar programının yapay zekaya sahip olup olmadığının belirlenmesinde temel araç olarak kabul edilmiştir (Turing, 1950).

İzleyen yıllarda, Princeton üniversitesinde Marvin Minsky ve arkadaşlarına John McCarthy'nin de katılmasıyla yapay zekanın ayrı bir disiplin olarak kabul edilmesini sağlamaya yönelik çalışmalar süreci başlamıştır. Öyle ki; 1956'nın yazında Dartmouth College'da organize ettikleri 2 ay süren çalıştayda günümüzde AI'nın kurucuları olarak anılan MIT, CMU ve IBM gibi farklı üniversite ve kurumlardan gelen araştırmacıları bir araya getirmeyi başarmışlardır. Simon ve Newell tarafından bu çalıştayda sunulan bildiride, geliştirdikleri bir bilgisayar programı olan *Logic Theorist*'in, *Principia Mathematica*'dan sembolik mantık ispatları yapabildiği gösterilmiştir. Ancak; Darthmouth çalıştayından çıkan netice sansasyonel bir buluştan çok, izleyen 20 yıl boyunca bu disiplini geliştirecek akademisyenlerin tanışması ve bu alanın günümüze kadar değişmeden gelen ismi olan "yapay zeka (artificial intelligence)" ismini alması olmuştur.

Değişik araştırmacıların farklı AI tanımları bulunmaktadır. AI'nın isim babası ve aynı zamanda AI araştırmalarının uzun yıllar gerçekleştirildiği temel dil olan LISP (McCarthy, 1965) programlama dilinin mimarı John McCarthy'dir. Ona göre yapay zeka; "zeki makineler, özellikle de zeki bilgisayar programları inşa etme bilimi ve mühendisliğidir" [5]. Genel kabul görmüş bir başka tanıma göre ise; yapay zeka, insanlar tarafından tasarlanmış ve inşa edilmiş makine veya bilgisayar programları gibi varlıkların sergiledikleri zekice sayılabilecek davranışları ortaya çıkaran zeka biçimi ve bunu inceleyen bilim dalıdır (Russell ve Norvig, 2003:55). Daha modern sayılabilecek bir tanıma göre yapay zeka disiplini, çevresini algılama ve bu sayede başarı şansını arttıran kararlar verebilme yeteneğine sahip zeki ajanlar tasarlama işidir (Poole, 1998:1-3).

Günümüzde AI araştırmaları son derece bağımsız alt alanlara ayrılmış olup neredeyse her bir alt alan ayrı bir bilimdalı olma niteliğine bürünmüş ve aralarındaki etkileşim sınırlı hale gelmiştir (McCorduck, 2004:424). Ancak, gerek tarihi ve kültürel bağlar nedeniyle bu alt dallar genellikle AI çatısı altında sıralanmaktadır. Yapay zeka araştırmalarında başta optimizasyon gibi uygulamalı matematiğin alt dalları olmak üzere çok çeşitli araçlar kullanılagelmiştir. Bu araçlara verilecek örnekler çoğaltılacak olursa; matematiksel mantık, olasılık teorisi, kontrol teorisi, teorik bilgisayar bilimleri, algoritma analizi ve programlama dilleri gibi değişik matematiksel ve matematiksel olmayan bilimsel disiplinleri de bu listeye eklemek gerekecektir. Bu sayılan teknikler ve daha başkaları, temelde *yapay zeka uygulamalarında* karşılaşılan *problemleri* çözmede benimsenen yaklaşımlar dahilinde kullanılmaktadır. Bu yaklaşımlar *sibernetik*, *sembolik*, *yarı-sembolik*, *istatistiksel* ve *zeki ajan paradigması* üzerine kurulu olan yeni nesil yaklaşımlar olarak kabaca sınıflandırılabilmektedir. Bu yaklaşımların hemen hepsinin çözmeye çalıştığı problemler usavurma (reasoning), bilgi gösterimi (knowledge representation), otonom planlama ve çizelgeleme (autonomous planning and scheduling), öğrenme (learning), doğal dil işleme (natural language processing), robotik, algılama (perception), sosyal zeka (social intelligence) ve genel zeka (general -strong- AI) altında gruplanmaktadır (Russell ve Norvig, 2003:27-28). AI, 1980'li yılların ardından yaşadığı gerileme döneminden güçlenerek geri çıkmış; hatta

[5] McCarthy'nin tanımı için bkz.:

`http://www-formal.stanford.edu/jmc/whatisai/node1.html`

(Erişim tarihi:25.01.2012)

2000'li yıllardaki sunucu tabanlı internet uygulamaları patlamasıyla beraber, bu alanda karşılaşılan birçok problemin çözümünde uygulamacıların ilk başvurdukları tekniklerden bazılarını barındırır hale gelmiştir.

1.2.2. **Yapay zekanın iktisadi uygulamalarda kullanım alanları**

Klasik AI olarak nitelendirilen konuların başında gelen *sembol işleme - hesaplama (symbolic processesing - computation)* yaklaşımının kurucuları arasında belki de en iz bırakmış bilim insanı Herbert Simon'dur. Simon'u diğer öncü AI araştırmacılarından ayıran en önemli özellik; bilişsel psikoloji, karar bilimleri ve iktisatta yaptığı çalışmalar sayesinde yapay zekanın iktisadi problemlerde kullanımına belirli bir zemin hazırlaması olmuştur (Frantz, 2003). Ancak, sembol işleme yaklaşımının kısıtlarının ortaya çıkmasıyla AI'nın iktisadi ve finansal konulardaki problemlerin çözümüne uygulanılabilirliği sonradan *"bilişimsel zeka" (computational intelligence) - CI* olarak anılacak olan bir grup teknik özelinde popüler hale gelmiştir. Bu tekniklerin ayırıcı özelliği matematiksel olarak ifade edilmesi daha güç ve karmaşık problemlere çözüm aramaları ve dolayısıyla matematiksel modellerden daha çok bilgisayar modellerine dayanan teknikler olmalarıdır. Ayrıca, bu tekniklerin büyük bir kısmının doğal ve biyolojik fenomenlerden esinlendiği bilinmektedir. Bulanık sistemler, yapay sinir ağları, evrimsel hesaplama, karınca kolonisi algoritmaları ve destek-vektör makinaları (Support Vector Machines - SVM) bu teknikler arasında sıralanmaktadır (Isasi; Quintana; Sáez ve Mochón, 2007).

Ekonomiler, heterojen karar vericilerin (ekonomik ajanların) etkileşim içerisinde bir araya geldikleri son derece dinamik ve karmaşık sistemlerdir. Bu heterojen karar vericilerin; hem kusursuz bilgiye erişim şanslarının olmaması hem de bu bilginin karar vericiler arasında adil olarak paylaşılmaması göz önüne alındığında bunların rasyonel davranabilmelerinin mümkün olmadığı ortadadır. Bu bağlamda; finansal krizler gibi birçok iktisadi fenomenin ne zaman ne şekilde ortaya çıkacağı, şiddetinin ne ölçüde olacağı ve ne zaman sona ereceği gibi sorular son derece ilginç ve cevaplaması bir o kadar da zor sorular haline gelmektedir. Hal böyleyken analitik çözümler elde etme amacı güden basitleştirilmiş klasik iktisadi modeller yerine birey tabanlı modelleme ve simülasyon (Agent Based Modelling and Simulation - ABMS) popülerlik kazanmaya başlamıştır. Bu tarz modellerde bireylerin şu veya bu biçimde bir öğrenme mekanizmasıyla donatılması gerekmektedir.

Bilişimsel zeka teknikleri tam da bu noktada ekonomik ajanların karar verme mekanizmalarını taklit etmede iyi birer alternatif olarak devreye girmektedirler. Bu karar verme mekanizmasını taklit etmede sıklıkla yararlanılan CI tekniklerinin başında evrimsel hesaplama algoritmaları gelmektedir. Evrimsel hesaplama algoritmaları, bireylerin edindiği bilgileri öğrenme ve aktarma süreçlerini, doğadaki evrim mekanizmasının canlıların kalıtsal özellik bilgilerini genç nesillere aktardığı genetik süreçleri taklit etmekle gerçekleştirmektedir. Bilişimsel zeka yöntemlerinin ve dolayısıyla evrimsel hesaplamanın iktisadi ve finansal uygulamalarını başlıklara ayırmak gerekirse; Chen ve Kuo tarafından 2002 yılında yayınlanmış olan bir derlemede konu üzerinde 1986 yılından 2001 yılına kadar yapılmış 380'den fazla yayını inceleyerek sınıflandırmada kullandıkları alt başlıklardan bazıları öne çıkmaktadır (S. Chen, 2002):

1.2.2.1. ***Finansal ekonomi***

Finansal ekonomi CI tekniklerinin belki de en yaygın uygulandığı ikitsadi uygulama alanıdır. Bu durumun sebebi örüntü tanıma problemlerinde yıllardan beri kendini ispatlamış olan bu tekniklerin, sermaye piyasalarında bu yeteneklerinin etkin bir biçimde kullanılabilmesi durumunda bilimsel ve ticari anlamda yüksek potansiyele sahip olmalarıdır. Gelişen bilgi teknolojisi ve yüksek frekanslı piyasa verilerine erişim imkanı bu tekniklerin kullanımını daha da cazip hale getirmektedir. Ayrıca, birçok kısıtlayıcı varsayımdan bağımsız bir biçimde veri güdümlü ve bilgisayar yazılımları inşa etmeye yardımcı olduklarından bu tekniklere olan ilgi giderek artmaktadır.

Yüksek frekanslı piyasa verisiyle gerçekleştirilen analizler (Bolland ve Connor, 1997), karmaşık finansal enstrümanların fiyatlandırılması (Malliaris ve Salchenberger, 1993) ve piyasa davranışlarının analizi (Palmer; Brian Arthur; Holland; LeBaron ve Tayler, 1994; Lettau, 1997) gibi uygulamalar öne çıkmaktadır. Bu alanda yapılan çalışmaların iki yöne doğru yöneldiği düşünülmektedir (Isasi vd., 2007:3). Bunlardan birincisi daha çok temelde *"birey (etmen) tabanlı modelleme ve simülasyon" (ABMS)* kapsamında değerlendirilen modeller olup daha çok piyasa simülasyonları biçimindedir. Bu modellere örnek verilecek olursa sanal hisse senedi piyasası (LeBaron, 2002) ve finansal kırılganlık modelleri sayılabilir (Gatti vd., 2005).

Diğer yön ise finansal tahminleme, al-sat stratejisi geliştirme, portföy yönetimi ve fi-

nansal varlıkların değerlemesi gibi uygulama alanları olup bu konudaki çalışmalarla ilgili detaylı bilgi ileride verilecektir..

1.2.2.2. ***Mikroekonomi, oyunlar teorisi ve endüstriyel organizasyon***

Daha çok mikroekonomi, yönetim ekonomisi, çalışma ekonomisi ve endüstri ilişkileri alanında klasik mikro analiz dışında alternatif teknik arayışlarına cevap ararken yapılmış çalışmalar bulunmaktadır. Bu sınıfa giren çalışmalarda da CI tekniklerinin sağladığı kolaylıklar, klasik iktisat modellerindeki özellikle rasyonellikle ilgili varsayımlardan kurtulma olanağı tanıması olarak özetlenebilir. Çalışmalar arasında; pazarlık (Andreoni ve Miller, 1991) ve açık arttırmaya dayalı fiyat tahmini modelleri (Saroop ve Bagchi, 2002; Bagchi ve Saroop, 2003) özgün olarak nitelendirilebilmektedir. Benzer şekilde oyunlar teorisiyle ilgili; (Andreoni ve Miller, 1993) ve (Harrald ve Fogel, 1996) gibi *"yinelemeli mahkumların açmazı" (repeated prisoners' dilemma)* oyununa ilişkin çalışmalar ile (Duffy ve Feltovich, 1999) gibi *"yinelemeli ültimatom" (repeated ultimatum)* oyunlarına ilişkin çalışmalar göze çarpmaktadır.

Bu alanda ayrıca mikroiktisadi analizde önemli bir yer tutan piyasa yapısına ilişkin bilgisayar modelleri içeren çalışmalar da bulunmaktadır. Örneğin; Chen ve Ni (S. Chen ve Ni, 2000) oligopol piyasalardaki rekabetin niteliği konusunda genetik algoritmalar yardımıyla bir benzetim modeli oluşturmuşlardır. Piyasa oyuncuları arasında rekabet dışında beliren diğer etkileşim türleri kooperasyon, koordinasyon ve koalisyon gibi ortak hareket etme biçimleridir. Bunlar da evrimsel hesaplama yaklaşımı yardımıyla bilişimsel modellere eklenmektedir (Vriend, 1995).

1.2.2.3. ***Makroekonomi***

Makroekonomi alanında, birey tabanlı modellere (ABM) ilişkin bir literatür son yıllarda giderek artan bir biçimde karşımıza çıkmaktadır. Bu çalışmaların temel motivasyonu, ekonominin adaptif karmaşık bir sistem olduğu görüşüdür. Buradan yola çıkılarak ABM simülasyon çalışmalarıyla "ortaya çıkma - belirme" fenomenleri yakalanmaya çalışılmaktadır. Buna örnek olarak gösterilebilecek bir çalışmada, Marimon ve arkadaşları paranın olmadığı değiş-tokuş ekonomisinde bir değişim aracı olarak paranın nasıl ortaya çıktığını göstermişlerdir (Marimon; McGrattan ve Sargent, 1990).

Makroekonomi alanında yapılan diğer çalışmalar arasında "örümcek ağı teorisi" (Arifovic, 1994) ve "çakışan kuşaklar teorisi" (Duffy ve Feltovich, 1999) ile ilgili modeller, CI tekniklerine başvurmada öncü olmaları sebebiyle önemlidir. CI tekniklerinin makroekonomide kullanımına ilişkin kapsamlı bir literatür çalışması da yine Arifovic tarafından gerçekleştirilmiştir (Arifovic, 2000). Ayrıca, Arifovic ve Maschek daha yakın zamanda örümcek ağı modellerinde evrimsel algoritmaların kullanımına ilişkin bir çalışma daha yayınlamıştır (Arifovic ve Maschek, 2006) . İlginç uygulama alanları arasında iktisadi büyüme (Silverberg ve Verspagen, 1994) ve geçiş ekonomileri de (S. Chen ve Yeh, 2000) sayılabilir.

1.2.2.4. ***Ekonometri***

CI tekniklerinin hem parametrik hem de parametrik olmayan ekonometride sıklıkla uygulandığı bilinmektedir (Isasi vd., 2007:3). Parametre tahmini (Dorsey ve Mayer, 1995), hipotez testi (S. Chen ve Yeh, 1996) veya model keşfetme (Fischer ve Leung, 1998) gibi alanlar bunlar arasında gösterilebilmektedir. Bu alanda yapay sinir ağlarının ve evrimsel hesaplama tekniklerinin bütünleşik olarak kullanıldığı çokça çalışmaya rastlamak mümkündür.

Bu kesime kadar CI tekniklerinin iktisadi uygulama alanlarına kısaca değinilmiştir. Bu noktada belirtilmesi gereken önemli bir husus da çalışmalardaki terminoloji karışıklığı ve bazı terimlerin sıklıkla birbiri yerine kullanılmasıdır. Bunun sonucunda teknikleri sınıflandırmada tam bir fikir birliğine varılamamaktadır. Sınıflandırmalarda karşılaşılan zorluğun bir diğer sebebi de bu tekniklerin sıklıkla birbirleriyle melez bir biçimde kullanılmasıdır. Böylece oluşturulan çözüm teknikleri farklı şekillerde adlandırmalarla kullanılabilmektedir.

1.2.3. **GP'nın ekonomi ve finans uygulamalarındaki özel yeri**

Evrimsel hesaplama adı altında yer alan bütün tekniklerin diğer bilişimsel zeka tekniklerinden farklı olarak finansal uygulamalar alanında ayrı bir yeri olduğu bilinmektedir. Evrimsel hesaplamanın ve dolayısıyla genetik programlamanın finansal zaman serileri gibi sık frekanslı zaman serilerinde uygulanmaya başlamasının öncesinde, bu alanda yapay sinir ağlarının ve değişik *otomatik öğrenme (machine learning)* algoritmalarının mevcut

istatistiksel tekniklerle beraber kullanımı söz konusuydu. Genel olarak, yapay zeka tekniklerinin bu alanda başarılı olabilecekleri düşüncesi, aslında bu tekniklerin hemen hepsinin *örüntü tanıma (pattern recognition)* uygulamalarında başarılı sonuçlar vermelerinden sonra ortaya çıkmıştır. Günümüzde ise; evrimsel algoritmalar bu alanda gerek diğer tekniklerle melez olarak, gerekse tek başına başarıyla kullanılmaktadır.

1.2.3.1. ***Sembolik regresyon***

Genetik programlama literatüründe, gözlem kümelerinden matematiksel model çıkarsamak için kullanılan sembolik regresyon modelleri başlı başına önemli bir yer tutmaktadır. İstatistiksel bir teknik olarak *regresyon*; çok değişik uygulama alanlarına ait ampirik modellerde başvurulan ortak bir araçtır. Regresyon modellerinde; eldeki gözlem değerleri modelleyici tarafından önceden belirlenmiş matematiksel bir forma uydurulmaya çalışılır. Bir diğer ifadeyle bu gözlem kümesini en iyi açıklayan katsayılar elde edilmeye çalışılır. Eğer model yeterince açıklayıcı değilse araştırmacı başka matematiksel formları denemeye devam eder. Bu yaklaşımın zayıf yönünü; yeterince açıklayıcı modeller kurmanın deneme - yanılma süreçlerine veya araştırmacının geçmiş deneyimlerine dayanması oluşturmaktadır. Bu durum; yalnızca doğurduğu iş yükü açısından değil modelleyicinin kişisel kabiliyet ve deneyimlerine fazlasıyla bağlı olduğundan bilimsel yönden de sakıncalıdır. Buna karşın *sembolik regresyon*, bu yaklaşımın tamamen tersi bir yol izlemekte; gözlem kümesine uydurulabilecek modele ilişkin hiçbir varsayımla yola çıkmamaktadır (Poli vd., 2008:114).

Sembolik regresyon ve doğru uydurma genetik programlamanın en erken uygulama alanlarından biri olup GP'nın mucidi Koza tarafından da üzerinde çok yoğun bir biçimde durulmuştur (Koza, 1992:237-288). Sembolik regresyon, GP'nın görece eski bir uygulama alanı olmasına rağmen bu konuda yapılan çalışmalar halen devam etmektedir. Özellikle çeşitli mühendislik alanlarındaki çalışmalar; optik (uzaktan algılama) (Tang; Michel ve Larouche, 2012), telekomünikasyon (sinyal kapsama alanı yönetimi) (Hemberg; Ho; O'Neill ve Claussen, 2011), makine mühendisliği (ısı iletimi) (Cai; Pacheco-Vega; Sen ve Yang, 2006) ve istatistiksel sistem modellemesi (Lew vd., 2006) gibi birbirinden çok farklı alanlarda yoğunlaşarak sürmektedir. Bu uygulama alanlarındaki başarılı sonuçlar, sembolik regresyonun istatistiksel doğru uydurma tekniklerine alternatif olarak kullanıldığını

ortaya koymaktadır. Sembolik regresyonu, daha etkin ve verimli hale getirmek amacıyla yürütülen çalışmalar; ölçeklendirme (Keijzer, 2004), bireylerin benzerliklerinin analizi ve genetik operasyonlara müdahale (Gustafson; Burke ve Krasnogor, 2005) ve çözümlerin karmaşıklık seviyelerinin belirlenmesi (Vladislavleva; Smits ve Den Hertog, 2009) gibi ayrıntılarda halen tekniğin teorik temellerine katkı sağlamaktadır. Teorik çalışmaların yanı sıra sembolik regresyon uygulamalarında genel GP kütüphanelerinden bağımsız olarak, araştırmacıların yararlanabileceği yazılımlar da ortaya çıkmıştır. Örneğin bir MATLAB *"araç kutusu" (toolbox)* olan GPTIPS (Searson; Leahy ve Willis, 2010) ve çok daha geniş kapsamlı bir çerçeve program olan HeuristicLab (Kronberger vd., 2012) bu yazılımlar arasında sayılabilir.

1.2.3.2. ***Kural tabanlı sistemler***

GP'nın finans alanındaki başarılı uygulamalarından sayılabilecek örnekler, kural tabanlı sistemler çatısı altında toplanabilmektedir. Bu sistemler, genellikle piyasaya giriş ve çıkış (bir başka ifadeyle al-sat) kararlarının bir dizi kural yardımıyla verilebileceği düşüncesi üzerine inşa edilmektedir. Bu kurallar bütünü, genellikle DOĞRU (TRUE) veya YANLIŞ (FALSE) sonucunu veren mantıksal (Boolean) ifadelerden oluşan ağaçlar (veya diğer veri yapıları) yardımıyla gösterilebilmektedir. Bu birleşik ifadelerin içinde yer alan yerel değişkenler, genellikle kararın alınması gerektiği andaki piyasa verilerinin veya bu verilerden elde edilen teknik analiz parametrelerinin yerini tutmaktadır. Böylelikle gerekli piyasa verileriyle beslenen kural ağaçları, verinin ait olduğu dönem için belli bir sinyal üretmektedir. Ayrıca, bu sistemler sinyaller yardımıyla elde ettikleri getirilerin piyasa getirilerine üstünlüğüne göre piyasanın etkinliği üzerinde fikir sahibi olmaya da yardımcı olmaktadır.

Bilindiği gibi, piyasa etkinliği finansal iktisadın temel kavramlarından biridir. Etkin piyasalarda menkul kıymetlerin piyasa katılımcıları tarafından olabilecek en iyi şekilde fiyatlandığını ileri sürer. Etkin Piyasa Hipotezine göre, etkin piyasalarda yalnızca geçmiş fiyat bilgisi yardımıyla piyasa getirilerinin üzerinde getiri elde etme şansının oldukça güç olduğu belirtilmektedir (Fama, 1970). Bu bağlamda; GP kullanılarak gerçekleştirilen kural tabanlı sistem çalışmaları, aynı zamanda piyasa verimliliğinin ampirik olarak sınandığı çalışmalar olarak ortaya çıkmaktadır.

Bu konuda yapılan çalışmalar arasında Bauer'inki öncü sayılabilir. Bauer, ikili dizilerle gösterilmiş al-sat stratejilerinden oluşan bir popülasyonu klasik GA kullanarak evriltmiştir (Bauer, 1994). Bu çalışmada al-sat kurallarından oluşan bireyler sabit uzunlukta kodlanmıştır. Makine tarafından elde edilebilecek çözümlerin değişken uzunluktaki kodlamalara nazaran yazarın benimsediği gösterime olan duyarlılığı daha yüksektir. Bu yüzden bu çalışmanın ardıllarınca kullanılan ağaç gösterimine sahip GP uygulamalarının al-sat stratejileri geliştirmek amacına daha uygun olduğu kabul edilmektedir. GP kullanarak al-sat stratejileri geliştiren sistemlerle ilgili ilk çalışma ABD St.Louis Federal Merkez Bankası araştırma departmanından Neely, Weller ve Dittmar tarafından gerçekleştirilmiştir (C. Neely; Weller ve Dittmar, 1997). Bu çalışmada ve ilerleyen yıllarda Neely ve Weller'in yaptığı diğer çalışmalarda, döviz piyasalarında GP yardımıyla gün içi işlemlerde piyasa üstü getiri elde etmenin mümkün olmadığı, bir başka ifadeyle döviz piyasasının verimli olduğu iddia edilmiştir (C. J. Neely ve Weller, 1999; C. J. Neely ve Weller, 2001). Benzer biçimde Allen ve Karjalainen de S&P 500 endeksinin 1928'den 1995'e kadar olan günlük verileri üzerinde al-sat kararları veren kural tabanlı bir GP sistemi geliştirmişler ve al-sat maliyetleri düşüldüğünde sistemin basit bir al-bekle stratejisinin üstünde tutarlı olarak ilave kazanç sağlayamadığını belirlemişlerdir (F. Allen ve Karjalainen, 1999). Marney ve arkadaşları ise, Neely ve Weller'in bulgularına karşı çıkmış ve 5 ABD hisse senedinin uzun vadeli geçmiş verileri üzerinde yaptıkları deneylerde GP'nin ekonomik olarak anlamlı üstün performanslar sergilediğini görmüşlerdir (JP Marney, 2001). Fyfe ve Marney devam niteliğinde olan çalışmalarında GP'nin teknik analiz yardımıyla üstün al-sat stratejileri geliştirebildiğini ancak risk düzeltmelerinden sonra bu üstünlüğün kaybolduğunu raporlamışlardır (Fyfe; Marney ve Tarbert, 2005). Neely ve Weller de Ulrich ile beraber yaptıkları çalışmayla yöneltilen eleştirileri doğrulayan sonuçlar elde etmişler bulgularını Lo'nun "Adaptif Piyasalar Teorisi" (Lo, 2004) ile bağdaştırmışlardır (C. J. Neely; Weller ve Ulrich, 2009). Burada vurgulanması gereken en önemli nokta şudur ki; al-sat kuralları Bauer'in öncü çalışmasında klasik GA ile *kısmen*; ardılı çalışmalarda ise bir çeşit GP sayesinde makine yardımıyla *tamamen* özgün olarak (bir ön tanımlama olmadan) ortaya çıkarılmaktadır.

Literatürde GP kullanılarak geliştirilen kural tabanlı sistemler kapsamında çok sayıda pragmatik uygulamalara ve yayınlara rastlamak mümkündür. Bunlardan Kaboudan, hisse

senedi getirilerinin daha güvenilir tahmin modelleri geliştirilebilmesi için bir ölçek geliştirmiştir. Bu ölçeğin kullanımıyla model arama uzayının önemli ölçüde küçültülebildiği raporlanmıştır (Kaboudan, 1999). Kaboudan, hisse senedi fiyatlarının tahminlemesinde önce bu ölçeği kullanarak GP'nın başarılı olacağı hisse senetlerini belirlemiştir. Ardından bir günlük fiyat tahmini yapan regresyon modellerini sanal evrime tabi tutmuş, 6 hisse senedi için 50 gün boyunca yalnızca bir gün sonrası için fiyat tahmini yapan bu modellerin üstün performans gösterdiğini belirlemiştir (Kaboudan, 2000). Ham petrol fiyatları için de benzer bir çalışma gerçekleştiren yazar (Kaboudan, 2001), döviz çapraz kurlarının tahminlemesine yönelik çalışmasında GP ile birlikte ANN'nı kullanmıştır. Bu son çalışma 3 çapraz kur verisine ilişkin ham ve *"dalgacık dönüşümü" (wavelet-transform)* uygulanmış seriler üzerinde gerçekleştirilmiştir. Böylelikle, GP'nın geliştirilmeye açık üstünlükleri raporlanmıştır (Kaboudan, 2005) .

Essex Üniversitesi'nden Edward Tsang ve arkadaşları tarafından geliştirilen EDDIE (Evolutionary Dynamic Data Investment Evaluator) adlı GP tabanlı sistem, bu alanda gerçekleştirilen çalışmalar içinde oldukça iddialı sonuçlar ortaya koymuştur. Başlangıçta, İngiltere'deki engelli at yarışlarına ait 180 yarışlık gerçek veri üzerinde deneysel olarak kullanılan EDDIE'nin, at yarışı bahislerinde kullanılan diğer genel kabul görmüş stratejilerden daha iyi sonuçlar verdiği gözlemlenmiştir (E. P. K. Tsang; Li ve Butler, 1998). Bu cesaretlendirici sonuçların ardından Tsang ve arkadaşları önce S&P-500'e ait 1963-1970 arasındaki 1800 işlem gününe ait gerçek veriyle EDDIE'yi eğitmişlerdir. Daha sonra, 1970-1974 arasında eğitim verisini izleyen 900 günlük bir periyotta gerçekleştirdikleri testlerle alternatiflere göre yüksek yıllık kazançlar elde edilebileceğini raporlamışlardır (Li ve Tsang, 1999; E. Tsang ve Martinez-Jaramillo, 2004).

Bir başka çalışma grubu olarak Cambridge Üniversitesi'ndeki Finansal Araştırmalar Merkezi (Center for Financial Research - CFR) ve yine Cambridge'de yer alan Judge Yönetim Enstitüsü (Judge Institute of Management) ve HSBC'den döviz piyasası uzmanlarının oluşturduğu ekip dikkat çekicidir. Bu grubun vadeli döviz piyasası işlemlerinde gerçek zamanlı ve adaptif sistemler üzerine yaptıkları çalışmalar öne çıkmaktadır. Dempster ve CFR'deki diğerlerince gerçekleştirilen erken dönem çalışmaları teknik analiz göstergelerini kullanan güniçi alsat kuralları üreten GP tabanlı sistemlere üzerine odaklanmaktadır (M. A. H. Dempster ve Jones, 2001; M. Dempster ve Jones, 2002). Daha sonraki çalış-

malarda bu çalışma gruplarının ilgisi piyasa oyuncularına ait kamuya açık olmayan *nakit akışı ve emir bilgilerinin* al-sat sistemlerine dahil edilmesine kaymıştır. Böylelikle, bu konuda FX piyasaların en büyük piyasa yapıcılarından HSBC'ye ait müşteri bilgileriyle yapılan çalışmalarla sistemlerin çok başarılı kazanç oranları yakalayabildiklerini ortaya konmuştur (Austin; Bates; Dempster; Leemans ve Williams, 2004; M. Dempster ve Leemans, 2006).

1.3. Çalışmada İzlenilecek Strateji

Evrimsel hesaplama tekniklerine dayalı finansal modeller incelendiğinde, bunların genel olarak üç sınıfta toplandıkları görülmektedir (Santini ve Tettamanzi, 2001). Bunlar;

1. bir zaman serisine ait veri türetici yapay sinir ağı mimarisini evrilterek optimize eden modeller,
2. sembolik ifadelerden oluşan basit bir bilgisayar program kümesini evrilterek belli bir veri türetici matematiksel ifade elde etmeye çalışan modeller,
3. zaman serisiyle ilgili kurallar bütününü içeren özel bir veri yapısı (örneğin kural ağacı) kümesini evrilten modeller,

olarak ifade edilmektedir.

Bu çalışmada; öncelikle yukarıdaki sınıflardan ikinci ve üçüncüsü bağlamındaki uygulamalara ilişkin araştırmalar ve deneme çalışmaları yapılmıştır. Ardından üçüncü sınıfta yer alan kapsamlı bir deney tasarımı ve uygulamaları dizisi geliştirilmiştir. Böylelikle; finansal zaman serileri gibi yüksek frekanslı veri yığınlarından, anlamlı örüntüleri yakalayabilecek güçte bir *otomatik öğrenme* tekniği olan genetik programlama yardımıyla anlamlı al-sat kararları verebilecek kural ağaçları üreten bir sistem geliştirilmiştir.

1.3.1. **Çalışmanın yaklaşımı**

Uygulamada izlenen yaklaşım dört ana adımdan oluşmaktadır:

1. Yeni baştan bir GP kütüphanesi (yazılımı) geliştirilebilmesine yönelik bir fizibilite çalışması gerçekleştirilerek diğer araştırmacılar tarafından geliştirilmiş ve literatürde tanınan GP kütüphanelerinin uygunluğu incelenmiştir.

2. Tercih edilen GP kütüphanesini kullanarak, eldeki problemi modelleyip çözecek sürücü yazılımlar geliştirilmiştir.

3. Geliştirilen sistem, Türkiye finansal piyasalarından İMKB'ye ait günlük veriler üzerinde denenerek bu piyasalarda tahminleme yapabilmeye yönelik matematiksel model ve kurallar bütünü elde edilmiştir.

4. Değişik modellerden, elde edilen sonuçlar karşılaştırılarak benimsenen yaklaşımın üstünlükleri ve zayıflıkları ortaya konulmuştur.

Bu doğrultuda toplamda üç sistem geliştirilmiştir. Geliştirilen bu sistemlerden ilk ikisi sembolik regresyon üçüncüsü ise kural tabanlı niteliklidir. İlk iki sistemde basit terminal ve fonksiyon kümelerinden tahmin edici matematiksel modeller türetilmeye çalışılmıştır. Bunun olumlu ve olumsuz yönleri okuyucuya sunulmuştur. Üçüncü uygulama kapsamında ise; geliştirilen sistemin hiçbir ön bilgiye sahip olmaksızın piyasaya giriş ve çıkış zamanlamalarında doğru sinyaller verebilirliği araştırılmıştır. Ardından belirli bir deney tasarımı uyarınca sistemin geçerliliği ve doğruluğu değişik öğrenme ve test periyotları için sınanmıştır.

1.3.2. **Çalışmanın amacı ve önemi**

Genetik programlama (GP) tekniğinin zaman serilerinde kullanımı çok eskilere dayanmamaktadır. Bu nedenle; tahminleme performansının Türk finansal piyasaları üzerinde araştırılmasında kullanılabilecek GP sistemleri geliştirme ve ampirik deneyler yardımıyla yöntemin güçlü ve zayıf özelliklerini geliştirilen sistemler özelinde ortaya koyma çalışmanın amacını oluşturmaktadır.

Çalışma sayesinde, konuyla ilgili Türkiye'deki uygulama eksikliğinin azaltılmasına katkıda bulunulmuştur. GP gibi yapay zeka yöntemlerinin Türk finansal piyasalarının tahmin edilebilirliği konusundaki etkinliği ortaya konulmuştur. Bu yolla araştırmacılara ve uygulamacılara benzer teknikler ve yeni uygulama alanları için oldukça kapsamlı bir fikir zemini sağlanmıştır.

Çalışmanın, akademik çıktılarının yanısıra pratik faydalarının da ortaya çıkabileceği düşünülmektedir. Akademik çıktı, Türkiye'de *finansal piyasaların verimliliğinin* ampirik olarak sınanmasıdır.Pratik fayda ise, menkul kıymet spekülasyonunda ve risk yöneti-

minde uygulamacılara karar desteği sağlayabilecek bir sistemin elde edilmesi biçiminde özetlenebilir.

2. Genetik Programlama ve İlgili Yazılımların İncelenmesi

Genetik programlama (GP), birinci bölümde kısaca değinildiği gibi evrimsel algoritmaların (EA) sınıflandırılmasında genetik algoritmalar (GA) grubu altında incelenmektedir. Bu bölümde GP'nin temel işleyişinin açıklanabilmesi amacıyla önce GA'nın çalışma prensipleri ele alınacaktır. İzleyen alt bölümlerde ise salt GP'ye özgü niteliklere gerekli olduğu kadarıyla değinilecektir.

GA ve GP'nin çalışma prensipleri yeterince açıklandıktan sonra, GP yazılımlarından (kütüphanelerinden) öne çıkanlara ilişkin kısa bilgiler verilecek ve uygulamada tercih edilecek yazılım seçiminde dikkat edilen kriterler ele alınacaktır.

2.1. Genetik Algoritmaların ve Genetik Programlamanın Temelleri

GA ve GP, *bilgisayar bilimlerinde* ve *yapay zeka* disiplininde, kendilerini ve ilgili diğer teknikleri kapsayan çatı kavram olan *evrimsel algoritmalar* başlığı altında incelenmektedir. Optimizasyon tekniklerinin sınıflandırılması açısından ise *stokastik global optimizasyon* teknikleri arasında yer almaktadır. Ancak GA ve GP'nin tavlama benzetimi gibi diğer stokastik global optimizasyon algoritmalarından farkı arama uzayında paralel, bir başka ifadeyle çok sayıda nokta üzerinden arama yapabilmesidir.

Evrimsel algoritmaların tarihçesi kısaca incelendiğinde, 1950'li ve 1960'lı yıllardan başlayarak çok sayıda bağımsız araştırmacının mühendislik alanlarında karşılaşılan optimizasyon problemlerinde kullanılabilecek araç arayışında olduğu görülmektedir. Bu amaçla doğadaki evrim fenomeninin incelendiği ve bilgisayar modeli bağlamında *evrimsel sistemler* üzerinde çalışıldığı görülmektedir. Bu çabaların başında, Alman araştırmacılar Rechenberg ve Schwefel'in 1960'lı yılların ortalarından 1970'lerin ortalarına kadar değişik çalışmalarında ortaya koydukları *evrimsel stratejiler* gelmektedir. Bu teknik, hava araçlarının kanat tasarımında karşılaşılan gerçel sayılı parametrelere sahip optimizasyon

problemlerinin çözümüne yardımcı olması amacıyla geliştirilmiştir (Rechenberg, 1973; Schwefel, 1965). Aynı yıllarda Amerikalı bir grup araştırmacı; Fogel, Owens ve Walsh günümüzde *evrimsel programlama* olarak bilinen tekniği geliştirmişlerdir. Bu teknik, problem gösterimi olarak sabit bir *sonlu durum makinesi* üzerinde çalışmakta olup, genellikle mutasyon yardımıyla durum geçiş diyagramının değiştirilmesi esasına göre işlemektedir (Owens; Walsh ve Fogel, 1966). Anılan iki teknik de günümüze kadar kendi bilimsel literatürünü oluşturup GA'dan yalıtılmış bir biçimde ilerlemiştir (Michell, 1998:2).

GA, John Holland ve doktora öğrencileri tarafından 1960'lar ve 1970'lerde Michigan Üniversitesi'nde gerçekleştirilen çalışmalar sonucunda ortaya çıkmıştır. Holland ve ekibinin amacı yukarıda bahsi geçen EH tekniklerinden farklı olarak yalnızca bir optimizasyon tekniği elde etmek değildi. Bunun ötesinde doğadaki evrim fenomeninin teorik özelliklerini daha iyi çözümlemeye yardımcı olacak ve bilgisayar üzerinde bu özellikleri taklit edebilecek programlar geliştirmeye daha uygun olan bir yaklaşım ortaya koymaktı. Ancak şurası bir gerçektir ki; GA popülaritesini optimizasyon problemlerinde kullanımına borçludur.

2.1.1. **GA'nın biyolojik kökleri**

Bütün canlı organizmalar *hücre (cell)* denilen yapıtaşlarından oluşmuştur. Hücreler dokuları, dokular bir araya gelerek organları, organlar ise belli bir işlevi yüklenen (örneğin mide ve bağırsaklardan oluşan sindirim sistemi gibi) sistemleri oluşturur. Her hücrede canlının genetik materyalini taşıyan *kromozomlar (chromosome)* bulunur. Kromozomlar DNA sarmallarından ibaret olup organizmanın genetik modelini bünyelerinde barındırır. Öyle ki; bu genetik model sayesinde bir türe ait bireyler gelecek kuşaklara sahip oldukları fiziksel ve diğer özellikleri aktarmaktadır. Kromozom üzerinde yer alan belli bloklara *gen* adı verilir. Her gen belirli tip proteinlerin sentezlenmesini sağlayan ve dolayısıyla belirli *özelliklerin (trait)* ortaya çıkması için gerekli emirlerin kodlandığı bir program parçacığı gibidir. Bir genin kodladığı özellikler bir insanın saç rengi olabileceği gibi halen anlaşılamamış bir fonksiyonu kontrol eden bir özellik de olabilir. Bir özelliğin alabileceği olası değerlere *alel (allele)* denir. Saç rengi örneği için bu değerler `{Sarı, Siyah, Kahverengi}` gibi bir kümeye ait olabilir. Her genin kromozom üzerinde *lokus (locus)* olarak adlandırılan belirli bir yeri bulunur. Bir canlıya ait bütün genetik materyal *genom*

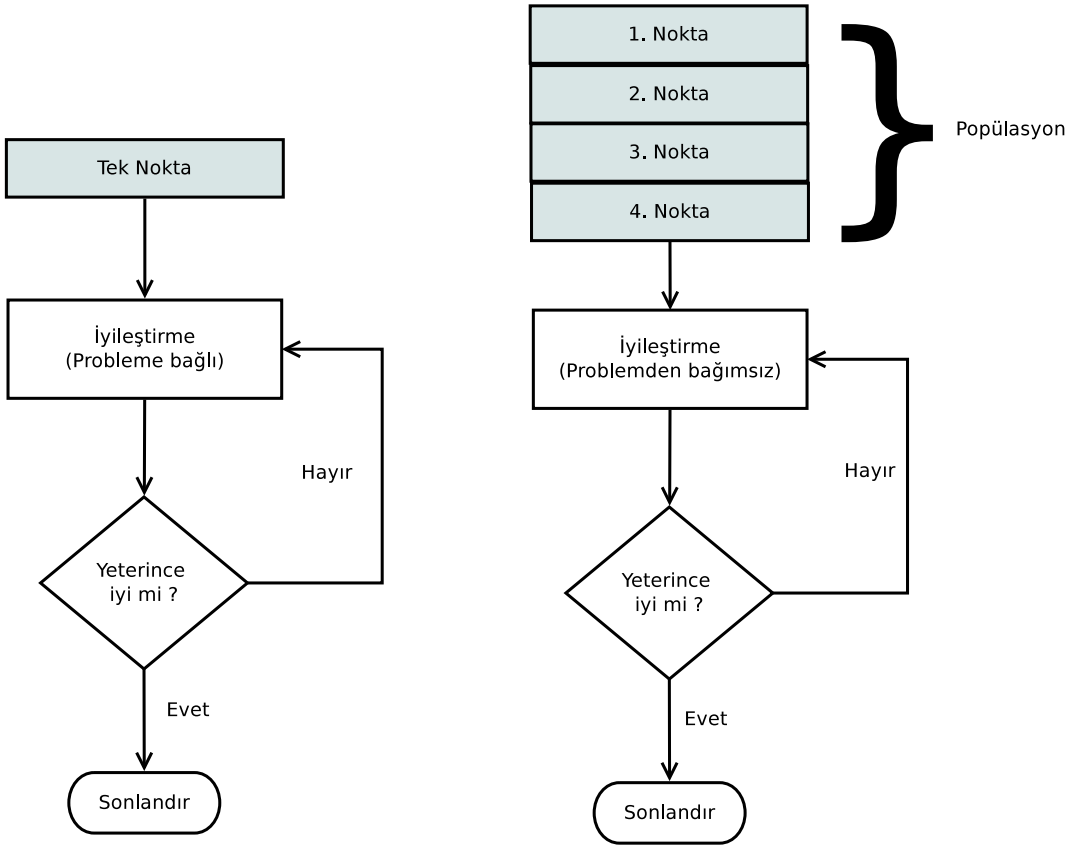
(genome) olarak adlandırılır. Genomda belli bir (işlevi yerine getirdiği düşünülen) gen kümesi *genotip (genotype)*, bunların canlıda ortaya çıkardığı özellikler bütünü de *fenotip (fenotype)* olarak adlandırılır. Bir canlının *uyumu (fitness)* da çevre şartları veriyken canlının hayatta kalma yeteneğinin bir ölçüsüdür (Sivanandam ve Deepa, 2007:16-23).

Biyolojik karşılığındakine benzer olarak GA'da kromozom, belli bir problem için aday çözümlerin (genellikle) bir dizi veya benzeri bir veri yapısı biçimindeki gösterimidir. Eldeki probleme göre kromozomun problemin çözümünde ne derecede iyi olduğunu belirleyen bir *uyum fonksiyonundan (fitness function)* söz edilir. Uyum fonksiyonu, genotipi fenotipe dönüştüren bir fonksiyondur. Dolayısıyla, GA'da uyum fonksiyonu değeri, söz konusu problemde ilgili kromozomun kodladığı çözümün iyiliğini ölçmeye yaramaktadır. Çözülmeye çalışılan probleme bağlı olarak uyum fonksiyonunun araştırmacı tarafından açıkça belirlenip GA sistemine entegre edilmesi gereklidir. Uyum fonksiyonunun tanımı bağımsız değişken olarak kromozom üzerinde belirli bir parça (genler) tarafından kodlanan karar değişkenlerinin varlığını gerektirir. Biyolojik karşılığına paralel olarak genlerin kodladığı değişkenlerin ait olduğu tanım kümesindeki alternatifler de alel olarak isimlendirilir. Genotip ve fenotip kavramlarının da sırasıyla kromozomdaki kodlanmış kısımla bunun uyum fonksiyonunda karşılık geldiği değer olduğunu bu açıklamalardan sonra tahmnin edilebilir hale gelmektedir. Dolayısıyla biyolojik kavramlarla GA terminolojisi arasında doğadan esinlenen diğer algoritmalara nazaran çok daha güçlü paralellikler bulunmaktadır.

2.1.2. **Genetik algoritmaların çalışma prensipleri**

GA paralel arama algoritmasıdır; bu anlamda diğer stokastik global optimizasyon algortimalarından ayrışır. Gerek deterministik (matematiksel) global ve lokal optimizasyon algoritmaları gerekse paralel olmayan global stokastik optimizasyon algortimaları (örn.: tavlama benzetimi) arama uzayını tek nokta üzerinden tarar. Geleneksel yöntemler olarak adlandırılabilecek bu deterministik optimizasyon algoritmaları genellikle en iyilenecek fonksiyonun gradyan bilgisine ihtiyaç duyarlar. Bu nedenle konveks olmayan arama uzaylarında yerel optimumlara takılma tehdidi altında kalır. Tavlama benzetimi benzeri stokastik global optimizasyon algoritmaları ise böylesi bir gradyan bilgisine ihtiyaç duymaz. Öte yandan bunlar, arama uzayının büyüklüğü karşısında popülasyon bazlı

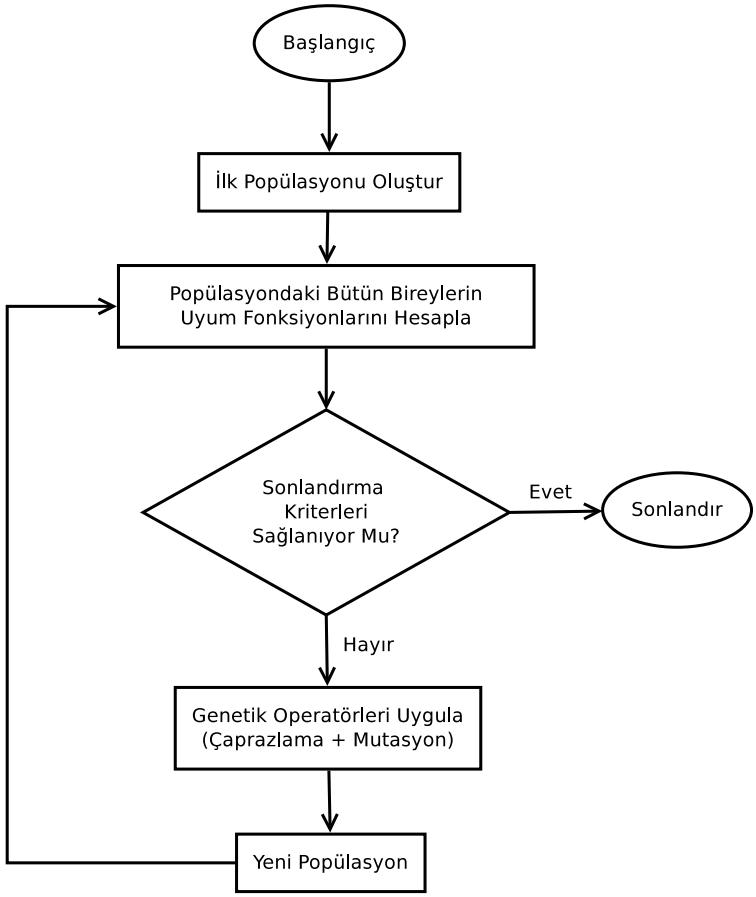
GA gibi arama tekniklerine göre güçsüz kalabilmektedir. Tek nokta üzerinden arama yapan geleneksel yöntemler mevcut çözümün iyileştirilmesinde probleme bağlı iyileştirmeler yaparken; GA gibi popülasyon tabanlı evrimsel algoritmalarda problemden bağımsız bir iyileştirme söz konusudur. Bu farklılık Şekil 2.1'de özetlenmiştir. Global stokastik optimizasyon algoritmalarının, geleneksel yöntemlere göre dezavantajını matematiksel olarak global optimumu bulduklarını garanti altına alamamaları oluşturur. Ancak, geleneksel algoritmalarla çözümün imkansız olduğu bilinen arama uzaylarında, çok kısa sürelerde *yaklaşık en iyi* çözümler sunmaları bakımından tartışılmaz öneme sahiptirler (Sivanandam ve Deepa, 2007:24-39).



Şekil 2.1: Geleneksel Optimizasyon Yöntemleriyle (solda) GA'nın (sağda) Kabaca Karşılaştırılması

GA'da ilk adım, önce rastgele bir popülasyon oluşturulmasıdır. Ardından gelen değerleme aşamasında popülasyonda bulunan bütün kromozomların uyum değerleri hesaplanır. Üçüncü adımda genetik operatörler olan çaprazlama ve mutasyon yardımıyla yavru kro-

mozomlar oluşturulur. İzleyen adımda, bir önceki adımda oluşan yavru kromozomlar ve ebeveyn kromozom havuzundan seçim operatörü yardımıyla seçilen kromozomlarla bir sonraki nesil oluşturulur. Yeni neslin uyum değerleri hesaplandıktan sonra sonlandırma kriteri sağlandığında algoritma sonlandırılır; aksi takdirde sonlandırma kriteri sağlanana kadar önceki işlemler tekrar edilir. GA'nın sonlandırılması, popülasyonun uyum değerinin ortalamasının belli bir değere yakınsaması ya da maksimum nesil sayısına ulaşılması gibi kriterlerce kontrol edilmektedir. GA'nın (ve aslında birçok EA için de geçerli olan) genel çalışma prensibi Şekil 2.2'de gösterilmiştir.



Şekil 2.2: Genel GA Akış Şeması

Klasik GA algoritması psödokod olarak Şekil 2.3' de gösterilmiştir. Burada P_t ve C_t ile sırasıyla t neslindeki ebeveyn ve yavru popülasyonlar ifade edilmektedir.

t ← 0;

P(t)'yi oluştur;

P(t)'deki kromozomların uyum değerlerini hesapla;

while *(sonlandırma kriteri == False)* **do**

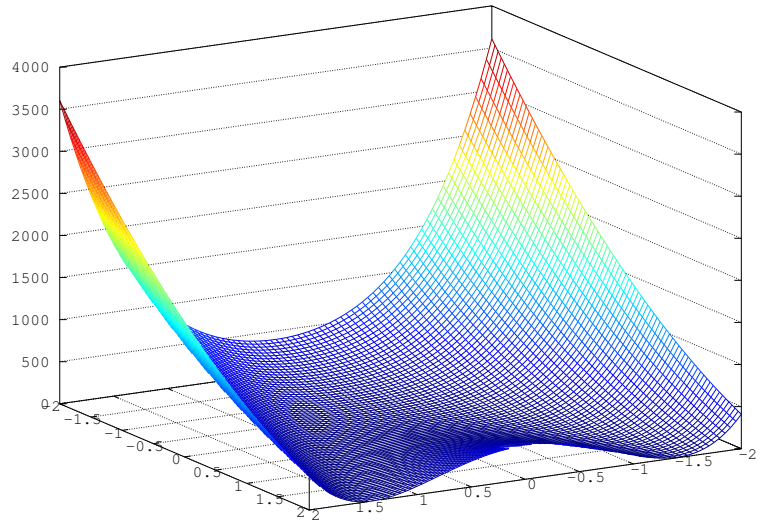
> P(t)'deki kromozomlardan C(t)'yi elde etmek için eşleştirmeye devam et;
>
> C(t)'deki kromozomların uyum değerlerini hesapla;
>
> P(t+1)'i, P(t) ve C(t)'deki kromozomlardan seçerek oluştur;
>
> t ← t+1;

end

Şekil 2.3: Klasik GA Prosedürü

2.1.2.1. ***Gösterim ve kodlama***

Gösterim (representation), GA'da kromozomların (aday çözümlerin) nasıl bir veri yapısı yardımıyla gösterileceği ve bu veri yapısını oluşturacak bileşenlerin nasıl bir alan üzerinde tanımlanacağı ile ilgilenir. GA'da kullanılan gösterim biçimleri genellikle *ikili (binary)*, *reel sayı (real)* ya da *sembol (symbol)* şeklinde özetlenebilir. *Kodlama (encoding)*, kromozomların oluşturulmasında, probleme ilişkin çözüme dahil edilen karar değişkenlerinin normalde ait oldukları tanım alanında aldıkları değerlerden ilgili gösterime çevrilmesi işlemlerinin bütünü olarak da nitelendirilebilmektedir.



Şekil 2.4: Rosenbrock Fonksiyonu

Gösterim ve kodlama, ikili gösterimin sürekli bir optimizasyon probleminde kullanımı ile örneklenebilir. Örneğin; matematiksel ifadesi Denklem 2.1'de, üç boyutlu grafiği Şekil 2.4'de verilen Rosenbrock fonksiyonu, kısıtsız sürekli optimizasyon algoritmalarının sınanmasında birçok test kümesinde (benchmark suit) kullanılır. Non-konveks sürekli fonksiyonlar tepe tırmanma algoritması gibi yerel arama tekniklerini yerel optimum noktalarında tuzağa düşüren tipik fonksiyonlardır. Rosenbrock fonksiyonu, $f^*(1,1) = 0$ noktasında global minimumu olan bir non-konveks fonksiyondur.

$$\min_{x \in \mathbb{R}^2} f(x,y) = (1-x)^2 + 100(y-x^2)^2 \tag{2.1}$$

Söz konusu fonksiyonun ikili kodlamasında gerekli olan bit uzunluğu, karar değişkenlerinin alabilecekleri değer aralıkları ve hesaplamanın hassasiyeti ile ilişkilidir. Bu örnek kapsamında, virgülden sonra yalnızca $\varphi = 2$ basamaklık bir hassasiyetin yeterli olduğu düşünülmektedir. Bu durumda v_j kromozomunu oluşturmak için x_j ve y_j karar değişkenlerinin değer aralıklarının her ikisinin de $[a_j, b_j] = [-2, 2]$ olduğundan ilk yarının x değişkenine, diğer yarının da y değişkenine ayrılması gerekmektedir. Bir başka ifadeyle değişkenlerin ikisi de aynı sayıda bitle kodlanabilecektir. Her iki değişken için,

en az $(b_j - a_j)x10^2$ değeriyle ifade edilebilecek bir gösterim gereklidir. Bir değişken için gerekli bit miktarı m_j, Denklem 2.2'deki eşitsizlikte gösterilmiş olup, örnek fonksiyon için $m_j = 9$ olarak bulunur. Bu durumda Rosenbrock fonksiyonunu kodlamak için x ve y değişkenlerinden her biri için 9, toplamda 18 bitlik bir uzunluk gerekmektedir.

$$2^{m_j-1} < (b_j - a_j)x10^{\varphi} < 2^{m_j} - 1 \tag{2.2}$$

GA'da ikili gösterime sahip ancak gerçel değerler alan değişkenlerin onluk düzene yeniden çevrilmelerinde Denklem 2.3'te verilen formülden yararlanılmaktadır. Burada $Dec()$ fonksiyonu aldığı ikili bir dizi parçasını ondalık değere dönüştüren basit bir fonksiyonu, s_j ise x_j karar değişkeninin kromozomda kodlandığı parçayı ifade etmekedir.

$$x_j = a_j + Dec(s_j)x\frac{b_j - a_j}{2^{m_j} - 1} \tag{2.3}$$

Bu örnek bağlamında popülasyonun oluşturulmasında, sadelik amacıyla popülasyon büyüklüğü $n = 5$ olarak alınacaktır. Popülasyon büyüklüğü yalnızca GA için değil bütün popülasyon tabanlı paralel arama algoritmalarında çok önemli bir parametredir. Popülasyon büyüklüğü arttıkça arama uzayının daha geniş yelpazesini tarama fırsatı sunar. Ancak getirdiği ilave hesaplama yükü nedeniyle de algoritmanın performansını olumsuz yönde etkiler.

2.1.2.2. ***Değerleme ve seçilim***

Doğada canlıların hayatta kalma mücadelesinin temel mekanizması ve Darwin'in evrim teorisinin özü *doğal seçilim (natural selection)* olgusudur. Evrim teorisinde, doğal seçilim kavramı Darwin tarafından bilinçli olarak *yapay seçilim (artificial evolution)* kavramının karşıtı olarak kullanılmıştır. Öyle ki; tarih içinde insanoğlu evcilleştirdiği yaban hayvanları ve ıslah ettiği yabani bitkileri görece kısa bir zaman zarfı içerisinde yapay seçilimle (kendisine daha yararlı olacak özellikleri barındıran) başka türler haline getirmiştir. Doğada ise türlerin oluşum sürecini belirleyen yalnızca doğal koşullar olmuştur. Darwin'in ölümünden çok sonra ortaya çıkan genetik bilimi de Darwin'in teorisini doğrulamıştır. Böylelikle; doğal seçilimin, nesiller boyunca doğal koşullara daha iyi uyum sağlayan bireylerin genetik materyallerinin, popülasyonun gen havuzunda baskın hale gelmesiyle

gerçekleştiği ortaya çıkmıştır. Doğal koşullara daha iyi uyum sağlayan bireylerin bu görece üstünlüklerini sağlayan fenotipleri ortaya çıkaran genler, bu bireylerin daha fazla hayatta kalma ve dolayısıyla daha fazla üreme şansı elde etmeleri neticesinde popülasyon gen havuzunda birikmektedir. Böylelikle; her yeni nesilde bu özellik, giderek gözle görülür ölçüde baskın hale gelmektedir. GA'daki *seçilim (selection) operatörü* de doğal seçilim analojisini kullanmaktadır.

Değerleme (evaluation) aşamasında popülasyondaki bütün kromozomlar ve kromozomlarda kodlanan karar değişkenleri yeniden fenotiplerine dönüştürülüp, bu değerlerin uyum (amaç) fonksiyonundaki yerlerine konulmasıyla uyum fonksiyonunun bu noktada aldığı değer hesaplanır. Ardından, GA literatüründeki en klasik seçilim yaklaşımlarından biri olan *rulet çarkı (roulette wheel)* gibi seçim teknikleriyle yeni popülasyon oluşturulur. Rulet çarkı yöntemi bireylerin seçilme olasılıklarının doğrudan uyum fonksiyonu değerlerine dayalı olarak belirlendiği *uyum-oranlı (fitness-proportional)* seçilim yöntemleri grubuna girmektedir. Buna göre öncelikle her kromozomun uyum değeri hesaplanır:

$$f(x_k) = Eval(v_k),\ k = 1,2,\ldots,n \tag{2.4}$$

İkinci adımda popülasyonun toplam uyumu elde edilir:

$$F = \sum_{k=1}^{n} f(x_k),\ k = 1,2,\ldots,n \tag{2.5}$$

Üçüncü adımda her kromozomun seçilme olasılığı hesaplanır:

$$p_k = \frac{f(x_k)}{F},\ k = 1,2,\ldots,n \tag{2.6}$$

Son aşamada ise her kromozomun birikimli olasılığı hesaplanır:

$$q_k = \sum_{j=1}^{k} p_j,\ k = 1,2,\ldots,n \tag{2.7}$$

Rulet çarkı tekniğinde yukarıda adımlardan sonra $[0,1]$ aralığından çekilen bir rassal r sayısının çarkta nereye denk geldiğine q_k birikimli olasılık değerlerine bakarak karar verilir ve ilgili v_k kromozomu seçilir. Rulet çarkı prosedürünün klasik formu Şekil 2.5 'de verilmiştir. Ancak bu prosedürün uygulanabilmesi için problemin maksimizasyon problemi olması ve herhangi bir karar değişkeni kombinasyonu için hesaplanan uyum fonksi-

yonu değerinin sıfır veya negatif olmaması gerekir. Bu durumun üstesinden gelinmesi için ham uyum fonksiyonu değerinin *ölçeklendirilmesi (scaling)* yoluna gidilir. Uygulamada benzer problemlerde rulet çarkı tekniğinin yanı sıra literatürdeki diğer seçilim yöntemleri arasında yer alan *turnuva (tournament) yöntemi* de sıklıkla kullanılmaktadır.

k ← 1 ;
r ← Rand([0,1]) ;
v^* ← [] ;
while *(r > q_k)* **do**
    k ← k+1 ;
    v^* ← v_k
end
return v^* ;

Şekil 2.5: Klasik Rulet Çarkı Seçim Prosedürü

Bu noktada kısaca ölçeklendirme kavramına değinmekte yarar bulunmaktadır. GA gibi evrimsel algoritmalarda bazı iyi kromozomlar erken jenerasyonlarda popülasyonda baskın gelerek erken yakınsama problemine yol açabilmektedir. Hem algoritmanın erken yakınsayarak yerel optimumlarda takılı kalmasını, hem de gen çeşitliliğinin erken azalmasını engellemek amacıyla ölçeklendirme ve sıralama tekniklerinin ortaya çıkarılması gerekmiştir. Bu durum $f'_k = g(f_k)$ gibi ham uyum fonksiyonunu ölçeklendirilmiş uyum fonksiyonuna dönüştüren bir g fonksiyonuyla gösterilir. Literatürde *doğrusal ölçeklendirme (linear scaling), dinamik doğrusal ölçeklendirme (dynamic linear scaling), sigma budama (sigma truncation), üssel ölçeklendirme (power law scaling), logaritmik ölçeklendirme (logarithmic scaling), pencereleme (windowing), normalleştirme (normalizing), Boltzmann ölçeklendirmesi (Boltzmann selection) ve sıralama (ranking)* gibi birçok teknik yardımıyla ham uyum fonksiyonunun ölçeklendirilebileceği belirtilmiştir (Gen ve Cheng, 1999:25-29). Bu yaklaşımlardan normalleştirme, dinamik ölçeklendirme sınıfına ait olup maksimizasyon ve minimizasyon problemlerinin hepsinde kullanılabilmesi bakımından hem basit ve hem de tercih edilen bir yaklaşımdır. Minimizasyon problemi için;

$$g_k = \frac{f_{max} - f_k + \gamma}{f_{max} - f_{min} + \gamma} \tag{2.8}$$

olarak ifade edilen normalleştirme denkleminde f_k, f_{max} ve f_{min} sırasıyla, mevcut nesildeki popülasyonun k kromozomuna ait uyum fonksiyonu değeri ile popülasyondaki en büyük ve en küçük uyum fonksiyonu değerlerini ifade etmektedir. Burada γ sabiti genellikle $(0,1)$ aralığına ait küçük bir pozitif reel sayıdır. Böylelikle denklemin sıfıra bölme sorunuyla karşılaşması engellenmektedir. Rosenbrock fonksiyonu örneğine geri dönülürse, 1. nesildeki bütün bireylerin normalleştirilmiş uyum fonksiyonu değerlerini Tablo 2.1'de g_k sütununda verilmiştir. Bu değerlere dayalı olarak her kromozomun seçilme olasılığı sırasıyla;

$$p = [0.0004, 0.1437, 0.2945, 0.2619, 0.2995]$$

ve kümülatif olasılıkları ise sırasıyla;

$$q = [0.0004, 0.1441, 0.4386, 0.7005, 1]$$

olarak hesaplanır. Bilindiği gibi rulet çarkı yönteminde, çark popülasyon büyüklüğü kadar çevrilmektedir. Bir başka ifadeyle $[0,1]$ aralığından rastgele çekilen n adet pozitif reel sayı yardımıyla yeni nesilde bulunmaya aday olan bireyler belirlenmektedir. Rassal sayı dizisi aşağıda verilen biçimde oluşturulduğunda;

$$r = [0.9788, 0.3674, 0.1124, 0.6432, 0.5531]$$

yeni nesilde bulunacak kromozomlar sırasıyla v_5, v_3, v_2, v_4, v_4 olacaktır. Buradan; bir kromozomun birden fazla seçilebileceği görülmektedir.

Tablo 2.1: Rosenbrock Fonksiyonu İçin Örnek GA Uygulaması

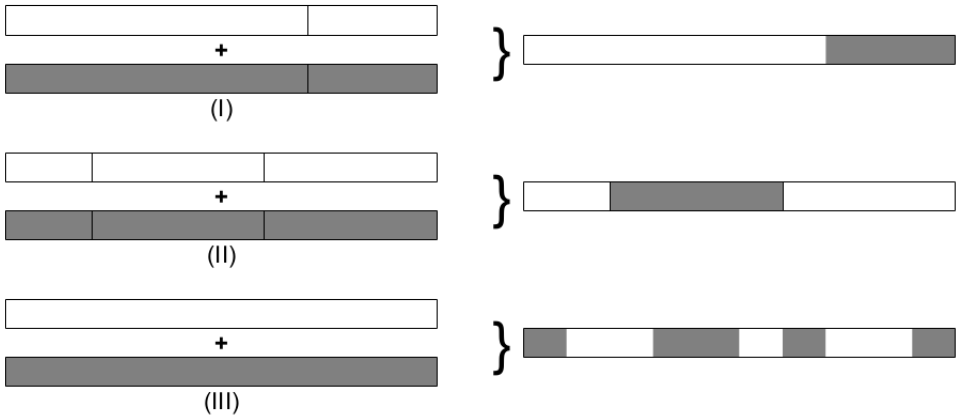
Kromozom	**İkili Gösterim**	**Gerçel Değer**	f_k	g_k
(A) Başlangıç Popülasyonu: 1. Nesil				
v_1	1111111111111111	(2.00, 2.00)	401.00	0.001
v_2	1001111100110011 0	(0.50, -1.20)	210.11	0.480
v_3	1011111110101100 0	(1.00, 0.70)	9.20	0.983
v_4	1011011000111111 1	(0.85, 0.00)	52.62	0.874
v_5	0101111110001100 1	(-0.50, 0.20)	2.57	1
(B) Çaprazlama ve Mutasyon Öncesi 2. Nesil				
v'_1	0101111110001100 1	(-0.50, 0.20)	2.57	1
v'_2	1011111110101100 0	(1.00, 0.70)	9.20	0.983
v'_3	1001111100110011 0	(0.50, -1.20)	210.11	0.480
v'_4	1011011000111111 1	(0.85, 0.00)	52.62	0.874
v'_5	1011011000111111 1	(0.85, 0.00)	52.62	0.874
(C) Çaprazlama İçin Seçilen Bireyler ve Oluşan Yavrular				
v'_2	1011111110101100 0	(1.00, 0.70)	9.20	-
v'_5	1011011000111111 1	(0.85, 0.00)	52.62	-
1. Yavru	1011111101111111 1	(1.00, 0.00)	100	-
2. Yavru	1011011001010110 0	(0.85, 0.70)	0.10	-
(D) Mutasyon İçin Seçilen Bireyin Öncesi ve Sonrası				
v'_4	1011011000111111 1	(0.85, 0.00)	52.62	-
v'_4	1010011000111111 1	(0.60, 0.00)	13.30	-
(E) Çaprazlama ve Mutasyondan Sonra 2. Neslin Son Hali				
v'_1	0101111110001100 1	(-0.50, 0.20)	2.57	0.988
v'_2	1011111101111111 1	(1.00, 0.00)	100	0.525
v'_3	1001111100110011 0	(0.50, -1.20)	210.11	0.002
v'_4	1010011000111111 1	(0.60, 0.00)	13.30	0.937
v'_5	1011011001010110 0	(0.85, 0.70)	0.10	1

2.1.2.3. ***Genetik operatörler***

Rekombinasyon veya daha çok bilinen adıyla *çaprazlama (crossover)* genetik operatörlerden en önemlisidir. Çaprazlama ile iki ebeveyn kromozomdan yeni bir yavru kromozom elde edilir. Böylelikle ebeveyn kromozomların uyumunu belirleyen genler yeni oluşturulan yavruda bir şekilde yığılacaktır. Seçilimle, görece daha iyi bireyler gelecek kuşaklara aktarılacağından, uyumu iyileştiren genler de genç kuşaklarda giderek artan bir biçimde yığılmaya başlayacaktır. Goldberg'in *temel yapıtaşı (building block) hipotezinin* özünde bu düşünce yatmaktadır.

Elde edilen yavru kromozom belirli bir olasılıkla *mutasyona* uğrayabilmektedir. Mutasyon; genetik biliminde canlının genetik materyalini kopyalarken çeşitli sebeplerden dolayı oluşabilen hatalı kopyalama işlemleri için kullanılan bir terimdir. Mutasyon, genellikle canlıda olumludan çok olumsuz değişimlere yol açmaktadır. Buna karşın bu değişimler, nadiren de olsa bireyde mevcut gen havuzunda daha önce rastlanmayan, ancak çevre şartlarına diğerlerinden daha iyi uyum sağlayabilmesine yol açan fenotip değişikliklerine yol açmaktadır. Bu durumda bu mutasyonla sağlanan geni taşıyan birey, bir çırpıda daha avantajlı konuma geçerek genlerini rakiplerine göre daha çok yayma fırsatı yakalayacak ve gen havuzunu ilerleyen kuşaklarda domine edebilecektir. Darwin'in evrim teorisinin belki de özü olan bu düşünce genetik algoritmaların da yerel optimumlardan kurtulmada ve arama uzayının daha avantajlı olan bölgelerini tarayabilmelerindeki temel mekanizmayı oluşturmaktadır.

Literatürde, üzerlerinde çalıştıkları gösterime göre farklılık gösterebilen çok sayıda çaprazlama ve mutasyon yaklaşımı bulunmaktadır. Çaprazlama özelinde; ikili gösterimde en çok kullanılan yaklaşımlar genellikle *tek noktadan çaprazlama (single point crossover)*, *iki noktadan çaprazlama (two point crossover)* ya da *uniform çaprazlamadır (uniform crossover)*. Bu yaklaşımlar Şekil 2.6'da sırasıyla (I), (II) ve (III) etiketleriyle gösterilmiştir. Şekilden kolayca anlaşılacağı gibi tek nokta çaprazlama, dizi üzerinde belirlenen rastgele bir bit'in pozisyonunun solunda kalanı birinci ebeveynden, sağında kalanı ise ikinci ebeveynden alınarak gerçekleştirilir. İki noktadan çaprazlamada rastgele iki pozisyon belirlenir, bu iki pozisyon arasında kalan kısım ikinci ebeveynden alınırken, başlangıçtan birinci pozisyona kadar olan kısım ve ikinci pozisyondan sona kadar olan kısım



Şekil 2.6: GA'da Bit Dizisi Şeklindeki Gösterimlerde Yaygın Kullanılan Çaprazlama Yaklaşımları

ilk ebeveynden alınır. Uniform çaprazlamada ise her genin hangi ebeveynden alınacağı uniform dağılımdan olasıklarla tamamen rastgele belirlenir.

Çaprazlama işleminin uygulanacağı ebeveyn kromozomlar çaprazlamaya seçilme olasılığı p_c uyarınca belirlenir. Diğer önemli parametreler gibi bu olasılık da sistemin başlangıcında sabit olarak belirlenebileceği gibi dinamik de olabilir. Örnek problem için $p_c = 0.40$ varsayılırsa normal şartlar altında popülasyondan $nxp_c = 5x0.40 = 2$ bireyin çaprazlama işlemine tabi tutulması beklenir. Ebeveynlerin çaprazlamaya nasıl seçileceği ise Şekil 2.7'de verilen prosedür yardımıyla belirlenir.

Veri: n,p_c
k $\leftarrow$ 1 ;
while $(k \leq n)$ **do**
  $r_k \leftarrow$ Rand([0,1]) ;
  if $(r_k < p_c)$ **then**
    v_k kromozomunu ebeveynlerden biri olarak belirle ;
  end
  k $\leftarrow$ k+1 ;
end

Şekil 2.7: Çaprazlamaya Katılacak Kromozomların Seçim Prosedürü

Örnekteki beş kromozom göz önüne alındığında, bunların her biri için

$$r = [0.8317, 0.3311, 0.4126, 0.5908, 0.0635]$$

biçiminde rassal sayılar çekildiğinde, çaprazlamaya katılacak olan kromozomlar v'_2 ve v'_5 olarak belirlenir. Basitlik amacıyla, tek nokta çaprazlamasıyla bu iki kromozom tam ortadan çaprazlanırsa elde edilecek yeni yavru kromozomlar Tablo 2.1 'in (C) kısmında verildiği gibi oluşur. Önceki kesimde çaprazlama teknikleri tek yavru elde etme bağlamında aktarılmıştı. Ancak, uygulamada sıklıkla bu tekniklerin uygulanmasıyla iki yavru elde edilebildiği görülmektedir. İkinci yavruyu elde etmek için, birinci yavruyu elde etmede izlenen adımlar ebeveynler yer değiştirilerek tekrar edilir. Örnekte tek nokta çaprazlamasından iki yavru elde edilmiş ve bunlar yeni oluşturulan popülasyonda ebeveynlerinin yerine geçmiştir.

İkili gösterime sahip GA sistemlerinde mutasyon, yaygın biçimde *bit değiştirme (bit flipping - shifting)* olarak gerçekleştirilir. Burada mutasyon için seçilen bit'in değeri tersine çevrilmesi[1] biçiminde bir yaklaşım uygulanmaktadır. Mutasyon genellikle çok küçük bir olasıkla gerçekleştiğinden GA sistemlerinde bu parametre çok küçük tutulma eğilimindedir. Bu olasılık herhangi bir kromozomun mutasyon için seçilmesi olasılığı olmayıp genomdaki tüm bitlerin içinden belirli bir bitin mutasyona uğrama olasılığını ifade etmektedir. Ele alınan örnek için, mutasyon olasılığının $p_m = 0.01$ olarak verildiği varsayımı altında 90 bitlik genomdan, $nxmxp_m = 5x18x0.01 = 0.9$ (yaklaşık olarak) yalnızca 1 bit'in mutasyona uğrayabileceği beklenir. Bit değiştirme işlemine ilişkin prosedür Şekil 2.8'de verilmiştir.

[1] Tersine çevirme 0 olan bit'i 1; 1 olan bit'i 0 haline getirme işlemine verilen isimdir.

Veri: n,m,p_m

$N \leftarrow n*m*p_m$;

i $\leftarrow$ 1 ;

while $(i \leq N)$ **do**

  $r_i \leftarrow$ Rand([0,1]) ;

  if $(r_i < p_m)$ **then**

    i 'nci biti tersine çevir ;

  end

  i $\leftarrow$ i+1 ;

end

Şekil 2.8: Mutasyona Uğrayacak Bitin Seçim Prosedürü

```
r=[0.73919,  0.58665,  0.46026,  0.15155,  0.80418,
   0.80770,  0.21909,  0.31079,  0.16977,  0.99568,
   0.05475,  0.11897,  0.58655,  0.50875,  0.67164,
   0.89551,  0.29050,  0.58374,  0.48819,  0.63571,
   0.35205,  0.59824,  0.75815,  0.29921,  0.06101,
   0.06265,  0.68040,  0.16370,  0.44439,  0.75676,
   0.50816,  0.15151,  0.50470,  0.11630,  0.80285,
   0.51865,  0.90191,  0.98443,  0.44915,  0.31961,
   0.90031,  0.64364,  0.06391,  0.14786,  0.22642,
   0.48315,  0.07546,  0.67687,  0.09913,  0.85865,
   0.06601,  0.33667,  0.39291,  0.45302,  0.86015,
   0.64924,  0.57920,  0.00824,  0.06927,  0.09155,
   0.30191,  0.22426,  0.67632,  0.57743,  0.14175,
   0.67835,  0.52834,  0.67837,  0.26878,  0.79115,
   0.62322,  0.58394,  0.58256,  0.28189,  0.97035,
   0.77822,  0.15563,  0.56860,  0.45716,  0.46590,
   0.51323,  0.31495,  0.80773,  0.16713,  0.38575,
   0.57883,  0.81246,  0.08400,  0.83616,  0.21827]
```

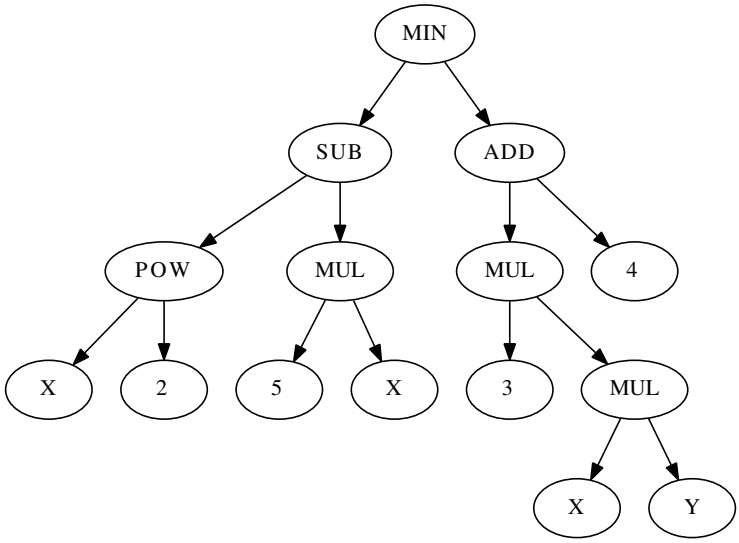
Örnekteki 90 bit büyüklüğündeki genom için yukarıdaki 90 rassal sayıdan oluşan r vektörü oluşturulmuştur. r vektörünün 58. elemanının değerinin (0.00824) $p_m = 0.01$'den küçük olduğu görülmektedir. Bu durumda v'_4 kromozomunun 4. bit'i tersine çevrilir. Bu işlem sayesinde 1 olan bit değeri 0 olarak değişirken karar değişkenlerinin değeri de (x, y) = (0.60, 0) olarak güncellenir. Mutasyon sonrası kromozomun değerlemesiyle ortaya çıkan uyum fonksiyonu değeri ise 13.30 olarak bulunur. Bu işlemlerin özeti Tablo 2.1'in (D) kısmında gösterilmiştir. Böylelikle ikinci nesil tamamen oluşturulmuş olup son hali aynı tablonun (E) kısmında verilmiştir.

2.1.3. **Genetik programlamanın genel prensipleri**

GA ile GP arasındaki temel fark, gösterim ile ilgili gibi görünmesine karşın aslında şekilsel olmaktan öte işlevseldir. Gerçekten de; GA'da kromozomlar birkaç değişik formda (ikili, gerçel veya karakter tabanlı) gösterilirken bunların hepsi sabit uzunlukta bir veri yapısıyla sınırlandırılmaktadır. Buna karşılık; GP'de her kromozom içinde fonksiyon ve verileri bir arada bulunduran bir *programdır*. GP gösterimi ağaç, dizi (doğrusal) veya yığın şeklinde olabilse de, kromozomların her biri bir *ayrışım ağacına (parse tree)* karşılık gelmektedir. GP temelde GA ile aynı mekanizmayı kullansa da kromozomların birer ayrışım ağacı olmaları klasik GA'nın yeteneklerinin dışında kalan problemlerde kullanılabilmesini sağlamaktadır. Bilgisayarların otomatik olarak problem çözmesi ve tamamen makine yardımıyla buluş yapma gibi alanlar çok daha büyük arama uzaylarını arama yeteneği gerektirmesi bakımından GA'dan ziyade GP'nin ilgi alanına girmektedir.

Daha önce de belirtildiği gibi GP'de kromozomlar (bireyler) programlardır. Burada program ile ifade edilmek istenen herhangi bir programlama dilinde yazılmış kaynak kodlardan çok *soyut sözdizimsel ağaçlardır (abstract syntax tree - AST)*. AST, bilgisayar bilimlerinde kaynak kodun ağaç şeklinde gösterimine verilen isimdir. GP'de programlar karmaşık işleri yerine getiren modüler tasarıma sahip büyük yazılımlardan ziyade daha küçük prosedür veya fonksiyonlardır.

Örneğin; $f(x,y) = min(x^2 - 5x, 3xy + 4)$ gibi bir fonksiyondan ele alındığında; bu fonksiyonda bulunan değişkenler ve sabitler `{X, Y, 2, 3, 4, 5}` olup bunların oluşturduğu kümeye *terminal kümesi* adı verilir. Ayrıca bu programda yer alan bileşen niteliğindeki temel fonksiyonlar `{MIN, POW, SUB, ADD, MUL}` şeklinde bir kümede topla-



Şekil 2.9: Örnek programın ağaç gösterimi

nabilir. Bu küme *fonksiyon kümesi* olarak adlandırılır. Fonksiyon kümesinde, gösterim kolaylığı açısından fonksiyonların matematiksel simgeleri yerine sembolik isimlerinden faydalanılabilmektedir. Terminal ve fonksiyon kümelerinden oluşan kümenin bütününe GP sisteminin *asli bileşen kümesi (primitive set)* adı verilir.

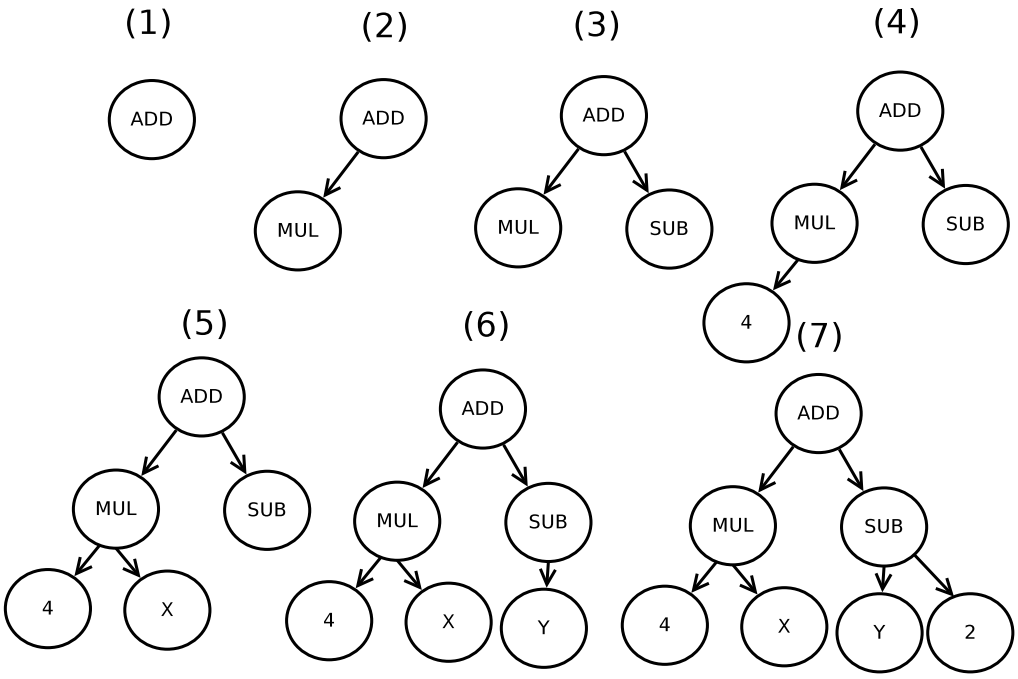
GP literatüründe programlar çoklukla ağaç olarak ifade edilmekte, bu yüzden yazılı gösterimde Lisp programlama dili ailesine mensup dillerdeki gibi *öntakı (prefix) notasyonu* tercih edilmektedir. Örnekte verilen programın öntakı notasyonuna çevrilmiş hali

```
(MIN (SUB (POW X 2) (MUL 5 X)) (ADD (MUL 3 (MUL X Y)) 4))
```

biçimindedir.

Fonksiyonların hangi sayıda argümana (girdiye) ihtiyaç duyduğuna *arite (arity)* denir. GP sistemleri tasarlanırken öncelikle fonksiyon kümesinde yer alan fonksiyonların ariteleri açıkça belirlenmelidir. Öyle ki; matematiksel arite ile GP sisteminin kodlandığı programlama dilinde kendine karşılık gelen fonksiyonun aritesi farklı olabilir. Örnek programın fonksiyon kümesinde yer alan fonksiyonların tümünün aritesinin 2 olduğunu belirtmekte fayda bulunmaktadır. Bu programın ağaç gösterimi Şekil 2.9'de verilmiştir.

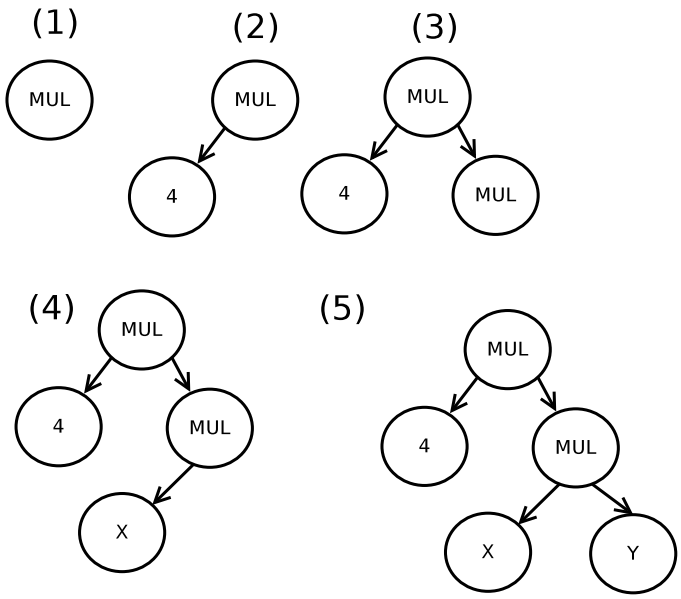
GP araştırmalarında geleneksel olarak kullanılan gösterim ağaçtır. Çok sayıda GP sistemi halen bu gösterimi kullanmaktadır. Ağaç tabanlı GP sistemlerinde ilk popülasyonun oluşturulmasında genellikle başvurulan birkaç geleneksel yöntem bulunmaktadır. Sıklıkla



Şekil 2.10: Temsili Bir Bireyin Doldurma Yöntemiyle Oluşturulması Aşamaları

kullanılan bu yöntemler arasında *doldurma (full)*, *büyüme (grow)* ve bu ikisinin bir bileşimi sayılan *eğimli yarı yarıya (ramped half-and-half)* hem en eski hem de en çok kullanılanlardır. Bunlardan ilki olan doldurma yöntemi, bir ağacın bütün derinliği dolana kadar fonksiyon kümesinden çekilen rassal bileşenlerle doldurulması; yalnızca yapraklarda terminal kümesine başvurulması ilkesine göre çalışır. Ağaç derinliği ağacın kökünden yapraklarına kadar gidildiğinde geçilmesi gereken düğüm sayısıdır. Maksimum ağaç derinliği GP sisteminin tasarımında baştan belirlenmesi gereken bir parametredir. Doldurma yöntemiyle oluşturulan bir popülasyonda bütün ağaçların derinliği birbirine eşit olup ağaçların genişliklerinin eşit olması ise fonksiyon kümesindeki bütün fonksiyonların aritelerinin eşit olmasına bağlıdır. Şekil 2.10 'da yukarıdaki örnek programın terminal ve fonksiyon kümelerinden yararlanılarak temsilen bir bireyin doldurma yöntemiyle nasıl oluşturulabileceği gösterilmiştir.

İlk popülasyonun oluşturulmasında kullanılan diğer bir yöntem büyüme yöntemidir. Büyüme yöntemi, daha heterojen ağaç oluşturma fikrine dayanır. Buna göre; ağaçların oluşturulmasında kök düğüm hariç hem terminal hem de fonksiyon kümesinden bileşenler kullanımaktadır. Böylelikle kimi ağaçlar, tamamen veya kısmen maksimum ağaç derinliğine ulaşamadan işlem tamamlanabilmektedir. Böylece daha çeşitli yapıda ağaçlardan

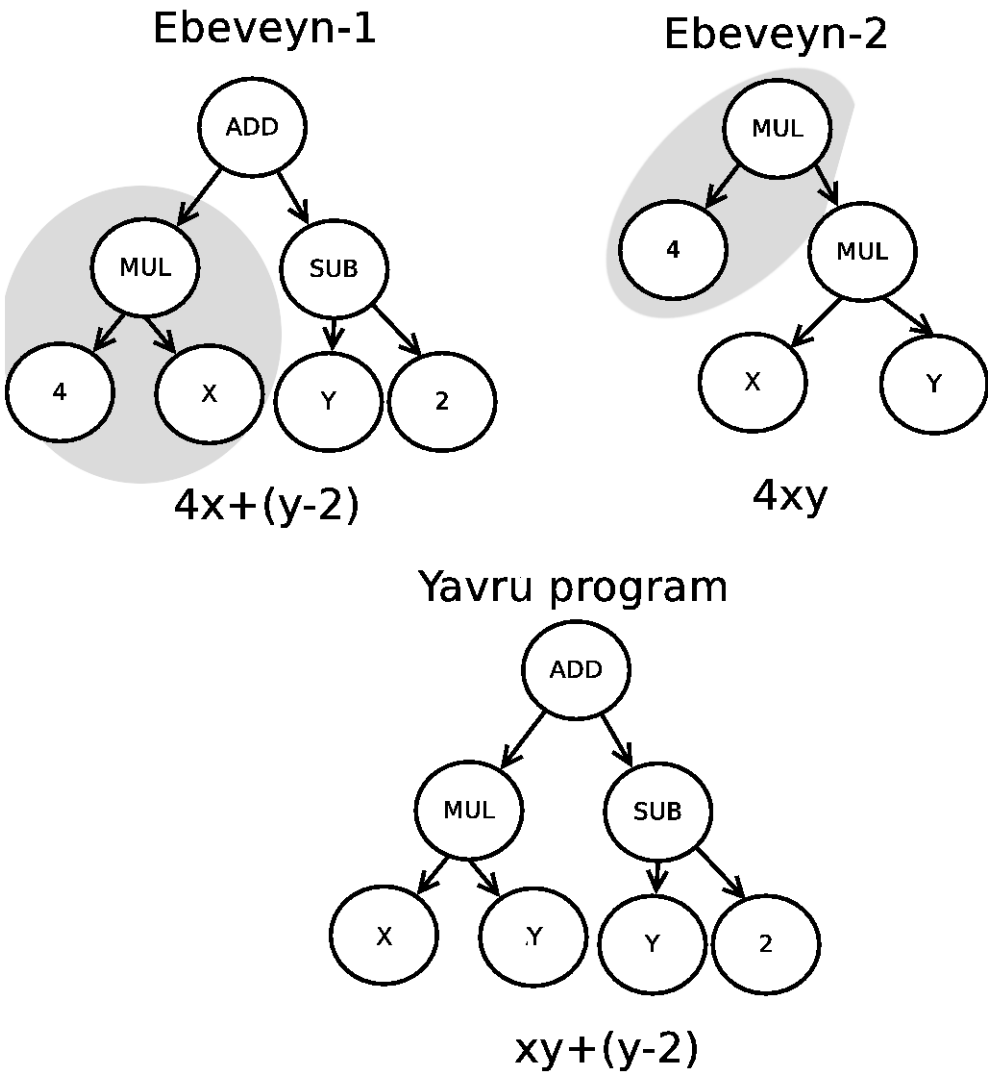


Şekil 2.11: Temsili Bir Bireyin Büyüme Yöntemiyle Oluşturulması Aşamaları

oluşan bir popülasyon elde edilmektedir. Teknik, ağaçların sürgün vererek büyümesine benzediğinden bu isimle anılmaktadır. Şekil 2.11'de örnek programın terminal ve fonksiyon kümesinden yararlanarak temsili bir bireyin büyüme yöntemiyle nasıl oluşturulabileceği gösterilmiştir. Görüldüğü üzere (2) aşamasında ağacın sol tarafı bir terminalle sonlanmış ancak sağ tarafı maksimum ağaç derinliğine kadar uzamıştır.

Diğer bir ilk popülasyon geliştirme tekniği de eğimli yarı yarıya tekniğidir. Burada adından da anlaşılacağı üzere popülasyonun yarısı doldurma, yarısı ise büyüme tekniği kullanılarak oluşturulur. Bu tekniği farklı kılan; rastgele seçilen değişik maksimum ağaç derinlikleri kullanılmasıdır. Nihai popülasyon genç ve yaşlı ağaçların bir arada bulunduğu bir ormanı anımsatmaktadır. 500 bireyden oluşan bir popülasyonda [2,6] arasındaki ağaç derinliklerinin her birinden 100 birey oluşturulma örneğinde; her derinlik için bireylerin yarısı doldurma yarısı da büyüme yardımıyla oluşturulacaktır. Bu durumda popülasyonda, maksimum ağaç derinliği 6 olan 50 tam dolu ağaç ve büyüme yöntemiyle geliştirilmiş maksimum ağaç derinliği 6 olan 50 ağaç yer alacaktır.

Ağaç tabanlı GP'de ilk popülasyonun oluşturulması sonrasında bireylerin değerleme ve seçilim tekniklerinde diğer EA'lardan bir farklılık yoktur. Farklılık yalnızca yazılım bağlamında ortaya çıkabilmektedir. Gerçekten de ağaçlar üzerinde yapılacak işlemler sabit büyüklükteki veri yapıları üzerinde yapılan işlemlere göre daha verimsizdir ve daha

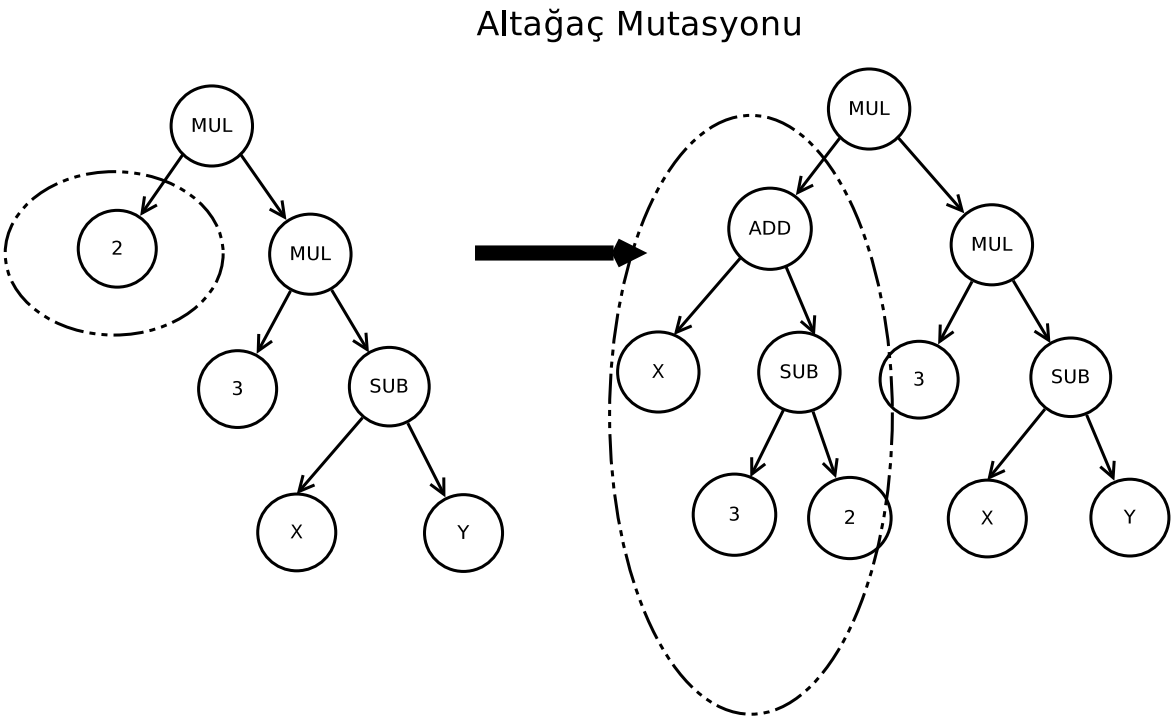


Şekil 2.12: Altağaç Çaprazlaması

çok hesaplama kaynağı tüketmektedir.

Genetik operatörler açısından bakıldığında, GP'de çaprazlamada en çok kullanılan tekniğin *altağaç çaprazlaması (subtree crossover)* olduğu görülmektedir. Bu çaprazlama tekniğinde, önce belirlenen iki ebeveynin kopyalarının üzerinde rastgele birer düğüm belirlenir. Ardından birinci ebeveynde belirlenen düğümden çıkan altağaç atılır, yerine ikinci ebeveynden seçilmiş olan düğümle başlayan altağaç getirilir. Önceki kesimde ilk popülasyon oluşturulmasında kullanılan teknikler yardımıyla elde edilen ağaçlar arasında gerçekleştirilen temsili bir altağaç çaprazlaması Şekil 2.12'de gösterilmiştir.

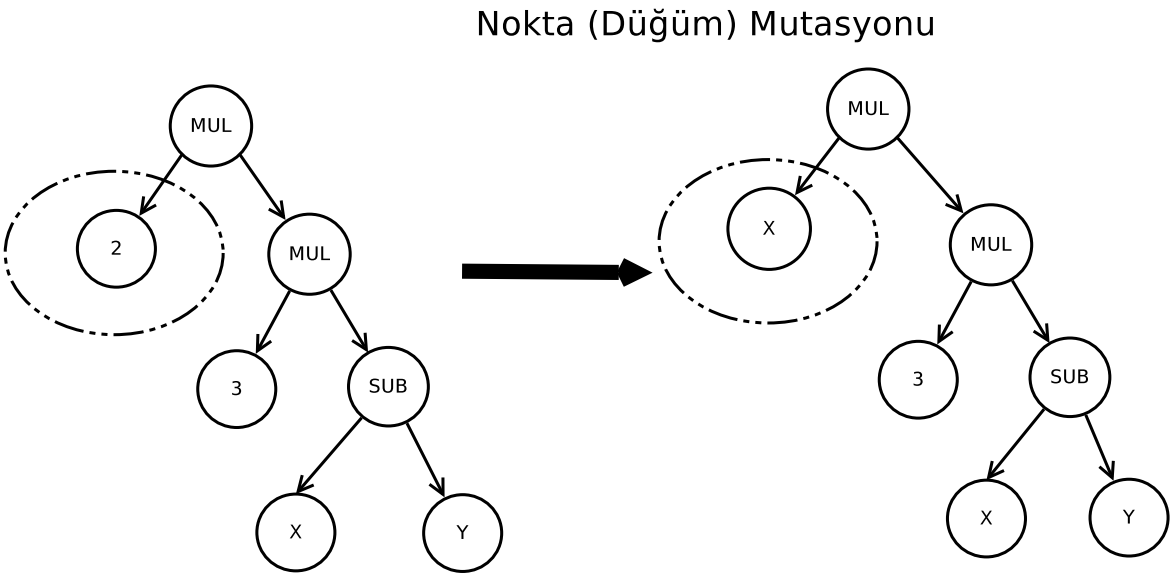
GP'de kullanılan mutasyon operatörleri incelendiğinde iki çeşit operatörün öne çıktığı görülmektedir. Bunlardan birincisi, bireyin rastgele seçilen bir düğümünden aşağı-



Şekil 2.13: Altağaç Mutasyonu

sını iptal ederek yerine maksimum ağaç derinliği sınırları dahilinde olmak üzere yeni bir altağaç türetmektir. Bu operasyon kısmen altağaç çaprazlamasına benzetilmekte ve bazı kaynaklarda da sanki ikinci ebeveyni olmayan bir çaprazlama işlemini andırdığından *başsız tavuk (headless chicken)* mutasyon olarak anılmaktadır (Angeline, 1997). İkinci operatör, rastgele belirlenen düğüm terminal ise ait olduğu terminal kümesinden başka bir elemanla; şayet fonksiyon ise aritesi uyan başka bir fonksiyonla değiştirilmesi ilkesine dayanan *nokta mutasyonudur (point mutation).*

Şekil 2.13'da sol tarafta verilen program ağacının altağaç mutasyonundan sonra alabileceği bir hal sağ tarafta verilmiştir. Burada, üretilen rassal ağacın derinliğinin mutasyon düğümüne eklendiğinde maksimum ağaç derinliğini zedelemediğine dikkat edilmelidir. Benzer şekilde Şekil 2.14'da ise aynı programın nokta mutasyonundan sonra alabileceği bir hal şeklin sağ tarafında verilmiştir. Burada terminal kümesinden tamamen sembolik olarak rastgele X 'in çekildiği varsayılmıştır.



Şekil 2.14: Nokta Mutasyonu

2.2. Genetik Programlamanın Yazılım Yönüne Bakış

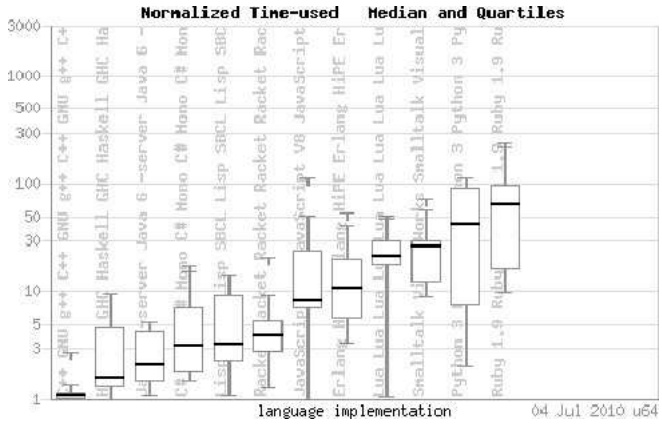
GP kütüphanelerini incelemeye geçmeden önce bu yazılımların geliştirildiği veya geliştirilebileceği programlama dillerinin (daha doğru bir ifadeyle bu programlama dillerinin derleyici, yorumlayıcı ve/veya sanal makinelerinin) *çalışma zamanı* performanslarını, *bellek kullanım* performanslarını ve program *kaynak kodu büyüklüğü* bakımından araştırmacılara getirdiği kolaylıkları ve külfetleri incelemekte fayda vardır.

2.2.1. Programlama dillerinin ve gerçekleştirimlerinin incelenmesi

Günümüzde genel amaçlı veya özel amaçlara yönelik olarak geliştirilmiş yüzlerce[2] ve hatta binlerce [3] programlama dili olduğu bilinmekte ve bunların arasına her sene yenileri eklenmektedir. Bunların arasında genel olarak her türlü yazılım geliştirmek için uygun olanlar olabildiği gibi sadece belirli alanlarda kullanılan diller de bulunmaktadır. Örneğin, `C` dili hemen her alanda kullanılırken `Erlang` dili yaygın olarak telekomünikasyon sistemlerinin programlanmasında kullanılmaktadır. Bu örnekleri çoğaltmak mümkündür. Belirli bir iş için en uygun programlama dilini seçme, söz konusu işin daha verimli ve

[2]Bkz. `http://www.99-bottles-of-beer.net`
(Erişim tarihi: 17.03.2010)

[3]Bkz. `http://en.wikipedia.org/wiki/Programming_language`
(Erişim tarihi: 17.03.2010)



Şekil 2.15: x64 Mimarili Intel® Q6600® İşlemciye Sahip GNU/Linux PC Üzerinde (Tek Çekirdekle) Bazı Gerçekleştirimlerin Çalışma Zamanı Kıyaslamaları

etkin bir biçimde görülebilmesi için elzemdir. Programlama dillerinin incelenmesi GP yazılımları geliştirebilmek ve hatta kullanabilmek için çok önemlidir. GP sistemleri hem kaynak tüketimi açısından iştahlı olabilmekte, hem de diğer GA kodlarındaki gösterimlerden ve veri yapılarından daha karmaşık olanlarını kullanabilmektedir. Bu nedenle diğer GA sistemlerine göre tasarlanması ve kodlanması programcı açısından daha zor olabilmektedir.

Bir programlama dilinin kabaca *spesifikasyon (specification)* ve *gerçekleştirim (implementation)* olmak üzere iki ayrı formu bulunur. Programlama dilinin spesifikasyon formu, dilin tasarlayıcılarının sözdizimsel ve semantik olarak dilin sahip olması gereken özelliklerini belirleyerek, bunları belgeleştirdikleri aşamadaki halidir. Spesifikasyon, dilin yaygınlaşması veya başka koşullar göz önüne alınarak yeri geldiği zaman ulusal veya uluslararası standartlara dönüşebilmektedir. Bu aşamadan sonra dilin niteliklerine göre gerek tasarlayıcıları gerek başka kişi veya kuruluşlarca dilin yorumlayıcı, derleyici veya sanal makineleri gibi gerçekleştirimleri üretilmektedir. Bu bağlamda programlama dillerinin karşılaştırılmasından söz edildiğinde, programa dillerinin kendilerinden çok bunların gerçekleştirimlerin karşılaştırılması ifade edilmektedir.

Günümüzde bu karşılaştırma işlevini yerine getiren bazı programlama *kıyaslamaları (benchmark)* bulunmakta olup bunlardan gerek süreklilik gerekse de kapsam olarak en güveniliri *"The Computer Language Benchmarks Game"*[4] kıyaslamasıdır. Burada, bu ça-

[4]Bkz. http://shootout.alioth.debian.org/
(Erişim tarihi: 01.04.2010)

lışmanın yapıldığı tarih itibariyle 12 problem tanımı için dünyada en çok kullanılan programlama dillerinden yaklaşık otuzuna ilişkin 1100 program bulunmakta ve bunların çalışma zamanı, bellek tüketimi ve kaynak kod büyüklükleri değişik donanım seçenekleri üzerinde kıyaslanarak sonuçlar kamuoyu ile paylaşılmaktadır.

Tablo 2.2: Bir Kısmı Şekil 2.15'deki Grafikte Gösterilen (Medyana Göre Sıralanmış) Çalışma Zamanı Performansları

Gerçekleştirim	**|-**	**|–**	**25%**	**Medyan**	**75%**	**–|**	**-|**
C GNU gcc	1	1	1.01	1.07	1.11	1.26	5.01
C++ GNU g++	1	1	1.02	1.09	1.16	1.37	2.75
ATS	1	1	1.01	1.1	1.32	1.77	5.55
Ada 2005 GNAT	1.01	1.01	1.15	1.48	3.05	5.91	6.66
Java 6 steady state	1.07	1.07	1.33	1.56	1.81	1.91	1.91
Haskell GHC	1	1	1.33	1.61	4.63	9.59	9.67
Lua LuaJIT	1.03	1.03	1.48	1.85	9.16	10.68	10.68
Scala	1	1	1.48	1.9	5.3	6.12	6.12
Java 6 -server	1.08	1.08	1.51	2.07	4.22	5.29	5.29
Fortran Intel	1	1	1.78	2.18	5.56	7.78	7.78
Go 6g 8g	1.45	1.45	2.28	2.44	15.24	34.68	88.59
Pascal Free Pascal	1.36	1.36	1.7	2.61	3.53	6.28	7.47
C# Mono	1.5	1.5	1.79	3.13	7.15	15.2	17.27
Lisp SBCL	1.09	1.09	2.3	3.22	9.33	14.23	14.23
Clean	1.56	1.56	1.9	3.25	7.08	11.37	11.37
OCaml	1.52	1.52	2.62	3.92	4.07	6.25	6.75
Racket	1.28	1.28	2.83	3.94	5.43	9.33	20.86
F# Mono	1.95	1.95	2.66	5.98	13.53	29.84	38.05
JavaScript V8	1	1	7.1	8.14	24.38	50.3	112.63
Erlang HiPE	3.39	3.39	5.82	10.61	20.06	41.41	53.29
Java 6 -Xint	8.65	8.65	10.2	18.66	31.89	64.42	97.8

İzleyen sayfada devam etmektedir.

Lua	1.03	1.03	17.65	21.36	29.46	47.18	51.06
Smalltalk VisualWorks	8.91	8.91	12.32	26.27	29.97	56.45	71.94
JavaScript TraceMonkey	1.95	1.95	9.98	28.49	47.76	104.44	938.89
Python 3	2	2	7.55	42.43	91.74	115.46	115.46
Ruby 1.9	9.94	9.94	16.53	64.24	98.35	221.09	241.59

Tablo 2.2 'de verilen sayısal değerler bu gerçekleştirimler üzerinde çalışan programların çalışma zamanlarını göstermektedir. Karşılaştırma kolaylığı sağlamak amacıyla tabloda medyanlarına göre sıralanmışlardır. Buradan da görüldüğü gibi en hızlı gerçekleştirimler *çalıştırılabilir program (executable)* üreten C ve C++ dilleri ile bunlar gibi düşük seviyeli olan Ada dili derleyicileridir. Ancak günümüzde daha yüksek seviyeli diller için geliştirilen bazı sanal makinelerin de çalışma zamanı performansları öne çıkmaktadır. Başta Java ™ ve .NET™ olmak üzere, bu sanal makineler için geliştirilen programlama dilleriyle yazılan programların çalışma zamanı performansı açısından çok daha iyi seviyelere geldikleri görülmektedir. *Fonksiyonel programlama* paradigmasını esas alan Haskell, OCaml, ATS ve bunların öncülü sayılabilecek Common Lisp için son yıllarda geliştirilen yüksek performanslı derleyicilerin varlığı diğer bir dikkat çekici noktayı oluşturmaktadır.

Günümüzde bilgisayar sistemlerinin konfigürasyonları içerisinde bellek maliyetlerinin işlemci gücü ve programcı zamanı karşısında görece ucuzlaması gerçekleştirimlerin bellek performanslarının, gömülü sistemler gibi özel alanlar haricinde çok önemsenmemesine yol açmaktadır. Ancak programcı ve bilişim uzmanı gereksiniminin giderek arttığı çağımızda; programlama dillerinin kıyaslanması yalnızca bunların referans gerçekleştirimlerinin çalışma zamanlarının analizini değil, aynı zamanda bu dillerde aynı işi yapan iki programın ne kadar programcı zamanına mal olduğunun belirlenmesi yardımıyla da ölçülmeye çalışılmaktadır. Söz konusu kıyaslamalar genellikle bu programların kod uzunluklarının veya büyüklüklerinin ölçülmesi yoluyla yapılmaktadır. Çalışma zamanı kıyaslamaları gibi örnek programların bellek tüketiminin ve kod uzunluklarının (büyüklüklerinin) kıyaslandığı benchmarklar aynı kaynakta mevcuttur. Ancak günümüzde kod büyüklükleri bellek tüketiminden daha önemli hale gelmiştir. Bu nedenle Tablo 2.3'te

kod büyüklüklerine göre programlama dilleri gerçekleştirimlerinin kıyaslamalarının verilmesiyle yetinilmiştir [5].

Tablo 2.3: Benchmarkta Temel Alınan Programların Normalleştirilmiş Kod Uzunlukları

Gerçekleştirim	Geometrik Ort.	Eksik Program Sayısı
Ruby 1.9	1.06	-
JavaScript V8	1.26	2
Lua	1.27	1
JavaScript TraceMonkey	1.27	4
Lua LuaJIT	1.33	1
Python 3	1.37	-
Go 6g 8g	1.73	-
Clean	1.85	4
Smalltalk VisualWorks	1.86	-
Erlang HiPE	2	-
F# Mono	2	3
Scala	2.03	-
Racket	2.03	-
Pascal Free Pascal	2.09	1
Fortran Intel	2.23	4
C# Mono	2.25	-
Java 6 steady state	2.34	3
OCaml	2.39	-
Java 6 -server	2.41	-
Lisp SBCL	2.43	1
İzleyen sayfada devam etmektedir.		

[5] Bkz. `http://shootout.alioth.debian.org/u64/which-language-is-best.php?calc=calculate&xfullcpu=0&xmem=0&xloc=1&binarytrees=1&chameneosredux=0&fannkuchredux=1&fasta=1&knucleotide=1&mandelbrot=1&meteor=0&nbody=1&pidigits=1&regexdna=1&revcomp=1&spectralnorm=1&threadring=0`
(Erişim tarihi: 22.04.2010)

Java 6 -Xint	2.48	-
Haskell GHC	2.49	-
C GNU gcc	2.81	-
C++ GNU g++	3.43	-
Ada 2005 GNAT	4.79	-
ATS	4.93	-

Programlama dili gerçekleştirimlerinin tasarlanması ve uygulanması sırasında, çalışma zamanı performansı ile yüksek seviyelilik arasındaki çatışma yukarıda verilen Tablo 2.2 ve Tablo 2.3 yardımıyla gözler önüne serilmektedir. Çalışma zamanı performansı en iyi olan gerçekleştirimlerin aynı anda en çok programcı çabasına mal olanlar oldukları görülmektedir. Bir programcı tarafından bir derleyiciye ne kadar çok detaylı bilgi aktarılırsa (değişkenlerin tip bilgisi veya belleğin nasıl yönetileceği vb.), derleyici o kadar verimli makine kodu üretecektir. Halbuki yüksek seviyeli diller, programcı yani insana daha yakın makineye daha uzak gerçekleştirimlere sahiptir. Programcıdan olabildiğince az bilgi alıp bu programı yine aynı işi yapan bir makine koduna dönüştürmektedir. Böylelikle daha öz programlarla aynı işin yapılabilmesini sağlamaktadır. Bu üstünlüklerine rağmen daha düşük seviyeli dillere göre çalışma zamanı performansı açısından daha zayıf kalmaktadırlar. GP kütüphaneleri incelenirken bu temel noktaların göz önünde bulundurulup buna göre bir hazır kütüphane veya yeni baştan yazılacaksa bir programlama dili seçilmesi kritik öneme sahiptir. GP uygulamasının geliştirileceği programlama dilinin, GP uygulamalarında karşılaşılabilecek karmaşık veri yapılarını kolayca ifade edip manipüle edebilecek kadar yüksek seviyeli ancak çalışma zamanı performansı açısından da kabul edilebilir bir seviyede makine (veya sanal makine) kodu üretebilecek bir derleyicisi veya yorumlayıcısı (ve/veya çalışma zamanı ortamı) olmalıdır.

2.2.2. **Genetik programlama kütüphaneleri**

Günümüzde evrimsel hesaplama (EC) üç ana kategoriye ayrılmış durumdadır: 1960-70 döneminde ABD'de Holland ve öğrencileri tarafından ortaya atılan Genetik Algoritma-

lar(GA); yine aynı zamanlarda Berlin-Almanya'da Rechenberg ve Schwefel tarafından ortaya atılan Evrimsel Stratejiler (ES) ve son olarak Fogel tarafından yine ABD'de ortaya atılan Evrimsel Programlama (EP). Bu sınıflandırma coğrafi ve tarihi olarak yapılmıştır. Genetik programlama, ayırt edici özelliklerine ve önemine rağmen bu sınıflandırmada genetik algoritmaların altında yer almaktadır (Gagné ve Parizeau, 2006:2).

Evrimsel hesaplama teknikleri için geliştirilen çeşitli kütüphane gerçekleştirimleri bulunmaktadır. Genellikle her teknik için başlı başına yazılmış *ad hoc* kütüphanelerin yanı sıra, araştırmacıların hemen hemen tüm teknikleri kullanabileceği çerçeve (*framework*) gerçekleştirimler de mevcuttur. Örneğin; GALib veya PGAPACK gibi GA kütüphaneleri kullanılarak yalnızca belirli gösterimler için GA modelleri geliştirebilmektedir. Benzer biçimde Little LISP, lil-gp veya puppy gibi GP kütüphaneleri kullanılarak sadece ağaç tabanlı birey gösterimlerine sahip GP modelleri geliştirilebilmektedir. Ancak bunların yanında hemen her çeşit evrimsel algoritmanın değişik problem uygulamalarında kullanılabilmesi için gerçekleştirildiği ve programcıların hizmetine sunulduğu ECJ ve OpenBEAGLE gibi çerçeve kütüphaneler bulunmaktadır. Bu çerçeve kütüphaneleri diğerlerinden ayırt eden temel kriter genelleştirilebilirlikleridir(*genericity*). Bu kriter Gagné ve Parizeau tarafından temel olarak altı alt başlıkta incelenmiştir (Gagné ve Parizeau, 2006:5). Bunlar genel gösterim *(generic representation)*, genel uyum fonksiyonu *(generic fitness)*, genel operasyonlar *(generic operations)*, genel evrimsel model *(generic evolutionary model)*, parametre yönetimi *(parameters management)* ve ayarlanabilir çıktıdır *(configurable output)* (Gagné ve Parizeau, 2006:5-7).

GP uygulamalarında kullanılan gerçekleştirimlerin esas aldıkları yaklaşımlar incelendiğinde genel olarak üç sınıfa ayrıldıkları görülmektedir. Bunlardan ilki LISP ve dolayısıyla yüksek seviyeli bir geliştirme ortamında liste veri yapısının kullanımıdır. Bir diğeri C vb. gibi "derlenen" dillerdeki herhangi bir veri yapısını kullanan ve bu dillerin (doğrudan makine diline derleyen) derleyicilerinden faydalanma amacını güden yaklaşımdır. Üçüncüsü ise doğrudan makine kodunda yani ikili kodda veya "assembly" dillerinden yararlanarak yüksek performans elde etme amacını güden yaklaşımdır (Banzhaf vd., 1997:315). Bu çalışmada incelenen gerçekleştirimler bu sınıflamanın ilk ikisinde yer almaktadır. Üçüncü sınıfta yer alan kütüphaneler hem sayıca az hem de ulaşılabilirliği daha zor olduğundan inceleme dışı bırakılmıştır.

Yapılan literatür incelemesinde aktif durumda bulunulan ve ulaşılabilen GP kütüphanelerine izleyen kesimde değinilmiştir. Bunlardan bazıları sadece GP uygulamaları için geliştirilmiş kütüphaneler iken, diğerleri başkaca EC algoritmalarını kullanan uygulamaları da destekler şekilde yazılmış çerçeve kütüphanelerdir.

1. **Little LISP**

   Little LISP[6] programı, konunun öncüsü John Koza tarafından geliştirlmiştir. Koza'nın konuya ilişkin "Genetic Programming: On the Programming of Computers by Means of Natural Selection" kitabında aktarılan bir GP kütüphanesidir. Common Lisp programlama dilinde yazılmıştır. Bir çekirdek program ve üzerinde probleme göre değişik türde sürücü programlar yazılabilmesini sağlayan rutinlerden oluşmaktadır (Koza, 1992:699-782).

2. **lil-gp Genetic Programming System**

   lil-gp Genetic Programming System, Little LISP sisteminin C programlama diliyle yeniden yazılmış şeklidir. Çalışma zamanı performansı açısından Little LISP'den daha verimlidir[7].

3. **GPC++ (Genetic Programming C++ Class Library)**

   GPC++, C++ diliyle yazılmış nesne yönelimli bir sınıf kütüphanesidir[8].

4. **EO (Evolving Objects)**

   EO[9], C++ ile yazılmış en kapsamlı EC çerçeve kütüphanelerinden biridir (Keijzer; Merelo; Romero ve Schoenauer, 2002).

5. **OpenBEAGLE (BEAGLE Puppy)** Open BEAGLE[10], C++ ile yazılmış bir EC

[6] `http://www.genetic-programming.org/gplittlelisp.html`,
(Erişim tarihi: 07.05.2010)

[7] `http://garage.cse.msu.edu/software/lil-gp/`,
(Erişim tarihi: 07.05.2010)

[8] `http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/weinbenner/gp.html`,
(Erişim tarihi: 10.05.2010)

[9] `http://eodev.sourceforge.net/`,
(Erişim tarihi: 12.05.2010)

[10] `http://beagle.sourceforge.net/`,
(Erişim tarihi: 12.05.2010)

çerçeve kütüphanesidir. Ağaç tabanlı GP'yı da destekleyen çerçeve yardımıyla GP uygulamaları da geliştirilebilmektedir. Sadece GP uygulamaları için ayrı olarak kullanılabilmesini sağlayan BEAGLE Puppy [11] adında ilgili bir kütüphane de geliştirilmiştir (Gagné ve Parizeau, 2002).

6. **ECJ**

   ECJ [12], Java diliyle yazılmış nesne yönelimli bir EC çerçeve kütüphanesidir. Mevcut sistemler arasında en çok özellik (dökümantasyon, birey ağaçlarının LaTeXortamına aktarılabilmesi vb. gibi) barındıranı olduğu söylenebilir.

7. **GenPro**

   GenPro, Java programlama diliyle yazılmış nesne yönelimli bir GP kütüphanesidir[13].

8. **ECF (Evolutionary Computation Framework)**

   ECF, C++ diliyle yazılmış bir EC çerçeve kütüphanesidir[14].

9. **PMDGP - Poor Man's Distributed Genetic Programming**

   PMDGP, C++ diliyle yazılmış nesneye yönelik ve dağıtık bir GP gerçekleştirimidir[15].

Yukarıda kısaca değinilen bu kütüphanelerin bazıları (örneğin Little LISP) yalnızca GP uygulamaları geliştirmek için kullanılabilirken, diğerleri ise GP'nın yanı sıra GA ve ES gibi hemen tüm evrimsel algoritmaların kullanılmasıyla uygulama geliştirilmesine olanak sağlamaktadır. Bu ikinci tip çerçeve kütüphaneler arasında son derece popüler olan

[11] `http://beagle.gel.ulaval.ca/puppy/`,
(Erişim tarihi: 12.05.2010)

[12] `http://cs.gmu.edu/~eclab/projects/ecj/`,
(Erişim tarihi: 15.05.2010)

[13] `http://code.google.com/p/genpro/`,
(Erişim tarihi: 10.05.2010)

[14] `http://gp.zemris.fer.hr/ecf/`,
(Erişim tarihi: 10.05.2010)

[15] `http://pmdgp.sourceforge.net/`,
(Erişim tarihi: 10.05.2010)

ECJ ve Open BEAGLE (Puppy), Evolving Objects ve GPC++ kütüphaneleri bu çalışma kapsamında derlenmeye çalışılmış, Puppy ve ECJ başarıyla derlenip örnek uygulamalar çalıştırılmıştır. Ancak Evolving Objects ve GPC++ kütüphaneleri, en son yayımlanmış kaynak kodları temel alınarak ve geliştiricilerinin sunduğu genel talimatlar doğrultusunda derlenmeye çalışılsa da başarılı olunamamıştır.

Genel çerçeve kütüphanelerin avantajları olabileceği gibi dezavantajları da bulunmaktadır. Bir programcının böyle bir çerçeve sistemi öğrenmek için başlangıçta harcayacağı süre diğer sistemleri öğrenmek için harcayacağı süreye göre çok daha uzundur. Bu olumsuzluk, çerçeve sistemlerin daha çok EC algoritmasını kapsayabilmek için getirdiği ilave soyutlamalardan kaynaklanmaktadır.

John Koza'nın Little Lisp GP kodu, Common Lisp programlama diliyle yazılmıştır. Bir çekirdek program ve bunun üzerinde değişik problemlere ilişkin GP modellerinin geliştirilmesini sağlayacak çeşitli sürücü rutinlerden oluşmaktadır. John Koza bu yazılımı geliştirirken neden Lisp programlama dilini tercih ettiğini yedi maddede aşağıdaki biçimde açıklamaktadır (Koza, 1992:71-72):

1. Lisp'te veri ve programın aynı formda olması *(S-expressions)* program üzerinde manipülasyon yapılabilmesini kolaylaştırmaktadır.
2. Yukarıda ifade edilen özellik sayesinde, bilgisayar programı (S-expression) aynı zamanda bu programın iç gösterimi olan *ayrıştırma ağacı (parse-tree)* ile aynı yapıda olduğundan programcının bu yapıya erişimini kolaylaştırmaktadır.
3. Lisp sistemlerinde yer alan `EVAL` fonksiyonunun bir kod parçasını zahmetsiz bir biçimde yorumlaması, veri halindeki kodun yorumlanmasını basit bir işlem haline getirmektedir.
4. Lisp gerçekleştirimleri çalışma zamanı boyunca dinamik olarak değişen boyutlarda ve şekillerdeki veri yapılarının kullanılmasını sağlamaktadır. Ayrıca *GC (garbage collection)* sayesinde belleğin otomatik olarak yönetilmesini sağlamakta, böylece programcının belleği manuel olarak yönetmesi sorununu ortadan kaldırmaktadır.
5. Lisp, geleneksel hiyerarşik yapıları iyi bir biçimde idare edebilmektedir.
6. Lisp'in en temel konsol çıktısı fonksiyonu olan `print` fonksiyonu *ayrıştırma*

ağaçlarının (parse tree) sunumunda bir çok değişik seçenek sunmaktadır.[16]

7. Programcılar için birçok verimlilik araçlarına sahip kaliteli ticari Lisp uygulamaları bulunmaktadır.

Her ne kadar ilk GP sistemleri Lisp ile yazılmışsa da; literatürdeki uygulamalarda ve raporlarda C ve C++ dilleri gibi derlenebilen dillerle gerçekleştirilmiş GP sistemleriyle giderek daha sık karşılaşılmaktadır. Bu dillerin tercih edilmesinde şu unsurların öne çıktığı bildirilmektedir (Kuhlmann ve Hollick, 1995):

- Performans: C hem hız, hem de bellek kullanımı bakımından yüksek performanslı gerçekleştirimlere sahiptir.
- Taşınılabilirlik: Hemen her donanım platformu için bir C derleyicisinin bulunabilmesi yazılan programların platformlar arası taşınılabilmesini kolaylaştırmaktadır.
- Veri Yapısı Desteği: C'nin değişik veri yapılarını desteklemesi, *özgün* sistemlerin değişik veri yapıları kullanmak suretiyle gerçekleştirilebilmesini sağlamaktadır.
- Alışkanlık: C'nin programlama tarihindeki özel yeri ve çok sayıda programcının bu dile olan aşinalığı programcılara hareket serbestisi sağlamaktadır.

Yukarıda verilen ve Kuhlmann ve Hollick'in raporunda belirttiği bu özelliklere rağmen, bir yandan da C'nin GP gerçekleştirimleri için çok da uygun olmadığı, aynı raporda şu şekilde açıklanmaktadır:

- C, Lisp'in "yorumlanabilen" (etkileşimli) doğasından yoksundur.
- C'de, yine C kodu üreten ve bu kod üzerinde işlemler yapan bir program yazmak gerçekten zorlu bir iştir.
- C'de, otomatik bellek yönetimi (garbage collector) olmadığından, bellekte dinamik olarak bireylerin ve popülasyonların idare edilmesi gerekmektedir. Bu durum, GP gerçekleştirimlerinin C'de geliştirilmesini çok daha zor hale getirmektedir.

[16] Modern Common Lisp ortamlarındaki `format` fonksiyonu bu konuda çok daha ileri düzeyde seçenekler sağlamaktadır.

GP gerçekleştirimlerinde; yalnızca bu sayılan dezavantajlar bile, C'den çok Lisp'in tercih edilmesi için yeterli olabilir. Gerek son yıllarda Common Lisp (CL) derleyicilerinin ve ortamlarının geldiği nokta, gerekse çok sayıda alternatif Lisp gerçekleştiriminin varlığı Lisp'i daha çekici hale getirmektedir. Diğer EC algoritmaları gibi, GP'de de asıl iş yükünün bireylerin uyum fonksiyonlarının hesaplandığı *değerleme (evaluation)* aşaması olduğu bilinmektedir. Bazı çalışmalar göstermektedir ki; ağaç tabanlı GP gösterimlerinde, bireylerin (program ağaçlarının) derinliği arttıkça, CL gerçekleştirimleri C gerçekleştirimlerini çalışma zamanı performansı olarak da geride bırakmaktadır (Svingen, 2006:2); Verna, 2006). Ayrıca; GCL ve ECL gibi Lisp derleyicileri, *konakçı-ev sahibi (client-host)* mimarisini kullanarak bir C derleyicisi (örneğin GCC) yardımıyla C kodu üretebilmektedir. Dolayısıyla C derleyicisinin çalıştığı her platform için CL'yi önce C'ye ardından da hedef platformun ikili koduna derleyen *konak* CL derleyicileri, taşınılabilirlik konusundaki sorunu tamamen ortadan kaldırmaktadır.

Çalışmanın önceki kesimlerinde değinilen, programlama dilleri gerçekleştirimlerinin günümüzdeki durumu göz önüne alındığında, bu tez kapsamındaki çalışmaların da Lisp dilinin günümüzde yaşayan lehçelerinden biri olan Common Lisp dili yardımıyla gerçekleştirilmesinin uygun olacağı düşünülmüştür. Bunun başlıca sebebi; Koza'nın sıraladığı Lisp'in GP uygulamalarındaki avantajlarının yanı sıra, kaliteli Common Lisp derleyicilerinin ürettiği derlenmiş kodların çalışma performansı açısından ulaştığı noktadır. Common Lisp derleyicilerinin bu bağlamda aşağı seviyeli dillerden hemen sonra yer alacak kadar başarılı olması dikkate değerdir. Dahası, bu başarıyı sıralamada kendisiyle hemen hemen aynı seviyelerde yer alan JVM ve .NET platformları için geliştirilmiş orta seviyeli dillerin (örn.: Java, C# vd.) aksine ardında herhangi bir ticari kurumun itici gücü olmadan elde etmeleri takdir edilmelidir. Ayrıca; Common Lisp'in bahsi geçen dillerden daha yüksek seviyeli olduğu dikkate alındığında "performans-yüksek seviyelilik" arasındaki ters yönlü ilişkide, *ağaç tabanlı GP uygulamaları özelinde* optimum noktaya çok daha yakın olduğu düşünülmektedir.

3. Finansal Veri Yönetimi ve Veritabanları

Bilindiği gibi, genetik programlamanın özellikle finansal uygulamalarının genel karakteristiği ampirik (data driven) olmasıdır. Uygulamalar çok miktarda kaliteli veriye ihtiyaç duyar. Bu nedenle; ihtiyaç duyulan yüksek miktardaki verinin elde edilmesi, depolanması, erişilebilmesi, değiştirilebilmesi ve güncellenmesi amacıyla veritabanı altında tutulması bir zorunluluk olarak ortaya çıkmaktadır. Bu durum aslında çoğu ampirik teknik için geçerlidir. Ancak, uygulamacılarca bu tür modellerin bu yanının üzerine kanımızca yeterli düzeyde düşülmemektedir.

Bu bağlamda, çalışma kapsamında İstanbul Menkul Kıymetler Borsası (İMKB)' de işlem gören menkul kıymetlerin genel bilgilerinin ve günlük piyasa verilerinin uygun bir biçimde saklanması ve güncellenmesi için bir veritabanı geliştirilmiştir. Sözü edilen bu aşama GP uygulaması öncesindeki en son ve en önemli altyapı çalışmasını oluşturmaktadır.

3.1. Finansal Veri Sağlayıcıları ve Veritabanları

Finansal veri; para, emtia ve menkul kıymet piyasalarından sağlanır. Bu verinin saklanması ve üçüncü kişilere iletilmesi günümüzdeki veritabanı uygulamalarının belki de en çok ticari potansiyel barındıranlarından biridir. Ulusal ve uluslararası politik ve makro ekonomik gelişmelere ilişkin haberler, şirket haberleri, mali tablolar ve şirketlere ilişkin temettü ödemeleri gibi diğer enformasyon da bütünleşik olarak bu veritabanlarında saklanmaktadır. Bunu, gerektiğinde yatırım kararlarına destek olacak biçimde yatırımcılara elektronik ortamda gerçek zamanlı iletilmesi izlemektedir. Bunlardan dolayı finansal veri sağlayıcılığı başlı başına bir sektör haline gelmiştir.

Günümüzde dünyada, başta ABD ve AB ülkelerinde olmak üzere, sadece finansal bilgilerin gerçek zamanlı dağıtımını yapmaya odaklanmış birçok finansal veri sağlayıcısı

bulunmaktadır[1].

Gelişen bilgisayar ve internet teknolojisi, finansal veri sağlayıcısı şirketlere yukarıda değinilen fonksiyonların yanı sıra başka boyutlarda hizmet sağlama olanakları da kazandırmıştır. Bunlar, genellikle bireysel kullanıcılara sunulan ve gerçek zamanlı verinin ancak bir teknik analiz yazılımı gibi aracı bir program ile sunulduğu ürünlerle gerçekleşmektedir. Kimi ikincil veri sağlayıcısı şirketler ise bu birincil dağıtım kanallarından sağladıkları finansal veriyi değişik yollarla kullanıcılarıyla paylaşmaktadır. Bunlar genellikle Yahoo! ve Google gibi arama motorlarının finans portallarında örneklerine rastlanan ABD ve dünyadaki birçok endeksin ve menkul kıymetin anlık değerlerini web tabanlı uygulamalarla takip edebilme yeteneğini kullanıcılara kazandıran uygulamalardır[2].

Özet olarak; birincil finansal veri sağlayıcıları, piyasa verisini oluştuğu borsalardan alıp kurumsal (banka, aracı kurum vb.) veya (piyasa takip programlarıyla kısıtlı olarak) bireysel ürünlerle nihai kullanıcılara veya ikincil veri sağlayıcılara (örn. medya kuruluşları, finans portalları vs.) ulaştırmaktadır.

Türkiye'de işleyişi temelde farklı olmamakla birlikte, birincil veri sağlayıcıları sayıca daha azdır. İMKB verilerinin dağıtılması ve yayınlanması konusunda yetkili veri yayın kuruluşları İMKB tarafından belirlenmektedir. Bu çalışmanın yapıldığı dönem itibariyle İMKB'nin ilgili web sayfasında belirttiği 5 birincil (alıcı) ve 96 ikincil (alt-alıcı) veri yayın kuruluşu bulunmaktadır [3]. Günlük tarihi veriler, ikincil (alt-alıcı) yayın kuruluşlarından bazılarınca kullanıcılara MS Excel veya `csv` formatında sunulmaktadır. Bu, bilimsel veya kişisel amaçlı yapılacak çalışmalarda günlük verilerin elde edilmesini kolaylaştır-

[1] Belli başlı finansal veri sağlayıcıları için bkz.:

`http://thomsonreuters.com/about/`

`http://www.bloomberg.com/about`

`http://en.wikipedia.org/wiki/Financial_data_vendor`

(Erişim tarihi: 03.02.2011)

[2] Yahoo! Finance ve Google Finance için bkz.:

`http://finance.yahoo.com`

`http://www.google.com/finance`

(Erişim tarihi: 03.02.2011)

[3] İMKB veri yayın kuruluşları listesini incelemek için bkz.:

`http://www.imkb.gov.tr/Data/DataDissemination/DataVendorsList.aspx`

(Erişim tarihi: 19.02.2011)

maktadır.

Sonuç olarak, veri ihtiyacı kalıcı ve büyük ölçekli bir proje için gerekli olduğunda, finansal verilerin veritabanı dışında saklanması uygun bir yaklaşım değildir. Bu nedenle, bu çalışmanın stratejisini de ikincil veri sağlayıcılarından elde edilen günlük (tarihi) veriden oluşan bir veritabanı kurma oluşturmuştur.

3.2. Veritabanı Tasarımı

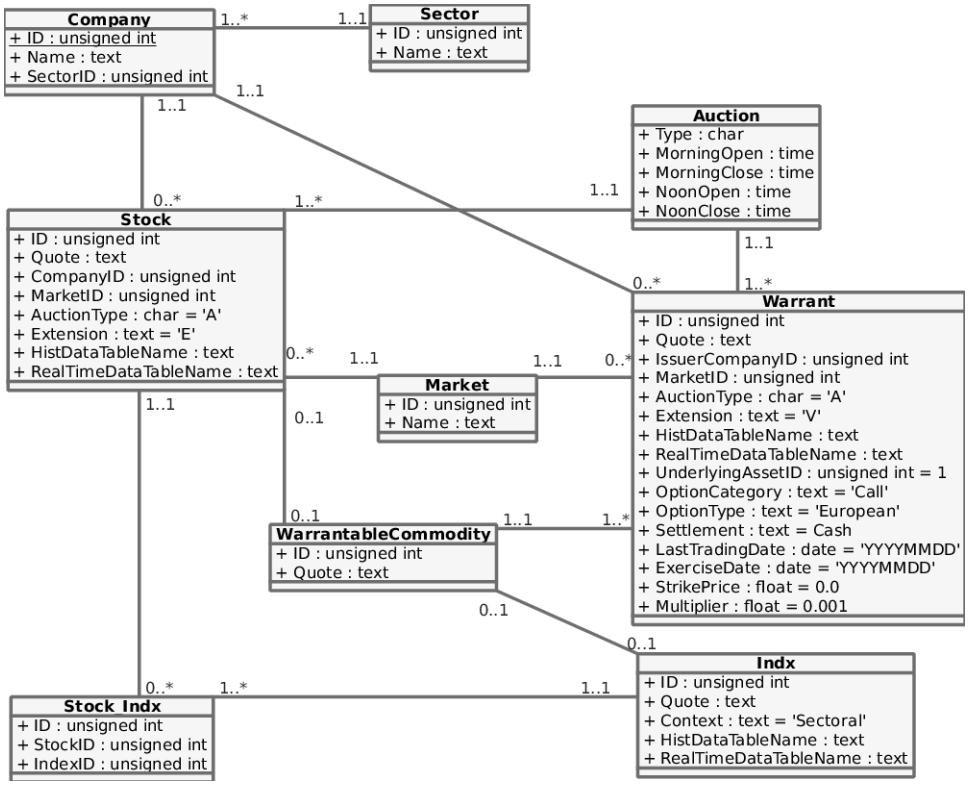
Veritabanı gerçekleştirimine geçilmeden önce titiz bir biçimde tasarlanması zorunluluğu, veritabanları konusunda bütün uzmanların üzerinde birleştiği noktadır. Bu doğrultuda yapılan araştırmalar ve eskizler uyarınca; menkul kıymetlerin ait oldukları şirket ve işlem gördükleri pazar bilgileri, dahil edildikleri endeksler ve destekleyici diğer tüm enformasyonun günlük piyasa verilerinden ayrı bir veri tabanı şemasında gösterilmesi yönünde bir yaklaşıma gidilmiştir. Bu amaçla *"şirket bilgilerinden"* ve *"tarihi verilerden"* oluşan iki ayrı veritabanı (veritabanı grubu) oluşturulması seçeneği benimsenmiştir. Bu yaklaşımın benimsenmesi konusunda, ilişkisel veritabanı tasarımı mantığı çerçevesinde gereksiz veri yinelenmesini (redundancy) engellemek amacıyla tasarımda normalizasyon sürecinin izlenmesi etkili olmuştur (Churcher, 2007:139-155).

3.2.1. **Şirket bilgileri veritabanı**

Şirket bilgileri veritabanı; hisse senetleri İMKB'de işlem gören şirketler, bunların hisse senetleri, bu hisse senetlerine dayalı olarak hesaplanan ve bunların değişik niteliklerine göre yer aldıkları çeşitli endeksler ve son olarak hisse senetleri ve endeksleri dayanak varlık olarak kullanan türev araçlar (varantlar) arasındaki ilşkileri barındıran veritabanıdır. Şirket bilgileri veritabanının geliştirilmesinin temel nedeni, tarihi verilerin saklanacağı veritabanlarının oluşturulması ve oluşturulma sonrası kullanımında gerekli sınıflandırma bilgisinin daha önceden hazır hale getirilmiş olması gerekliliğidir.

3.2.1.1. ***Veri modeli***

Şirket bilgileri veritabanının oluşturulması amacıyla ilk önce İMKB'de işlem gören menkul kıymetler ve bunların özellikleri göz önüne alınarak bir UML sınıf diyagramı yardı-



Şekil 3.1: UML Sınıf Diyagramı Yardımıyla Gösterilen Şirket Bilgileri Veritabanına Ait Veri Modeli

mıyla veri modeli oluşturulur. Çalışmanın veri modeli Şekil 3.1'de verilmiştir.

Bu veri modelinde; sektör, şirket, pazar, seans, hisse senedi, varant (menkul kıymetleştirilmiş opsiyonlar), endeks, varant olabilecek varlık ve hisse senedi - endeks ilişkilerini gösteren toplam 9 adet sınıf (tablo) öngörüldüğü ve bu sınıfların nesneleri (kayıtlar) arasındaki çokluk ilişkilerinin gösterildiği görülmektedir.

Şirket bilgilerinin güncel ve sağlıklı bir biçimde otomatik olarak elde edilmesi sanılabileceğinden zor bir iştir. Neredeyse her gün; bir çok yeni şirket halka arz olmakta, hisse senetleri dışında yeni menkul kıymetler işlem görmeye başlamakta, bazı hisse senetlerinin işlem gördükleri pazarlar ya da dahil edildikleri endeksler değişmekte ve hatta bazı hisse senetleri işlem görmekten tamamen men edilebilmektedir. Bu örnekler, sermaye piyasası otoritelerince gerçekleştirilen değişik düzenlemelerden yalnızca bazılarıdır. Menkul kıymet piyasalarının dinamik yapısı içinde yeni uygulamalarla karşılaşmaya sıklıkla rastlanılmaktadır. Bu nedenle; şirketler, hisse senetleri ve endekslerle ilgili genel bilgileri elde etmede İMKB'nin kendi web sayfasında bulunan "Şirketler > HSP Şirket Bilgileri"

sayfası kullanılmıştır [4].

3.2.1.2. *Veritabanı yönetim sistemi seçimi*

Günümüzde, çok sayıda veritabanı yönetim sistemi (Database Management System - DBMS) yazılımı mevcut olup halen yenilerinin geliştirilmesine devam edilmektedir. Bunlar arasında çok çeşitli veritabanı paradigmalarını destekleyen ve her türlü ölçekteki işlevselliği karşılayabilecek nitelikte yazılımlar bulunmaktadır. Gerek ticari olarak kurumsal yazılım şirketlerince gerekse kar amacı gütmeyen bağımsız organizasyonlar veya bireylerce geliştirilen çok sayıda DBMS mevcuttur. Nesne yönelimli programlama paradigmasının yükselişe geçtiği seksenli yıllardan bu yana DBMS paradigmaları arasında *ilişkisel veritabanı yönetim sistemleri (Relational Database Management Systems - RDBMS)* öne çıkmıştır. RDBMS listesine günümüze gelinceye kadar çok sayıda yazılım eklenmiştir. [5].

Doksanlı yıllardan itibaren nesne yönelimli programlama paradigmasını destekleyen programlama dilleri genel kabul görmeye başlamıştır. Bu dillerin getirdiği avantajlardan daha fazla yararlanabilmek amacıyla görece daha yakın zamanda *nesne-ilişkisel veritabanı yönetim sistemleri- (Object RDBMS)* adı verilen DBMS'ler de kullanılmaya başlamıştır. Bu yazılımlar nesne yönelimli dillerin desteklediği veri tipi mirası ve nesnelerin izlenebilmesi gibi özellikleri bünyelerinde barındırmaktadır[6].

Semantik web ve Web 2.0 uygulamalarının son yıllarda geldiği nokta, veritabanı ve DBMS paradigmalarında büyük değişimlere yol açmıştır. Geleneksel RDBMS'lerinin yerine MongoDB veya CouchDB gibi *NoSQL* olarak adlandırılan, ilişkisel veritabanı şema tasarımlarını ve SQL dilini kullanmayan DBMS'ler ortaya çıkmıştır. Bu yazılımlar dağıtık ve ölçeklendirilebilir web tabanlı uygulamaların yüksek performans gereksinmelerini

[4] İlgili sayfaya erişmek için bkz.:

`http://www.imkb.gov.tr/sirketler/sirketler.aspx`

(Erişim tarihi:22.01.2011)

[5] Mevcut RDBMS'lerin yetenekleri, destekledikleri veritabanı paradigmaları ve lisans bilgileri bakımından daha fazla bilgi için bkz.:

`http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems`

(Erişim tarihi:22.01.2011)

[6] ORDBMS'ler ile ilgili daha fazla bilgi için bkz.:

`http://en.wikipedia.org/wiki/Object-relational_database`

(Erişim tarihi:22.01.2011)

karşılayabilmek adına alışılagelen ilişkisel modeli terk etmiştir. NoSQL DBMS'ler veri modellerinde klasik tablo yapısı kullanmadıklarından temel sorgulama dili olarak SQL'i tercih etmemektedir [7].

Yukarıda kısaca değinilen paradigmalar arasında bilinirliliği ve tasarım kolaylığı açısından ilişkisel model hala öne çıkmaktadır. Çalışma kapsamında gerçekleştirilen uygulamanın çalışacağı donanımın tekil bir kişisel bilgisayar ya da iş istasyonu olduğu düşünüldüğünde ilişkisel modeli destekleyen bir DBMS'nin yeterliliği daha iyi anlaşılmaktadır. Çalımadaki veritabanlarının yönetimi için gerekli olan DBMS'nin geleneksel RDBMS'leri arasından seçiminde aşağıdaki kriterler göz önünde bulundurulmuştur:

- İlişkisel veritabanı paradigmasını temel almalıdır.
- Kolay kurulup ayarlanabilmelidir.
- SQL standardını (SQL 92) desteklemelidir.
- İstemci-sunucu mimarisinden çok, çalıştırılabilir program ve/veya kütüphane şeklinde kullanımı desteklemelidir.
- Değişik programlama dilleri için arayüzlere sahip olmalıdır.
- Yeterli performans sağlayabilmelidir.
- Esnek bir lisansa sahip olmalıdır.
- Yeterli dökümantasyona sahip olmalıdır.

Çalışmada yukarıdaki kriterleri fazlasıyla sağlayan SQLite DBMS yazılımı tercih edilmiştir. SQLite DBMS'nin kurulum, ayarlama ve kullanımına ilişkin talimat ve bilgiler resmi internet sitesindeki dökümantasyonun yanı sıra, (Newman, 2004) ve (Kreibich, 2010) gibi kaynaklarda detaylı bir biçimde bulunmaktadır. Bütün geliştirme aşamaları ve testlerde programın 3.7.2 versiyonu kullanılmıştır[8].

[7] NoSQL veritabanları ve DBMS'ler ile ilgili daha fazla bilgi için bkz.:

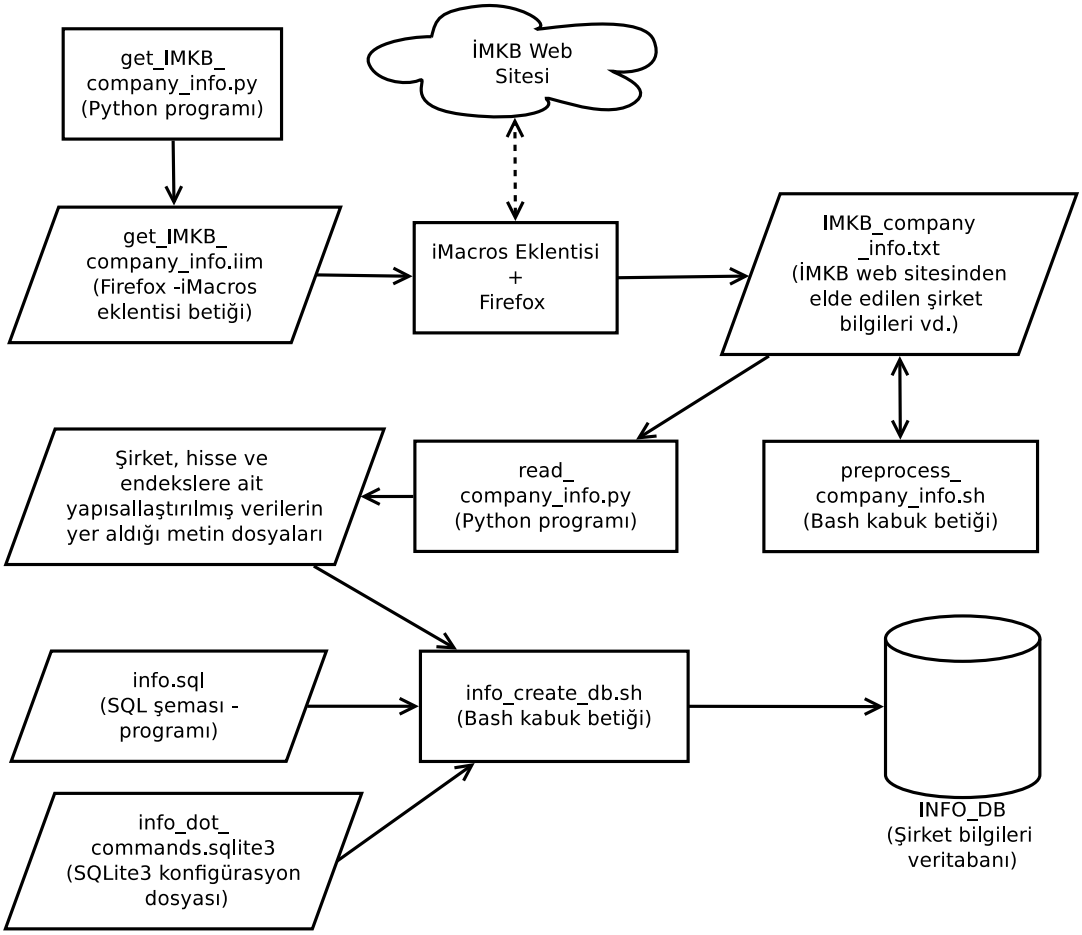
`http://en.wikipedia.org/wiki/NoSQL`

(Erişim tarihi:22.01.2011)

[8] SQLite DBMS için bkz.:

`http://www.sqlite.org/about.html`

(Erişim tarihi:22.01.2011)



Şekil 3.2: Şirket Bilgileri Veritabanının Oluşturulma Aşamaları

3.2.1.3. ***Gerçekleştirim ve veri girişinin otomasyonu***

Şirket bilgileri veritabanının gerçekleştirimi; İMKB web sitesinden verilerin elde edilmesi, bunların düzenlenmesi, veri tabanına doğrudan aktarılabilir hale getirilmesi, veritabanının veri modeline uygun bir biçimde oluşturulması ve bu verilerle doldurulması gibi aşamalara ayrılmaktadır. Veritabanının oluşturulması esnasında izlenen aşamalar Şekil 3.2'de verilmiştir.

Veri modeli Şekil 3.1'de gösterilen veritabanının gerçekleştirimi için geliştirilen SQL programı Ek-B.1'de verilmiştir. Bu SQL dosyasını çalıştırdıktan sonra şirket bilgileri veritabanını kullanılabilir hale getiren ikinci SQL dosyası da Ek-B.2'de yer almaktadır. Söz konusu bu iki SQL programı, İMKB şirket bilgileri veritabanını oluşturmada kullanılan temel SQL programlarıdır. Bu aşamadan sonra, veritabanına girilecek verileri elde etme ve bunların veritabanına girilmesi aşamasına geçilir.

Verileri elde etme ve veritabanına girişi süreci büyük ölçüde otomatik hale getirilmiştir. Bu amaçla İMKB web sayfasındaki şirket bilgilerini içeren interaktif tablodan verileri çıkartıp bir metin dosyasına eklemesi için Firefox tarayıcısı üzerinde çalışan iMacros eklentisi için gerekli makroları üreten bir Python programı geliştirilmiş olup, bu program Ek-B.3'te verilmiştir. Program, Firefox tarayıcısını İMKB'nin ilgili sayfasında çalıştırıp, şirket bilgilerini metin dosyasına yazdıran (aşağıda sadece bir bölümü verilen) iMacros makrolarını üretmektedir. iMacros, web sayfalarından içerik çıkartmak gibi rutin işlemleri otomatikleştirmeye yardımcı olan bir Firefox eklentisidir [9]. Bu işleme, web sayfalarından içerik elde etmek amacıyla normal bir kullanıcının davranışlarını, özel amaçlı bilgisayar programları ya da internet tarayıcılarıyla taklit etmek yoluyla, bilgi ve içerik çıkartmak anlamına gelen *"web scraping"* denmektedir [10].

```
VERSION BUILD=7021019 RECORDER=FX
TAB T=1
URL GOTO=http://www.imkb.gov.tr/sirketler/sirketler.aspx
SET !EXTRACT_TEST_POPUP NO
TAG POS=1 TYPE=A ATTR=ID:ctl00_cphContent_ctl00_lbtnA

TAG POS=1 TYPE=A ATTR=TXT:1
WAIT SECONDS=5
TAG POS=1 TYPE=TABLE ATTR=TXT:* EXTRACT=TXT
SAVEAS TYPE=EXTRACT FOLDER=* FILE=IMKB_company_info.txt

TAG POS=1 TYPE=SPAN ATTR=TXT:2
WAIT SECONDS=5
TAG POS=1 TYPE=TABLE ATTR=TXT:* EXTRACT=TXT
SAVEAS TYPE=EXTRACT FOLDER=* FILE=IMKB_company_info.txt
```

[9] Diğer tarayıcılar için de sürümleri olan bu programla ilgili daha fazla bilgi için bkz.:
`http://www.iopus.com/imacros/`
(Erişim tarihi:11.10.2010)

[10] Daha fazla bilgi için bkz.:
`http://en.wikipedia.org/wiki/Web_scraping`
(Erişim tarihi:09.01.2011)

```
TAG POS=1 TYPE=SPAN ATTR=TXT:3
WAIT SECONDS=5
TAG POS=1 TYPE=TABLE ATTR=TXT:* EXTRACT=TXT
SAVEAS TYPE=EXTRACT FOLDER=* FILE=IMKB_company_info.txt
```

Yukarıdaki örnek makro; İMKB'nin ilgili sayfasında şirket bilgilerinin sunulduğu dinamik içerikli tablodan *'A'* harfi ile başlayan şirket ünvanlarına sahip şirketler için istemde bulunduktan sonra, sunucu tarafından içeriklendirilen bu tablonun ilk üç tabındaki bilgilerin Firefox tarafından kopyalanıp bir metin dosyasına yazdırılmasını sağlamaktadır. Elde edilen makro dosyasının tamamı çok uzun olduğundan bütününe yer verilmemiş olup yukarıdaki kısa örnekle yetinilmiştir. Benzer şekilde makronun geri kalanı IMKB web sitesinden alfabetik sırayla bütün şirketlerin bilgilerini Firefox ve iMacros eklentisi yardımıyla programın çalıştığı bilgisayardaki bir metin dosyasına aktarmaktadır.

İMKB şirket bilgileri metin dosyasına böylelikle alınmıştır. Bundan sonra yapılan incelemede, bunların veritabanına yerleştirilmesi için öncelikle bu metin dosyasının bir düzenlemeye ihtiyacı olduğu belirlenmiştir. Bu düzenlemenin manuel olarak gerçekleştirilmesinin öncelikle zaman ve çaba kaybına yol açacağı görülmüştür. Bu büyüklükteki bir veri yığınının el ile düzenlenmesi insandan kaynaklanan hata yapma eğilimini kesinlikle arttıracaktır. Çalışmada, bu işlemi gerçekleştiren bir `bash`[11] kabuk betiği geliştirilmiştir. Aşağıda kaynak kodu verilen betik temelde boşluk ve diğer gereksiz karakterleri yok etmek amacıyla `sed` [12] programı yardımıyla *yerinde düzenleme (in-place editing)* işlemi gerçekleştirmektedir.

[11] `bash`, günümüzde türlü Unix benzeri işletim sistemlerinde kullanılan çok popüler bir kabuktur. Daha fazla bilgi için bkz.:

`https://en.wikipedia.org/wiki/Bash_(Unix_shell)`

`http://www.gnu.org/software/bash/`

(Erişim tarihi:09.01.2011)

[12] `sed`, Unix benzeri sistemlerde otomatikleştirilmiş metin düzenleme işlemlerinde kabuk betikleri aracılığıyla sıklıkla kullanılan bir yardımcı programdır. Daha fazla bilgi için bkz.:

`http://en.wikipedia.org/wiki/Sed`

(Erişim tarihi:09.01.2011)

```
#!/bin/bash
# preprocess_company_info.sh
# A simple script for editing raw text files extracted by Firefox.

DIR="/home/thorin/iMacros/Downloads/"
FILE="IMKB_company_info.txt"

cd $DIR
echo "Changed current directory to:"
echo ‘pwd‘

sed -i ’s/^\s*//g’ $FILE
sed -i ’s/[\t"]*//g’ $FILE
sed -i ’s/^.[a-z0-9].*//g’ $FILE
```

Şirket bilgilerini barındıran metin dosyasına böylelikle bir ön düzeltme uygulanmıştır. Sonrasında, verilerin veritabanına doğrudan okunabilmesi amacıyla; bu düzenlenmiş metin yığınını barındıran dosyanın, veri modeline uygun olarak yapısallaştırılmış metin dosyalarına parçalanması gereklidir. Bu yapısallaştırılmış metin dosyaları sektörleri, menkul kıymet pazarlarını, şirket ünvanlarını, endeksleri, hisse senedi ve varant sembollerini ve bunların hangi endekslerde yer alacağı bilgilerini barındırır. Bu yapısallaştırılmış metin dosyalarının elde edilebilmesi için bir Python programı geliştirilmiştir. Bu program, şirket bilgileri veritabanı için veri girişi işlemi gereksinimini böylelikle gidermiş bulunmaktadır. Programın kaynak kodu Ek-B.4’te verilmiştir.

Bu veritabanının oluşturulması aşamasında SQLite komut satırı uygulamasının ve SQL kaynak dosyalarındaki SQL komutlarının senkronize bir biçimde kullanılması zorunludur. Bu zorunluluk interaktif olarak SQLite komut satırı uygulamasına baş vurmayı gerektirmektedir. Bu yaklaşımın yerine bir diğer `bash` kabuk betiği geliştirilmesi uygun görülmüştür. Bu amaçla geliştirilen kabuk betiği Ek-B.5’te verilmiştir. Böylelikle bu son betik şirket bilgileri veritabanını kullanıma hazır hale getirmektedir.

3.2.2. Günlük tarihi veriler veritabanı

Günlük tarihi verilerin (günlük fiyat ve işlem hacmi verilerinin) ayrı bir veritabanında tutulacağına daha önce değinilmişti. Bu stratejiyi izlemenin gerekçesi, tarihi verilerin ilişkisel veri tabanı mantığında gösterilmesinin zorluğudur. Bu zorluğa, tarihi verileri tutan tabloların hacim olarak çok kolay büyüyebilmeleri de eklendiğinde bunların ayrı bir veritabanı dosyasında yer alması fikri pratik olarak öne çıkmaktadır. Bu veritabanı dosyasında İMKB'de işlem gören her menkul kıymete ait tarihi veriler ayrı tablolarda saklanmaktadır. Tablolardaki bir kaydı; tarih (gün), açılış fiyatı, güniçi en yüksek fiyat, güniçi en düşük fiyat, kapanış fiyatı ve hacim miktarı alanları oluşturmaktadır. Şirket bilgileri veritabanında yer alan her menkul kıymetin fiyat tablolarıyla ilişkisi ise yalnızca tablo adları yardımıyla kurulmaktadır. Bu yüzden; şirket bilgileri veritabanında hisse senedi, varant veya endeks gibi fiyat bilgisi oluşan varlıkların `"HistDataTableName"` adlı niteliklerinde, tarihi veriler veritabanında günlük verilerinin tutulduğu tablonun adı yer almaktadır.

Daha önce de değinildiği gibi tarihi verileri ikincil veri sağlayıcılarından elde etmek mümkündür. Bu amaçla yapılan araştırmada, Yahoo!Finance portalının ülkemizdeki benzerlerinden olan MynetFinans üzerinden günlük verilerin elde edilebileceği kanısına varılmıştır. MynetFinans üzerinde bulunan teknik analiz sayfalarında kullanıcıların tarihi verileri bilgisayarlarına *'csv'* formatında aktarabilmeleri için bağlantılar bulunmaktadır. Bu bağlantılar sayesinde tarihi verilerin otomatik olarak edinilebileceği ortadadır.

3.2.2.1. Günlük tarihi verileri çeken yazılımının geliştirilmesi

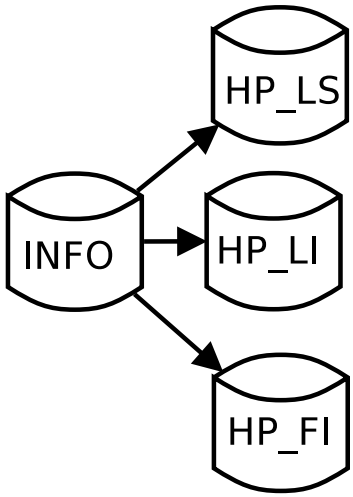
Yukarıda değinilen tarihi verilerin elde edilmesinin yanı sıra uygulama esnasında diğer veri ihtiyaçları da doğabilir. Bunların tümünü karşılayacak şekilde MynetFinans web sayfasından veri elde etme amacıyla kullanılmak için geliştirilen Python modülü Ek-B.6'da verilmiştir.

Bu modülde bulunan `"get_historical_data"` fonksiyonu, İMKB'de işlem gören herhangi bir menkul kıymetin tarihi verilerini istenen aralık için çekme yeteneğine sahiptir.

3.2.2.2. *Günlük tarihi verileri saklayan veritabanının oluşturulması*

Yukarıda değinilen Python modülü ve SQLite3 veritabanı yönetim sisteminin Python arayüzü kullanılarak bir program geliştirilmiştir. Bu programın içinden şirket bilgileri veritabanına erişip buradan ulusal endekslerdeki hisse senetlerinin yanı sıra Dow Jones, FTSE, Nikkei vd. gibi yurtdışı endekslerden 46 endeksin günlük verilerini bir veritabanı dosyasına kaydetmektedir. Söz konusu Python programı Ek-B.7'de verilmiştir. Yurt dışı endekslerin tarihi verileri ise Yahoo! Finance sitesinden elde edilmektedir.

Böylelikle finansal verilerin yönetilmesinde yalnızca tek bir veritabanı yerine değişik amaçlarla oluşturulmuş 4 (dört) veritabanı kullanılmaktadır. Birinci veritabanı İMKB'de hisseleri işlem gören şirketlere ve bunlara ait hisse senetlerine ve dolayısıyla bu hisse senetlerini barındıran endekslere ilişkin genel bilgileri depolamaktadır. Diğer üç veritabanı ise hisse senetlerinin, İMKB endekslerinin ve yurtdışı endekslerin tarihi piyasa verilerini saklamaktadır. Söz konusu bu 4 (dört) veritabanı temsili olarak Şekil 3.3' te gösterilmiştir.



Şekil 3.3: Finansal Veri Tabanlarının Birbirleriyle İlişkisi

Söz konusu günlük fiyat bilgilerini tutan veritabanlarının oluşturulması için geliştirilen bu program, veritabanlarına yalnızca verilen tarih aralığı için ilgili fiyat bilgilerini yazmaktadır. Oysa, fonksiyonel olarak kullanılabilmeleri için bu veritabanlarının güncel olmaları gerekir. Bu amaçla geliştirilen ayrı bir program Ek-B.8'de verilmiştir. Bu program çalıştırıldığında veritabanları, MynetFinans ve Yahoo!Finance sitelerindeki fiyat bilgileriyle güncel hale gelmekte; böylelikle fiyat bilgisi eksikliği kalmamaktadır.

Veritabanları üzerindeki çalışmalar tamamlandıktan sonra, uygulamalar esnasında ihtiyaç duyulacak belli başlı teknik analiz göstergelerinin ve araçlarının hesaplanabilmesini sağlayan teknik analiz kütüphanesi oluşturulmuştur. Bu yazılım dördüncü bölümde değinilecek olan GP uygulamalarından sonuncusu için geliştirilmiş olup Common Lisp programlama dilinde kodlanmıştır.

Şirket bilgilerinin ve menkul kıymetlerin günlük piyasa verilerinin saklanabileceği bir veritabanı geliştirilmesi yardımıyla ortaya çıkabilecek her türlü veri gereksiniminin karşılanır hale geldiği düşünülmektedir. Bu bakımdan GP uygulamasının başarılı sonuçlar üretebilmesi için gerekli altyapı çalışmalarının en çok çaba gerektiren kısmı gerçekleştirilmiştir.

Uluslararası emtia ve değerli maden borsalarında oluşan fiyatlar ve uluslararası para piyasalarında oluşan çapraz kur oranları da ilginç uygulama fırsatları yaratabilme potansiyelleri bakımından veritabanına ileride eklenmesi düşünülen bileşenlerdir.

4. Uygulama ve Bulgular

4.1. Finansal Zaman Serilerinde GP Uygulamaları

Birinci bölümde GP'nın finansal zaman serilerinde kullanımına ilişkin uygulamaların üç temel kategoriye ayrıldıkları belirtilmişti. Bu çalışma kapsamında temel ilgi alanını bunlardan ikinci ve üçüncü kategorilere giren uygulama tipleri oluşturmaktadır. Bu amaçla GP'nın finansal zaman serilerinde kullanımına ilişkin üç farklı sistem geliştirilmiştir. Bunlardan ilk ikisi GP literatüründe *sembolik regresyon* olarak adlandırılan sınıfa girmektedir. Üçüncü uygulama ise tamamen farklı bir uygulama olup sermaye piyasalarında yatırımcıların yatırım kararlarını alırken sıklıkla baş vurduğu teknik analiz araçlarını kullanmaktadır. Üçüncü uygulama kapsamında geliştirilen sistem, al-sat kararları vermede kullanılabilecek bir karar destek sistemi geliştirme amacını gütmektedir. Bu sistem; belli başlı teknik analiz araçlarından bazılarını kullanarak kural ağaçlarından oluşan bir popülasyona sahiptir. Bu kural ağaçlarının verdiği al-sat sinyallerinin kullanımıyla gerçekleştirilen spekülasyon performansına göre bu karar ağaçlarını GP yardımıyla evriltmeye çalışmaktadır.

4.1.1. **Garanti Bankası A.Ş. hisse senedi için çok değişkenli sembolik regresyon uygulaması**

Bu uygulama kapsamında Garanti Bankası A.Ş. hisse senedinin (GARAN) kapanış fiyatını GARAN, U30 ve U100 endekslerinin bir gün önceki kapanış fiyatlarını kullanarak tahmin etmeye çalışan bir sembolik regresyon sistemi geliştirilmiştir. Bu model GARAN hisse senedinin fiyat hareketlerinin açıklamaya çalışan bir matematiksel model bulmaya çalışmaktadır.

Uygulama SBCL Common Lisp derleyicisinde 400 bireyden oluşan bir popülasyon için 200 nesil boyunca çalıştırılmıştır. Bu örnek çalıştırmada 27. nesilde elde edilen bir birey 449 isabet ve 14.432 standart uyum değeri ile en iyi birey olmuştur. Bu bireyin

sembolik ifadesi aşağıdaki gibidir:

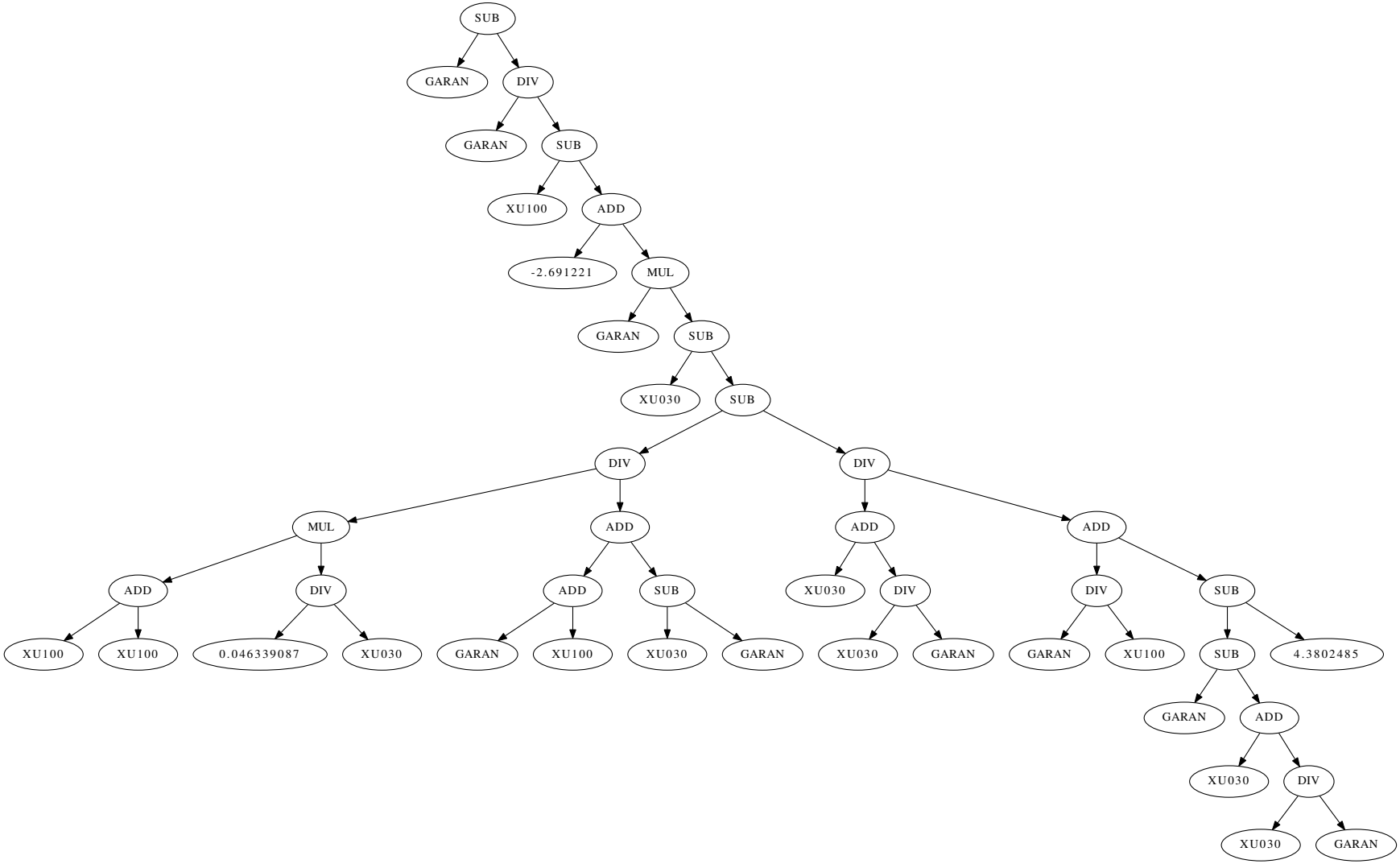
```
(- GARAN
   (% GARAN
      (- XU100
         (+ -2.691221
            (* GARAN
               (- XU030
                  (-
                   (% (* (+ XU100 XU100) (% 0.046339087 XU030))
                      (+ (+ GARAN XU100) (- XU030 GARAN)))
                   (% (+ XU030 (% XU030 GARAN))
                      (+ (% GARAN XU100)
                         (- (- GARAN (+ XU030 (% XU030 GARAN))))
                         )))))))))
```

Bu uygulamaya ilişkin olarak verilen GP tablosu Tablo 4.1 'de verilmiştir. Buna göre terminal kümesi GARAN, XU030 ve XU100'ün kapanış fiyatları ve bunun yanında $0 <= x <= 1$ olan ondalıklı bir rassal değişkendir. Fonksiyon kümesi ise toplama (+), çıkarma (-), çarpma (*) ve korumalı bölme (%) operatörlerinden oluşmaktadır. Burada değinilmesi gereken özel fonksiyon korumalı bölmedir. Korumalı bölme, GP literatüründe sıklıkla kullanılan bir fonksiyon olup paydası sıfır olan kesirli ifadelerle karşılaşıldığında programların doğru davranabilmesi için ortaya atılmış bir düşüncedir. Korumalı bölme, bir kesirli ifadenin paydasında 0 (sıfır) olduğunda, $0/0$ tanımsız durumu da dahil olmak üzere, 1 değerini veren aksi takdirde ise normal bölme işlemini uygulayan bir operatördür (Koza, 1992:83).

Tablo 4.1: Garanti Bankası Hisse Senedinin Fiyatını Tahminleyen Programlar Üreten Sembolik Regresyon Sistemine Ait GP Tablosu

Amaç:	Garanti Bankası A.Ş.'nin kapanış fiyatını bu hisse senedinin, U30 ve U100 endekslerinin bir gün önceki kapanış fiyatlarını kullanarak tahmin etmeye çalışan programın bulunması.
Terminal Kümesi:	`T={GARAN, XU030, XU100, Rassal ondalıklı sayı (0<=x<=1)}`
Fonksiyon Kümesi:	`F={+, -, *, % }`
Popülasyon Oluştuma Yöntemi:	`Ramped Half & Half`
Uyum Durumları:	Terminal kümesindeki menkul kıymetlerin 1 Ocak 2000'den itibaren 1000 işlem günündeki kapanış fiyatları.
Ham Uyum:	Tüm uyum durumları için hedef değer ile programın ürettiği değer arasındaki mutlak farkların toplamı (toplam mutlak hata).
Standartlaştırılmış Uyum:	Ham uyum ile aynı.
İsabet:	Program sonucu elde edilen tahmin ile hedef değer arasındaki mutlak farkın `0.01` 'den küçük olduğu durumlar.

Bulunan en iyi programın ağaç gösterimi Şekil 4.1'de verilmiştir.



Şekil 4.1: Garanti Bankası Hisse Senedi Fiyatını Tahminleyen Programlar Üreten Sembolik Regresyon Sisteminden Bir Çalıştırma Sonucunda Elde Edilen En İyi Bireyin Ağaç Gösterimi

Sistemin bulduğu en iyi programın matematiksel ifadesi sadeleştirildikten sonra Denklem 4.1'deki gibi ifade edilmektedir. Bu ifadede, daha sade bir gösterim amacıyla x, y ve z sırasıyla $GARAN$, $XU030$ ve $XU100$ yerine kullanılmıştır. Bu denklem; Garanti Bankası A.Ş. hisse senedinin bir gün sonraki tahmin edilen fiyatını, sözü edilen menkul kıymetlerin bir gün önceki kapanış fiyatlarından yararlanarak tahmin etmektedir.

$$x_{(t+1)} = \left(x_t - \frac{x_t}{-x_t \left(-\frac{0.092678174\, z_t}{y_t\,(z_t+y_t)} + \frac{\frac{y_t}{x_t}+y_t}{\frac{x_t}{z_t}-\frac{y_t}{x_t}-y_t+x_t} + y_t \right) + z_t + 2.691221} \right) \tag{4.1}$$

4.1.2. U30 endeksi için çok değişkenli sembolik regresyon uygulaması

Bu uygulamaya ilişkin olarak verilen GP tablosu Tablo 4.2'de verilmiş olup U30 endeksinin kapanış değerini belli başlı yerli ve uluslararası hisse senedi ve emtia endekslerine dayalı olarak tahminleyen programlar elde etme amacındadır. Terminal kümesinde ulusal endekslerden U30 ve U100 yer alırken; uluslararası hisse senedi endekslerinden Nikkei (Japonya), AORD (Avustralya), HSI (Hong Kong), BSE Sensex (Hindistan), DAX (Almanya), CAC40 (Fransa), FTSE (İngiltere), Dow Jones (ABD), Nasdaq (ABD), S&P500 (ABD) ve Bovespa (Brezilya) endeksleri yer almaktadır. Bunlara ek olarak XOI (AMEX petrol endeksi), XAU (PHLX altın/gümüş sektörü endeksi) ve VIX (S&P 500 volatilite endeksi) endeksleri yer almaktadır. Bu son üç endeksin modele konulmasının sebebi, emtia ve değerli maden piyasasındaki hareketlerin ve piyasalardaki psikolojik faktörlere ilişkin bilgi veren bu endekslerin, sistemin yaratıcı çözümler bulmasında işini kolaylaştırabileceği düşüncesidir. Fonksiyon kümesi birinci sistemdeki ile aynıdır.

Uygulamanın değişik seferler değişik parameterelerle çalıştırılması sonucunda, sistemin kayda değer çözümler üretemediği, uyum fonksiyonu ortalama değerlerinin nesiller ilerledikçe herhangi bir yakınsama göstermediği saptanmıştır.

4.1.3. U30 endeksi için al-sat kararları veren kural ağacı uygulaması

Teknik analizin; sermaye/hisse senedi, mal/emtia ve opsiyon piyasalarında karar almada en sık kullanılan araçlardan biri olduğu bilinmektedir (NYIF, 1989:3). Tarihçesinin 17. ve 18. yüzyıl Avrupalı tüccarlara ve Japon pirinç tüccarlarına değin uzandığı belirtilen tek-

Tablo 4.2: U30 Endeksi Kapanış Seviyesini Dünya Endekslerine Bağlı Olarak Tahminleyen Programlar Üreten Sembolik Regresyon Sistemine Ait GP Tablosu

Amaç:	Ulusal 30 endeksinin kapanış değerini ulusal ve uluslararası endekslerin belli başlılarından toplam 16 endeksin bir gün önceki kapanış fiyatlarını kullanarak tahmin etmeye çalışan programın bulunması.
Terminal Kümesi:	`T={XU030 ,XU100 ,N225 ,AORD ,HSI ,BSESN ,GDAXI ,FCHI ,FTSE ,DJI ,IXIC ,GSPC ,BVSP ,XOI ,XAU ,VIX, Rassal ondalıklı sayı (0<=x<=1) }`
Fonksiyon Kümesi:	`F={+, -, *, %}`
Popülasyon Oluştuma Yöntemi:	`Ramped Half & Half`
Uyum Durumları:	Terminal kümesindeki endekslerin 1 Ocak 2000' den itibaren 2000 işlem günündeki kapanış fiyatları.
Ham Uyum:	Tüm uyum durumları için hedef değer ile programın ürettiği değer arasındaki mutlak farkların toplamı. (toplam mutlak hata)
Standartlaştırılmış Uyum:	Ham uyum ile aynı.
İsabet:	Programın sonucu elde edilen tahmin ile hedef değer arasındaki mutlak farkın, hedef değerin % `0.25` 'inden küçük olduğu durumlar.

nik analizin bugünkü şeklini almasında en büyük katkı kuşkusuz ki Dow Jones'a aittir[1]. Dow Jones'un yayınlanmış bir teorisi bulunmasa dahi, hisse senetlerinden seçtiği bazı sepetlerin "ortalamaları" üzerinde yaptığı çalışmalardan elde ettiği sonuçları takipçilerinin belirli bir temele oturtması sayesinde günümüzde Dow Teorisi'nden söz edilmektedir (NYIF, 1989:4). Dow teorisinin ve teknik analizin dayandığı üç temel varsayım şu şekilde özetlenmektedir:

- Piyasa herşeyi fiyatlar.
- Fiyatlar trend halinde hareket eder.
- Tarih tekerrür eder.

Bu varsayımlar altında geliştirilmiş teknik analiz göstergeleri genellikle dört sınıf altında toplanmaktadır. Bunlar; *trende, momentuma (fiyat değişim hızına), işlem hacmine* ve *oynaklığa (volatiliteye)* yönelik olarak geliştirilen göstergeler olarak ayrışmaktadır (Colby, 2003:7-8).

Bu bağlamda, teknik analiz geçmiş fiyatlardan gelecek fiyatların yönünü *kestirmede* kullanılan araçların bütünüdür. Söz konusu kestirim, istatistiksel bir kestirimden daha çok, bir uzmanlık bilgisi ışığında alınacak olan yatırım kararlarını detekleyici niteliktedir. Üçüncü ve son modelde de bu bilgi göz önünde bulundurularak, doğrudan bir tahmin yapmak yerine bazı teknik analiz göstergelerinin güncel değerlerinin belirli değerlerle kıyaslandığı mantıksal ifadelerden oluşan bireyler bulunmaktadır. Bu bireylerin her biri piyasada oluşan günlük verileri ve bu verilerden hesaplanan teknik analiz göstergelerini girdi olarak almakta olup çıktı olarak ise piyasada nasıl bir pozisyon alınması gerektiğini belirtmektedir. Burada; her birey bir kural ağacı olup, bu kurallara göre al-sat yapıldığında elde edilen kazanç; bireylerin uyumunu ölçmede temel alınmıştır. Bu modelin merak uyandırıcı ve güdüleyici özelliklerinden biri de sistemin çalıştırılması sonucu bulunacak olan en iyi bireyin, bir diğer ifadeyle al-sat sinyalleri veren bu kural ağacının, gerçek alan uzmanlarıyla (teknik analistlerle) kıyaslandığında ne derecede etkin kararlar verebildiğinin kıyaslanabilmesidir.

[1] Teknik analiz tarihçesi için bkz.:
`http://en.wikipedia.org/wiki/Technical_analysis#History`
(Erişim tarihi: 09.10.2011)

Uygulama kapsamında yararlanılan teknik analiz göstergelerinin belirlenmesinde dikkat edilmesi gereken bazı hususlar bulunmaktadır. Bunlar; mevcut veri altyapısının yeterliliği, teknik analiz uzmanlarınca benimsenmiş olup olmadığı ve bu göstergelerin alan uzmanlarınca nasıl yorumlandığı, başka bir ifadeyle menkul kıymetin hangi niteliklerini izlemede kullanıldığıdır. Uygulamada kullanılan teknik analiz göstergelerinin tercih edilmesinin sebepleri ve bu göstergelerin nasıl hesaplandığı aşağıda aktarılmıştır.

4.1.3.1. ***MACD (moving average convergence/divergence)***

MACD göstergesi, Gerald Appel tarafından *"Systems and Forecasts"* adlı eserinde ortaya atılmış olup günümüzde teknik analistlerce en çok kullanılan göstergelerden biridir. MACD, temelde bir fiyat momentumu osilatörü olup hesaplanması üç aşamada gerçekleştirilir (Colby, 2003:412). Birinci aşamada, kapanış fiyatının 26 günlük (yavaş) üssel hareketli ortalamasının, 12 günlük (hızlı) üssel hareketli ortalamasından çıkarılması yardımıyla *fiyat hızı (price velocity)* tabir edilen fark elde edilir. İkinci aşamada, birinci aşamada hesaplanan serinin yine genel kabul görmüş şekliyle 9 günlük üssel hareketli ortalaması hesaplanır. Bu hesaplamada elde edilen osilatöre *sinyal veya tetik çizgisi (trigger line)* denmektedir. Üçüncü aşamada ise, fiyat hızından sinyal çizgisini çıkarmak yardımıyla hesaplanan *fiyat ivmesi (price acceleration)* hesaplanır.

$$PV = EMA(C,12) - EMA(C,26) \tag{4.2}$$

$$PA = PV - EMA(PV,9) \tag{4.3}$$

Genellikle MACD hesaplamasında kullanılan vadeler analistlerce genel kabul görmüş ve standart olarak benimsenmiş vadeler olmakla birlikte, farklı parametrelerle de bu hesaplamalar gerçekleştirilebilir. Bu göstergenin yorumlanması; genellikle fiyat hızının tetik çizgisini geçince al altında kalınca da sat sinyali vermesi biçimindedir. Aynı anlama gelen başka bir yorumlama biçimi de fiyat ivmesinin histogramının sıfırın üstüne çıkınca al, altına inince ise sat sinyali şeklinde yorumlanmasıdır.

4.1.3.2. ***RSI (relative strength index)***

RSI, fiyat momentumu göstergelerinin en popülerlerinden biridir. J. Welles Wilder tarafından 1978 tarihli *"New Concepts in Technical Trading Systems"* adlı kitabında tarif edilmiştir (Colby, 2003:610-617). Matematiksel olarak RSI şu şekilde hesaplanmaktadır:

$$RSI = 100 - \frac{100}{1+RS} \tag{4.4}$$

Bu formül uyarınca *RS* değeri, kazançların *n* periyotluk üssel hareketli ortalamasının, kayıpların *n* periyotluk üssel hareketli ortalamasının mutlak değerine bölünmesiyle elde edilir. Genellikle RS'nin hesaplanmasında kullanılan formül şu şekilde gösterilmektedir:

$$RS = \frac{EMA(U,n)}{EMA(D,n)} \tag{4.5}$$

Burada U menkul kıymetin fiyatının bir önceki işlem gününe göre arttığı D ise azaldığı günlere göre hesaplanan seriler olup n ise periyot uzunluğudur. Wilder'in önerdiği periyot uzunluğu (günlük veri için) 14 gün olup değişik periyot uzunlukları da uzmanlarca kullanılmaktadır. İlgili serilerin hesaplanması ise aşağıdaki şekilde gerçekleştirilmektedir:

$$U_t = \begin{cases} C_t - C_{t-1} & \text{, eğer } \ C_t > C_{t-1} \text{ ise} \\ 0 & \text{, eğer } \ C_t \leq C_{t-1} \text{ ise} \end{cases} \tag{4.6}$$

$$D_t = \begin{cases} 0 & \text{, eğer } \ C_t > C_{t-1} \text{ ise} \\ C_{t-1} - C_t & \text{, eğer } \ C_t \leq C_{t-1} \text{ ise} \end{cases} \tag{4.7}$$

4.1.3.3. ***Bollinger %b***

John A. Bollinger tarafından ortaya atılan *"Bollinger Bands" (Bollinger Kanalı)* göstergesine dayalı olarak hesaplanan ayrı bir gösterge olup menkul kıymet fiyatının, *kanal hareketi* içerisinde aşırı alım veya satım bölgelerinde olup olmadığının, bir başka ifadeyle fiyat kanalının alt veya üst sınırına yakın veya uzak olup olmadığının belirlenmesinde kullanılmaktadır (Colby, 2003:114-120).

Bollinger kanalının alt (LBB), orta (MBB) ve üst (UBB) olmak üzere üç çizgiden oluşmaktadır. Orta çizgi (MBB) genellikle 20 günlük basit hareketli ortalama iken alt ve

üst bandın belirlenmesinde 20 günlük anakütle standart sapması ve k gibi bir parametre kullanılır.

$$MBB = MA(C,20) \tag{4.8}$$

$$UBB = MBB + k\sigma \tag{4.9}$$

$$LBB = MBB - k\sigma \tag{4.10}$$

Bollinger kanalı hesaplamalarında genel olarak $k = 2$ şeklinde kabul görmektedir. Bollinger kanalından türetilen $\%b$ osilatörü ise mevcut fiyat seviyesinin Bollinger bandının neresinde seyrettiğini belirlemede yardımcı olmaktadır.

4.1.3.4. ***William's %R***

Larry Williams tarafından geliştirilen bir osilatördür (Colby, 2003:795). Uygulamaya dahil edilmesinin sebebi "Stochastics" göstergesi gibi bağıl olarak hesaplanan bir eşik değeri yanı sıra mutlak br sayısal değerden oluşan bir eşik değere sahip bir osilatörün de terminal kümesi içerisinde bulunmasının yararlı olabileceği düşüncesidir. William's %R şu şekilde hesaplanmaktadır:

$$\%R_t = \frac{H_{n^*} - C_t}{H_{n^*} - L_{n^*}} x100 \tag{4.11}$$

Bu formül uyarınca t zamanında William's %R osilatörünün değerinin hesaplanmasında menkul kıymetin son n periyot boyunca aldığı en yüksek ve en düşük değerler olan H_{n^*} ve L_{n^*} ile t zamanındaki kapanış fiyatı olan C_t'den yararlanılır.

4.1.3.5. ***Stochastics***

"Stochastics" osilatörü George Lane tarafından geliştirilip popüler hale getirilmiş bir teknik analiz göstergesidir ve temelde Williams %R ile aynıdır (Colby, 2003:664). Ancak ayırt edici özelliği Williams %R 'nin matematiksel olarak tam tersi şeklinde hesaplanan bir %K değerine sahip olması ve bu değere bağlı olarak hesaplanan iki ayrı bağıl tetik çizgisi hesaplanmasıdır.

Böylelikle; uygulamada kullanılmak üzere seçilmiş teknik analiz göstergeleri daha önce değinilen sınıflandırma anlamında işlem hacmi hariç olmak üzere farklı sınıflardan

gelen bir çeşitlilik sağlamıştır. Öyle ki; MACD göstergesi trend sınıfını temsil etmekteyken RSI, Stochastics ve Williams %R göstergeleri momentum sınıfını temsil etmektedir. Bollinger kanalından türetilen % b göstergesi ise oynaklık sınıfını temsilen terminal kümesinde yer almaktadır. İşlem hacmine dayalı olarak geliştirilen bir göstergeye bu uygulamada yer verilememesinin sebebi, üçüncü bölümde değinildiği gibi ikincil veri kaynaklarından sağlanan hacim verisinin yeterince kaliteli olmamasıdır. Yukarıda değinilen teknik analiz göstergelerinin hesaplanabilmesi için Common Lisp programlama dilinde geliştirilen kütüphane Ek-C.3'te verilmiştir.

Uygulamaya ilişkin GP tablosu Tablo 4.3'te verilmiş olup sistemin terminal kümesi teknik analistlerce sıklıkla kullanılan *MACD (Moving Average Convergence/Divergence), Stochastics, Bollinger %b, RSI (Relative Strength Index) ve Williams %R* göstergelerine dayanan terminallerden oluşmaktadır. Bu terminaller teknik olarak göstergelerin hesaplanan değerleri ile bunların karşılaştırıldıkları bir eşik değerini tutan veri yapısı nesneleridir.

Bu uygulama için geliştirilen fonksiyon kümesinde `MOR<=`, `MOR>`, `MAND<=` ve `MAND>` fonksiyonları yer almaktadır. Bu fonksiyonlar bu uygulama bağlamında özgün olarak geliştirilmiş olup fonksiyon tanımları Ek-C.4'te verilmiştir. Özetle; bu fonksiyonlar teknik analiz göstergelerinin ilgili gündeki değerleri ile söz konusu indikatör için tanımlanmış eşik değerini kıyaslamakta ve Boolean bir değer döndürmektedir. Hatırlatmak gerekir ki; bu sistemde her bir birey bu fonksiyonların değişik kombinasyonlarından oluşan ve Boolean bir değer döndüren bir programdır.

Söz konusu sistem değişik parametre kombinasyonlarıyla deneme çalıştırmalarından sonra populasyon büyüklüğü 100000 olarak 50 nesil boyunca CCL Common Lisp derleyicisi üzerinde çalıştırılmıştır. Sistemin 35. nesilde belirlediği en iyi birey bu çalıştırmanın en iyi bireyi olarak bulunmuş olup standart (normalleştirilmiş) uyum değeri 0.00445672'tir. Bu bireyin sembolik ifadesi aşağıda verilmiştir:

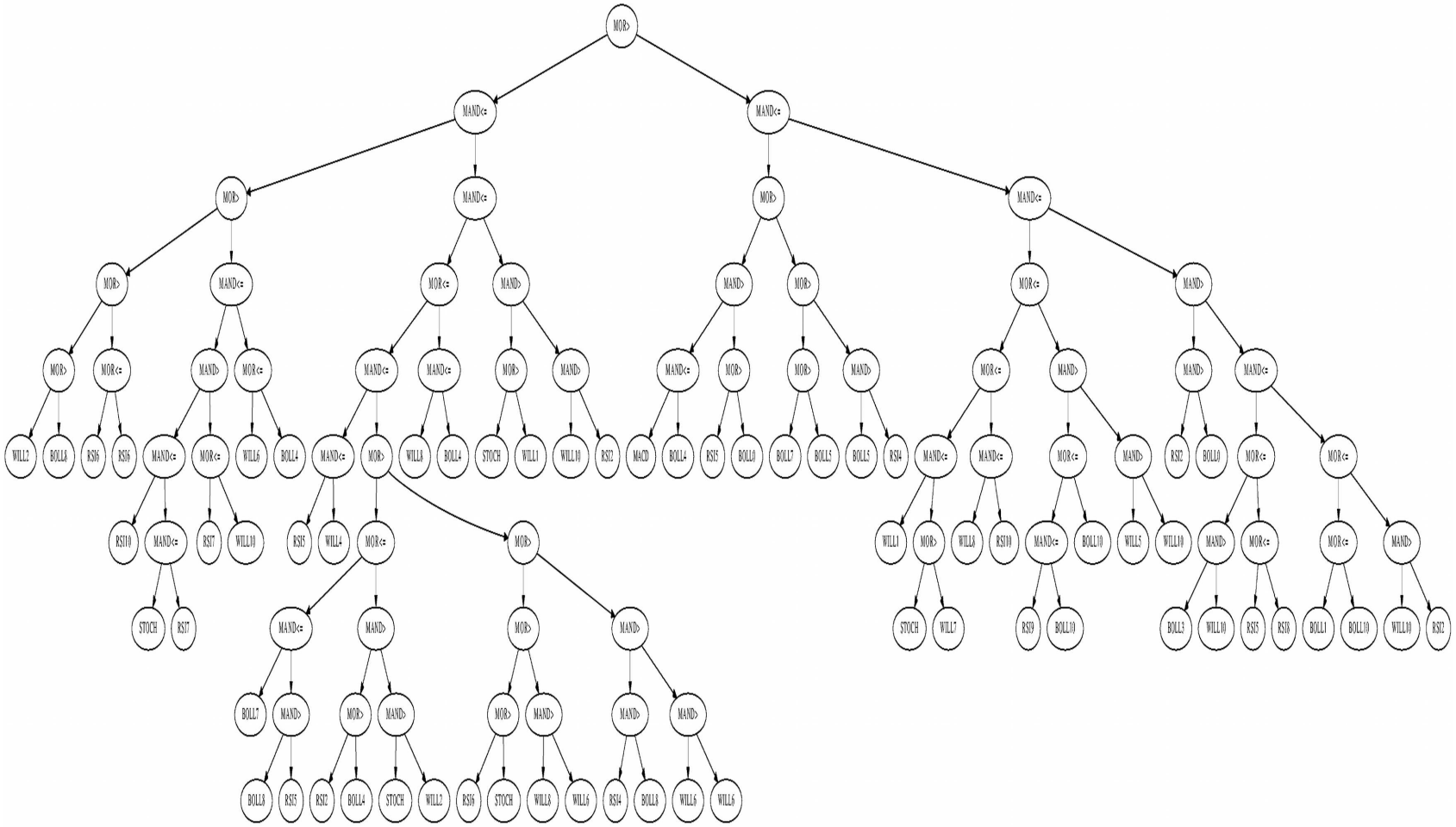
Tablo 4.3: Ulusal 30 Endeksi İçin Al-Sat Kararları Veren Kural Ağacı Üreten Sisteme Ait GP Tablosu

Amaç:	U30 endeksi için günlük çerçevede hesaplanmış bazı teknik analiz araçları yardımıyla en iyi alsat kararlarının veren kural ağacını oluşturan programı bulmak.
Terminal Kümesi:	T={MACD, STOCH, BOLL0 ,BOLL1 ,BOLL2 ,BOLL3 ,BOLL4 ,BOLL5 ,BOLL6 ,BOLL7 ,BOLL8 ,BOLL9 ,BOLL10, RSI0 ,RSI1 ,RSI2 ,RSI3 ,RSI4 ,RSI5 ,RSI6 ,RSI7 ,RSI8 ,RSI9 ,RSI10, WILL0 ,WILL1 ,WILL2 ,WILL3 ,WILL4 ,WILL5 ,WILL6 ,WILL7 ,WILL8 ,WILL9 ,WILL10}
Fonksiyon Kümesi:	F={MOR<=, MOR>, MAND<=, MAND>}
Popülasyon Oluşturma Yöntemi:	Ramped Half & Half
Uyum Durumları:	Terminal kümesindeki U30 için hesaplanmış teknik analiz gösterge ve osilatörlerinin 1 Ocak 2000'den 1 Ocak 2011 tarihine kadar 2720 işlem günündeki değerleri.
Ham Uyum:	Uyum durumları için bir bireyin ürettiği sinyal dizisi gereğince işlem yapılması durumunda, yatırımcının ilk sermayesinin öğrenme dönemi sonunda ulaştığı portföy büyüklüğüne oranı.
Standartlaştırılmış Uyum:	Ham uyum ile aynı.
İsabet:	-

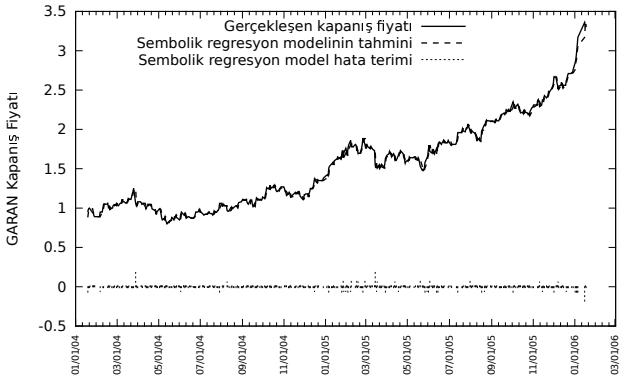
```
(MOR> (MAND<= (MOR> (MOR> (MOR> WILL2 BOLL8) (MOR<= RSI6 RSI6))
                    (MAND<= (MAND> (MAND<= RSI10 (MAND<= STOCH RSI7))
                                   (MOR<= RSI7 WILL10))
                            (MOR<= WILL6 BOLL4)))
              (MAND<= (MOR<= (MAND<=
                              (MAND<= RSI5 WILL4)
                              (MOR>
                               (MOR<=
                                (MAND<= BOLL7 (MAND> BOLL8 RSI5))
                                (MAND>
                                 (MOR> RSI2 BOLL4)
                                 (MAND> STOCH WILL2)))
                               (MOR>
                                (MOR>
                                 (MOR> RSI6 STOCH)
                                 (MAND> WILL8 WILL6))
                                (MAND>
                                 (MAND> RSI4 BOLL8)
                                 (MAND> WILL6 WILL6)))))
                            (MAND<= WILL8 BOLL4))
                      (MAND> (MOR> STOCH WILL1)
                             (MAND> WILL10 RSI2))))
      (MAND<= (MOR> (MAND> (MAND<= MACD BOLL4) (MOR> RSI5 BOLL0))
                    (MOR> (MOR> BOLL7 BOLL5) (MAND> BOLL5 RSI4)))
              (MAND<= (MOR<= (MOR<=
                              (MAND<= WILL1 (MOR> STOCH WILL7))
                              (MAND<= WILL8 RSI10))
                             (MAND>
                              (MOR<= (MAND<= RSI9 BOLL10) BOLL10)
                              (MAND> WILL5 WILL10)))
                      (MAND> (MAND> RSI2 BOLL0)
```

```
(MAND<=
 (MOR<=
  (MAND> BOLL3 WILL10)
  (MOR<= RSI5 RSI8))
 (MOR<=
  (MOR<= BOLL1 BOLL10)
  (MAND> WILL10 RSI2)))))))
```

Bu sistemin ağaç şeklinde gösterimi Şekil 4.2'de verilmiştir.



Şekil 4.2: Ulusal 30 Endeksi İçin Al-Sat Kararları Veren Kural Ağacı Üreten Sistemden Bir Çalıştırma Sonucunda Elde Edilen En İyi Programının Ağaç Gösterimi



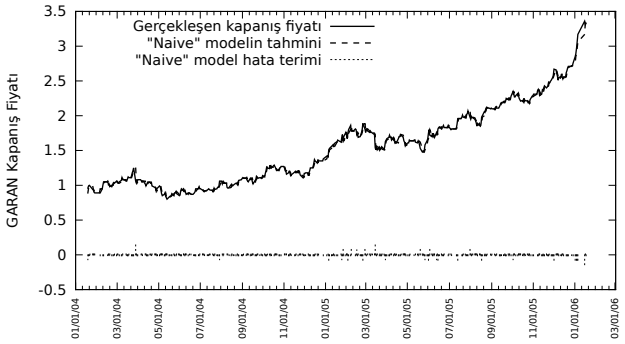
Şekil 4.3: Garanti Bankası A.Ş. Hisse Senedi İçin Bulunan En İyi Programın Ürettiği Tahmini Kapanış Fiyatlarının Gerçekleşen Kapanış Fiyatlarıyla Karşılaştırılması

4.1.4. **Sonuç ve bulgular**

Gerçekleştirilen bu üç uygulama sonucunda varılan yargıları kısaca özetlemek gerekirse birinci ve üçüncü uygulamadan alınan sonuçlar başarılı ikinci uygulamadan alınan sonuç ise başarısızdır. İkinci modelin parametreleri üzerinde bazı ayarlamaların yapılması gerekliliği ortadadır. Bu ayarlamalar; fonksiyon kümesine eklenebilecek yeni fonksiyonlar olabileceği gibi, gereksiz terminallerin tümden çıkartılarak daha sade bir model elde edilmesi şeklinde de olabilir. Yine başka bir önlem olarak, terminal kümesindeki değişkenler üzerinde kümeleme veya toplulaştırma analizleri gerçekleştirildikten sonra GP modeli daha yalın hale getirilebilir. Birinci ve üçüncü modelde ise araştırmaya başlarken öngörülen sonuçların elde edildiği görülmüştür.

Birinci uygulamada sistemin eğitim sürecinin bitişini takip eden 500 gün için elde edilen tahminlerle gerçek fiyatların kıyaslandığı grafik Şekil 4.3'te verilmiştir.

Görülmektedir ki; sistemin ürettiği tahmin modelinin çıktıları gerçekleşen değerlerle neredeyse örtüşmektedir. Sembolik regresyon modelinin eğitim sürecindeki uyum durumlarından % 45'inde de tam isabetli tahminler yaptığı göz önüne alındığında GP yardımıyla ne kadar güçlü tahmin modelleri üretilebileceği daha iyi anlaşılacaktır. Ancak; bu bulgunun sistemin temelde AR(1) tabir edilen zaman serisi bileşenini yakaladığının ötesinde bir bilgi taşımadığı da bilinmelidir. Öyle ki; aynı dönem kapsamında *naïve* tahmin edici $\hat{Y}_t = Y_{t-1} + \varepsilon$ ile gerçekleşen değerleri bir arada gösteren grafik Şekil 4.4 'te gösterilmiştir. Bu grafik sembolik regresyon yardımıyla bulunan tahmin edici modelin temelde bir



Şekil 4.4: Garanti Bankası A.Ş. Hisse Senedi İçin Naïve Tahminlerin Gerçek Kapanış Fiyatlarıyla Karşılaştırılması

AR(1) süreci ortaya çıkardığını göstermektedir. Bu amaçla hem bu modelin, hem de *naïve* modelin hata terimleri karelerinin toplamının karekökü hesaplanmış olup bu değer sembolik regresyon yardımıyla elde edilen tahmin modeli için 0.8073 *naïve* model için ise 0.808 olarak bulunmuştur. Bu bulgu, daha önce ifade edilen ve grafiklerle de desteklenen görüşleri güçlendirmektedir.

Tablo 4.4'te; kural ağacı sisteminin bulduğu en iyi bireyin ürettiği al-sat sinyalleri, uygulamanın çalıştırıldığı tarihten önceki son 3 ayı kapsayacak şekilde verilmiştir. Bu tabloda; bu kural ağacının ilgili t tarihindeki veriler girdiyken ürettiği sinyaller ς_t ile gösterilmektedir. Bu sinyaller; T piyasada olma (elinde menkul kıymet bulundurma) ve F ise elindeki tüm menkul kıymetleri açığa satıp piyasa dışında bekleme anlamına gelen Boolean değerlerdir. Görülmektedir ki; bu model öğrenme ve test periyotlarının ayrılmadığı böyle bir deneme çalıştırmasında son derece doğru al-sat kararları verebilmektedir. Ancak bu yaklaşım; bir başka ifadeyle test periyodunun ayrı tutulmaması, bilimsel anlamda zayıf bir yaklaşımdır. Bu sebepten ilerleyen bölümlerde öğrenme ve test periyotları ayrılarak belli bir deney tasarımı altında uygulama yinelenecektir.

Tablo 4.4: Bulunan En İyi Programın Ürettiği Al-Sat Sinyalleri

t	$U30$	ς_t	t	$U30$	ς_t	t	$U30$	ς_t
20110630	76930.43	T	20110802	74766.85	F	20110907	66939.22	T
20110701	76921.96	T	20110803	74488.48	F	20110908	68982.0	F
20110704	78310.77	T	20110804	72016.57	F	20110909	67967.1	T
20110705	77867.88	T	20110805	68160.56	F	20110912	67692.75	T
20110706	76972.11	T	20110808	63476.95	F	20110913	68997.8	T
20110707	78141.15	T	20110809	64539.48	F	20110914	68232.83	T
20110708	77409.66	T	20110810	60685.9	F	20110915	69503.96	T
20110711	76513.4	T	20110811	62968.22	T	20110916	70421.68	T
20110712	76253.78	T	20110812	62960.07	T	20110919	70470.91	T
20110713	76842.1	T	20110815	64542.44	T	20110920	74646.79	T
20110714	76189.12	F	20110816	65498.63	T	20110921	74383.47	T
20110715	76061.75	F	20110817	65578.86	T	20110922	70562.16	T
20110718	75234.62	T	20110818	62702.18	F	20110923	68582.45	F
20110719	75035.15	T	20110819	64430.58	T	20110926	69031.39	F
20110720	74763.51	T	20110822	64801.73	T	20110927	71042.27	T
20110721	74024.32	T	20110823	63422.45	F	20110928	72045.19	T
20110722	72601.0	F	20110824	64411.17	T	20110929	73347.33	T
20110725	74254.39	T	20110825	64227.58	T	20110930	73498.05	T
20110726	74550.13	T	20110826	65328.01	F	20111003	73258.82	T
20110727	74218.41	T	20110829	65600.19	F	20111004	70235.26	F
20110728	76022.32	T	20110902	67364.31	T	20111005	70243.12	F
20110729	75614.55	F	20110905	65624.54	T	20111006	69925.98	F
20110801	74962.13	F	20110906	66875.09	T	20111007	70125.5	F

4.1.5. **Karşılaşılan güçlükler ve eksiklikler**

Uygulamalarda karşılaşılan en büyük güçlük üçüncü modeldeki gibi karmaşık bir sistemin uyum fonksiyonunun hesaplanmasında ortaya çıkan hesaplama kaynağı gereksinimidir. Özellikle de bu örnekteki gibi çok geniş bir arama uzayı söz konusu iken, popülasyon hacminin oldukça büyük tutulması gerekmektedir. Bu zorunluk da daha uzun süre çalış-

ması gereken programlara yol açmaktadır.

Ayrıca GP çok iyi uyun fonksiyonu değerlerine sahip çözümler bulsa bile, bu çözümlerin öğrenme evresi dışındaki verilerdeki performanslarının istatistiksel olarak test edilmesi gerekmektedir. Bulunan sonuçların rastgele mi yoksa bilinçli olarak mı bu sistemlerce üretildiği ancak bu şekilde ortaya konabilir. Bunun için; bu sistemleri literatürde kabul görmüş minimum sayılarda çalıştırıp elde edilen çözümlerin daha önce karşılaşmadığı veriler üzerinde sınanması zorunludur. Ancak, önceki paragrafta belirtilen durum, üçüncü model gibi bir modelin onlarca kez çalıştırılması durumunda katlanılacak olan hesaplama külfetinin çok büyük boyutlarda olacağını göstermektedir.

4.2. Deneyler

Etkin Piyasa Hipotezi (Efficient Market Hypothesis - EMH) modern finans teorisinin dayandığı temellerden biridir. Fama; *"Efficient Capital Markets: A Review of Theory and Empirical Work"* adlı öncü makalesinde piyasa etkinliğinin üç formundan bahsedilmektedir. EMH'nin zayıf formu olarak adlandırılan durumda; bir yatırımcının geçmiş fiyatları kullanarak piyasaya üstün gelme şansının olmadığı, bütün bilgilerin piyasa tarafından fiyatlanmış olduğu, teknik olarak fiyatların stokastik sürecinin Markov süreci olduğu söylenmektedir (Fama, 1970). Bu hipotezin geçerliliği ile ilgili çok sayıda çalışma yapılmış olup bunlardan kimileri de teknik analiz araçları yardımıyla piyasanın üstünde getiri elde etme imkanının olduğunu savunmaktadır.

Teknik analizin, bilimsel olarak incelenmeye değer görülmesi ise tartışmalı bir konudur. Buna rağmen, gerçek hayatta yatırım profesyonellerinin büyük çoğunluğunun baş vurduğu bir araç olduğu tartışma götürmez bir gerçektir. Taylor ve Allen, görüştükleri Londra borsasındaki yabancı para birimine dayalı işlem yapan döviz tüccarların (foreign exchange -FX- dealer) ve al-satçıların (trader) % 90'ından fazlasının bir çeşit teknik analiz aracını kullandıklarını belirlemişlerdir (H. Allen ve Taylor, 1990; Taylor ve Allen, 1992). Benzer bir çalışmada Lui ve Mole, 1995 yılı itibariyle Hong Kong'daki FX tüccarlarıyla yaptıkları anket çalışması sonucunda, kısa zaman periyotlarında karar almada, teknik analizin temel analizden önemli ölçüde daha yaygın olduğunu ortaya koymuşlardır (Lui ve Mole, 1998).

Teknik analizin yatırımcılar arasında yaygınlığının yanı sıra, teknik analizi bilimsel

olarak inceleyen erken dönem çalışmalardan bazıları, basit teknik analiz göstergelerinin kimilerini kullanarak (Brock; Lakonishok ve LeBaron, 1992), teknik analizin uzun periyotlarda karlı olabileceğini göstermişlerdir (Levich ve Thomas, 1993).

Piyasa oyuncularının pek çoğu, teknik analizi karar almada mümkün olduğunca sistematik olarak kullanmaya çalışmakta iken, bunların bir kısmı ise al-sat kararlarını otomatikleştiren sistemler oluşturmada temel olarak kullanmaktadır (M. A. H. Dempster ve Jones, 2001). GP'nin bu tür sistemlerde kullanılması ile ilgili öncü çalışmalar, belirli koşullarda piyasa getirilerinin üzerinde getiriler sağlayan kurallar ortaya çıkaran sistemler geliştirebildiklerini rapor etmişlerdir (C. Neely vd., 1997; F. Allen ve Karjalainen, 1999).

Benzer şekilde bu çalışmada yapılan da, GP yardımıyla herhangi bir ön bilgi olmaksızın yalnızca eğitim dönemi verileri kullanılarak ortaya çıkarılan kural ağaçlarının; eğitim sürecinde kullanılmayan test dönemi verileri üzerinde verdikleri al-sat sinyalleri uyarınca yapılan işlemlerde elde edilen getirinin her seferinde al-bekle stratejisinden istatistiksel olarak daha fazla olup olmadığının ortaya konulmasıdır.

Bu çalışmada geliştirilen sistem 5 test döneminin her biri için 50 defa çalıştırılmış, sistemin bulduğu kural ağaçlarının ürettiği sinyallere göre yapılan alım-satımlar sonucunda elde edilen getirinin, al-bekle stratejisinden üstünlüğü istatistiksel olarak test edilmiştir.

4.2.1. **Matematiksel model**

Kural ağaçlarının değerlendirilmesi basit bir iş gibi görünse de, bu amaçla bir uyum fonksiyonunun tanımlanıp gerçekleştirilmesi problemin doğasından kaynaklanan sebeplerle zorlaşabilmektedir. Uyum fonksiyonu değerinin kabaca iki aşamada hesaplandığı söylenebilir. Birinci aşama, uyum değeri hesaplanacak kural ağacının bütün bir eğitim süreci boyunca menkul kıymetin (hisse senedi veya endeks) günlük fiyat bilgilerine göre ürettiği al-sat sinyallerinin hesaplanması; ikinci aşama ise bu sinyallerden hareketle yapılacak olan işlemlerden elde edilecek getirinin hesaplanmasıdır.

Sistemdeki kural ağaçları aslında çıktı olarak yalnızca `DOĞRU/YANLIŞ (T/F)` değerlerini veren Boolean fonksiyonlar olup girdi olarak menkul kıymetin ilgili günkü fiyat değerlerine ihtiyaç duymaktadırlar. Böylelikle herhangi bir kural ağacı öğrenme dönemindeki her gün için bir T/F değeri oluşturur.

$$\varsigma_t = f(p_t) \in \{T, F\} \tag{4.12}$$

Bu T/F değerleri, gösterim kolaylığı açısından doğal olarak sırasıyla 1 ve 0 ile ilişkilendirilebilir. Bu değerler burada *ham (raw signals) sinyaller* olarak anılacaktır. Ham sinyaller, t zamanı (işlem gününü) göstermek üzere $t = 1, 2, \ldots T$ için şu şekildedir:

$$\rho_t \in \{0, 1\} \tag{4.13}$$

Ham sinyallerden elde edilen *dönüştürülmüş sinyaller (transformed signals)* ise sistemin hem sinyalin verildiği zamanındaki (mevcut) durumu hem de bir önceki duruma göre olan durumunu belirtir. Dönüştürülmüş sinyaller $t > 1$ için şu şekilde gösterilebilir:

$$\tau_t = f(\rho_t, \rho_{t-1}) = \rho_t - \rho_{t-1} \tag{4.14}$$

Böylece dönüştürülmüş sinyallerin alabileceği değerler

$$\tau_t \in \{-1, 0, 1\} \tag{4.15}$$

kümesinde tanımlıdır. Dönüştürülmüş sinyal, $t = 1$ için ham sinyalin değerine eşittir. Bu notasyona ihtiyaç duyulmasının sebebi, sistemin devamlı olarak bir önceki zamandaki durumuyla karşılaştırılmasının getirdiği gösterim zorluğunun yarattığı kısıtlamadan kurtulmaktır.

Dönüştürülmüş sinyaller; sistemin piyasanın mevcut durumuna karşılık hangi eylemi gerçekleştireceğini, *yalnızca ilgili günkü sinyale dayanarak* gerçekleştirebilmesi sağlamaktadır. Bu ifadenin daha iyi anlaşılabilmesi için Tablo 4.5'de 13 gün için verilen hipotetik sinyaller ve bu sinyallere karşın sistemin gerçekleştirmesi gereken eylemlerin verildiği Tablo 4.6 incelenebilir.

Sistemde kural ağaçlarının verdiği sinyallere göre al-sat işlemlerini yapan basit bir portföy modelinin tasarlanması gereklidir. Bu portföy modeline ilişkin varsayımlar şu şekilde özetlenebilir:

- İşlem maliyetleri yoktur.
- Sinyalin gerektirdiği işlem kesinlikle gerçekleştirilmektedir.

Tablo 4.5: Sinyallerin Hipotetik Bir Örnek Üzerinde Gösterimi

t	1	2	3	4	5	6	7	8	9	10	11	12	13
ς_t	T	T	T	F	F	T	F	F	F	F	T	T	T
ρ_t	1	1	1	0	0	1	0	0	0	0	1	1	1
τ_t	1	0	0	-1	0	1	-1	0	0	0	1	0	0

Tablo 4.6: Tablo 4.5'de Verilen Hipotetik Sinyaller Karşısında Sistemin Davranışı

Gün	**Eylem**
1	Al
2	Mevcut pozisyonu koru
3	Mevcut pozisyonu koru
4	Sat + Açığa sat
5	Mevcut pozisyonu koru
6	Açık pozisyonu kapat
7	Sat + Açığa sat
8	Mevcut pozisyonu koru
9	Mevcut pozisyonu koru
10	Mevcut pozisyonu koru
11	Açık pozisyonu kapat
12	Mevcut pozisyonu koru
13	Mevcut pozisyonu koru

- Bütün işlemlerde sinyalin verildiği günkü kapanış fiyatı esas alınmaktadır.
- Portföyde yanlızca ilgilenilen menkul kıymete yatırım yapılmaktadır.
- Portföyde uzun pozisyona geçildiğinde eldeki nakdin hepsinin ederince menkul kıymet alınmaktadır.
- Uzun pozisyon kapatılırken menkul kıymetin tamamı elden çıkarılmakta ve elde edilen nakit karşılığı açığa satış yapılmaktadır.

Özetlemek gerekirse, kural ağaçlarını değerlendirmede kullanılan portföy modelinin, sistem tasarımı açısından, yalnızca iki durumda bulunabileceği açıktır. Başka bir ifadeyle, portföy sisteminin durumu ya menkul kıymeti taşıdığı uzun (long) pozisyon, ya da daha sonradan almak koşuluyla açığa sattığı (short sell) durumudur. Durumlar arasında geçis, kural ağacından gelen dönüştürülmüş sinyallere göre gerçekleştirilmektedir. Portföy sisteminin durumları arasında geçişin özetlendiği *UML durum diyagramı* Şekil 4.5 'da verilmiştir.

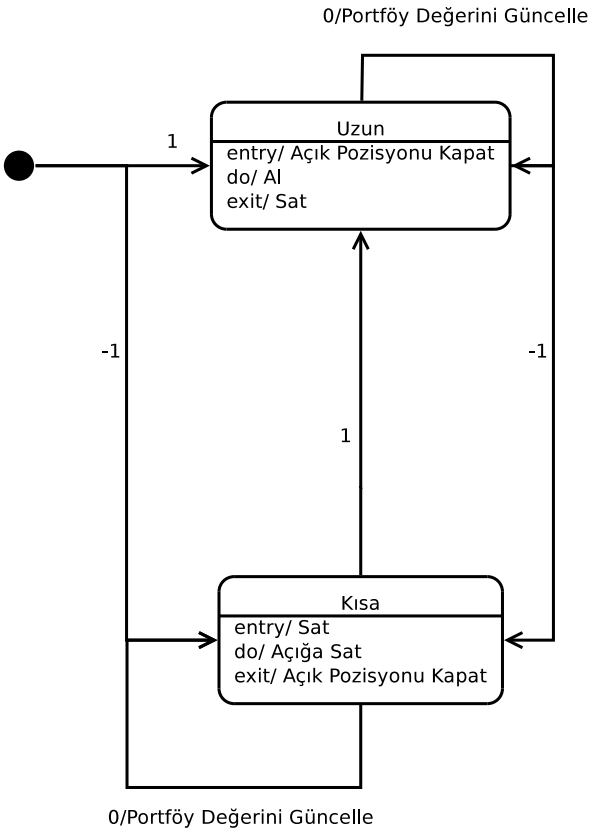
Mevcut bir pozisyonun getirisi Denklem 4.16'deki gibi tanımlanabilir:

$$r_t = -\tau_{t^*}\left(\frac{p_t - p_{t^*}}{p_{t^*}}\right) \tag{4.16}$$

Burada p_{t^*}, sistemin sıfırdan farklı en son dönüştürülmüş sinyali verdiği tarihteki kapanış fiyatı olup r_t ise mevcut pozisyonun o anki getirisidir. Pozisyon değişimleri söz konusu değilken dönüştürülmüş sinyaller her zaman sıfır olacağından r_t de sıfıra eşit olacaktır. Ancak, bir kural ağacının belli bir periyottaki getirisi hesaplanırken mevcut pozisyonların sürdürüldüğü zaman aralıkları için r_t tanımlı değildir. Böylelikle bir kural ağacının tüm eğitim dönemindeki getirisi Denklem 4.17'deki gibi hesaplanabilir

$$R = \prod_{t \in T^*}^{T} (r_t) \tag{4.17}$$

Sonuç olarak, deney çalıştırmalarındaki uyum fonksiyonu $f_{min} = \frac{1}{R}$ biçiminde ifade edilebilecektir.



Şekil 4.5: Model Portföyün Durumlar Arasında Geçişi

4.2.2. **Deney çalıştırmaları**

Deney çalıştırmaları beş farklı test dönemi için gerçekleştirilmiştir. Buna göre; sistemin kısa, orta ve uzun vadede nasıl davrandığını; başka bir deyişle değişik test dönemi uzunlukları için al-bekle stratejisinden devamlı olarak üstün stratejiler geliştirip geliştiremediğinin sınanması amacıyla 5 iş günü (1 hafta), 20 iş günü (1 ay), 60 iş günü (3 ay), 120 iş günü (6 ay) ve 240 iş günü (12 ay) olmak üzere beş farklı test döneminde çalıştırılmıştır. Her test dönemi için sistem değişik rassal besleyiciler kullanılarak 50 defa çalıştırılmış olup her çalıştırmada 500 bireyden oluşan popülasyon 250 nesil boyunca evriltilmiştir. Deneylerde değerlendirilen toplam birey sayısı 5x500x250= 625000'dir.

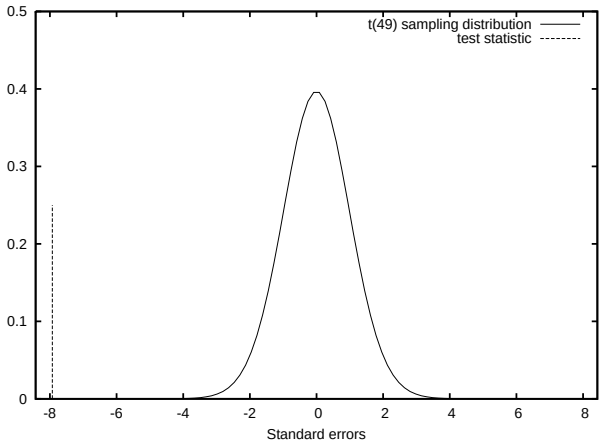
Test dönemi olarak belirlenmiş bu sürelerin teknik analiz göstergelerinin hesaplanmasında genel kabul görmüş süreler olduğu bilinmektedir. Ayrıca sermaye piyasalarında hisseleri işlem görmekte olan şirketlerin her üç ayda bir finansal tablolarını raporladıkları göz önüne alındığında belirlenen test sürelerinin kabul edilebilir ve finansal takvim anlamında önemli süreler oldukları ortadadır.

Sistem; tarihi veriler veritabanından IMKB U30 endeksine ait, 1 Ocak 2000 tarihinden, çalıştırıldığı tarihteki en güncel veri olan 15 Mart 2012 tarihine kadar olan (yaklaşık 12 yıllık) 3077 gözlemden oluşan günlük veriyi kullanmaktadır. Sistemin ihtiyaç duyduğu teknik analiz araçları geriye dönük veri gereksinimine sahip olduğundan bu göstergelerin rahatlıkla hesaplanabilmeleri için ilk 50 gözlem bu ön hesaplamalar döneminde gözden çıkarılmış olup eğitim ve test dönemlerinin bütününde geriye kalan 3027 günlük veri kullanılmıştır.

4.2.2.1. ***Test periyodu: 1 hafta (5 iş günü)***

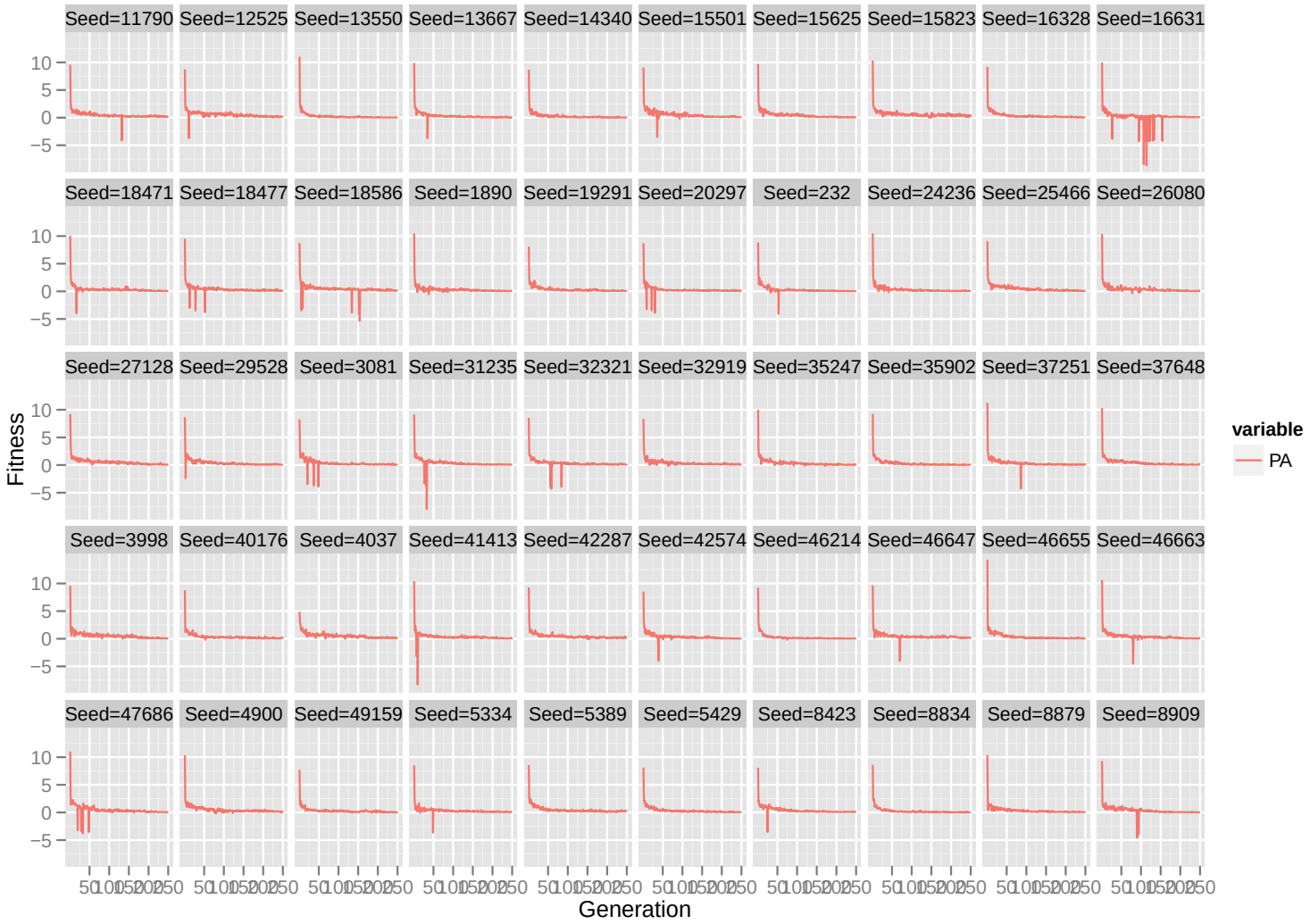
Bu deneyde sistem 1 Ocak 2000 tarihinden 8 Mart 2012 (dahil) tarihine kadar olan 3072 iş günü verisi üzerinde eğitilerek, eğitim süresinin bittiği tarihi izleyen ilk 5 iş gününde test edilmiştir. Test dönemi başlangıç tarihi 9 Mart 2012, bitiş tarihi ise 15 Mart 2012'dir.

Sistem, eğitim döneminde, 50 değişik rassal besleyici verilerek çalıştırılmıştır. Bu çalıştırmaların her biri için uyum fonksiyonunun popülasyon ortalamasının nesiller boyunca değişimi Şekil 4.7'de verilmiştir. Görülmektedir ki; popülasyon ortalaması, beklendiği gibi sıfıra yakınsamaktadır. Yine bu çalıştırmalar esnasında elde edilen her nesilde bu-

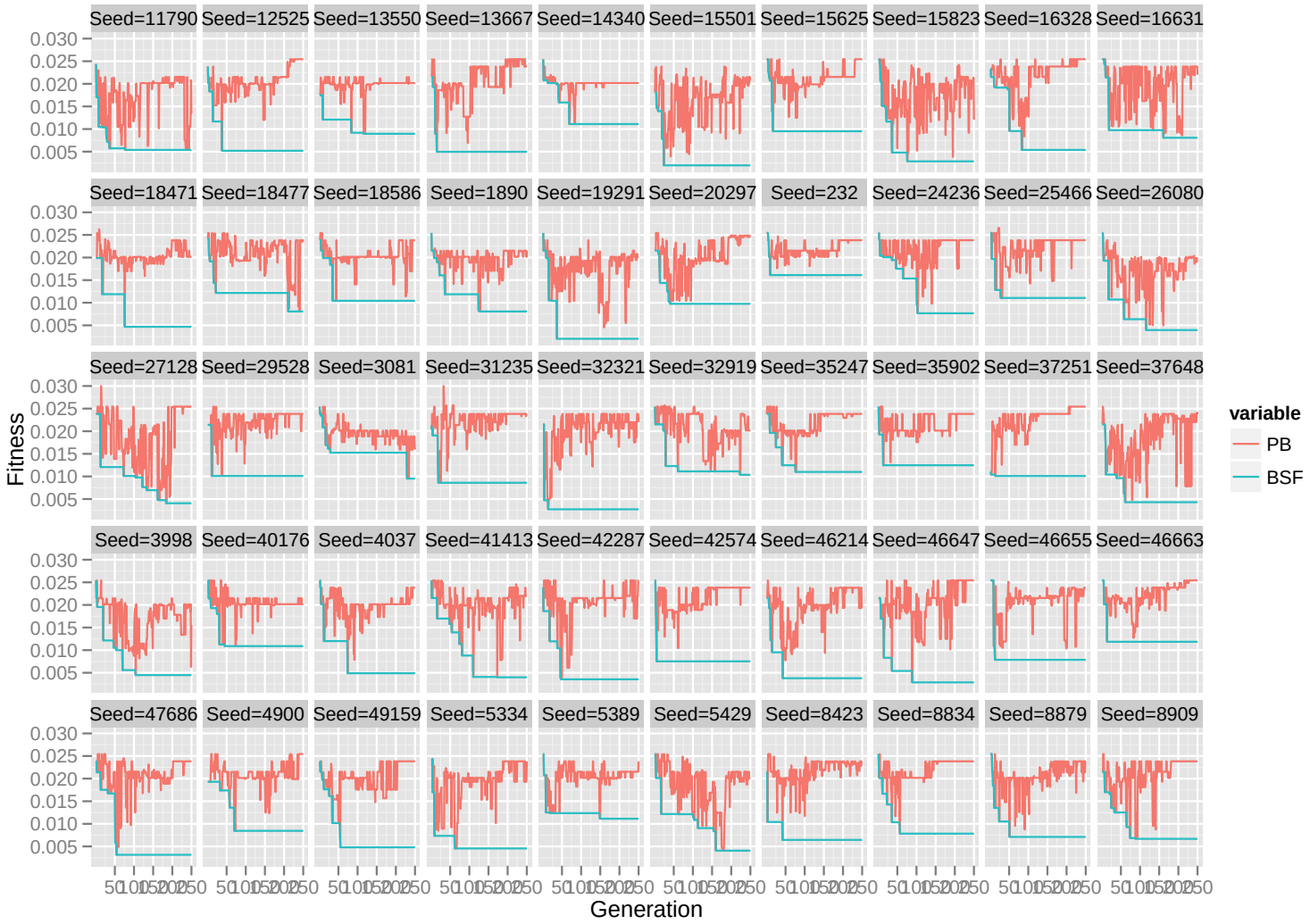


Şekil 4.6: 5 Günlük Test Periyodu İçin Elde Edilen 50 Modele İlişkin t Testi

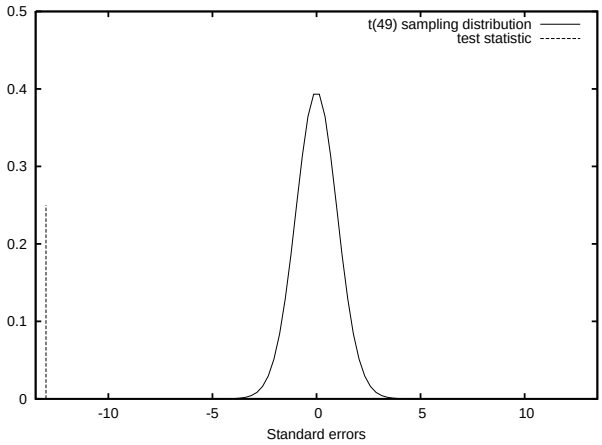
lunan popülasyon en iyisi ve o ana kadar elde edilen en iyi bireyin uyum fonksiyonu değerleri Şekil 4.8'de verilmiştir.



Şekil 4.7: 5 Günlük Test Periyodu İçin Ortalama Uyum Değerleri



Şekil 4.8: 5 Günlük Test Periyodu İçin Popülasyon En İyisi Ve Bulunan En İyi Bireyin Uyum Değerleri

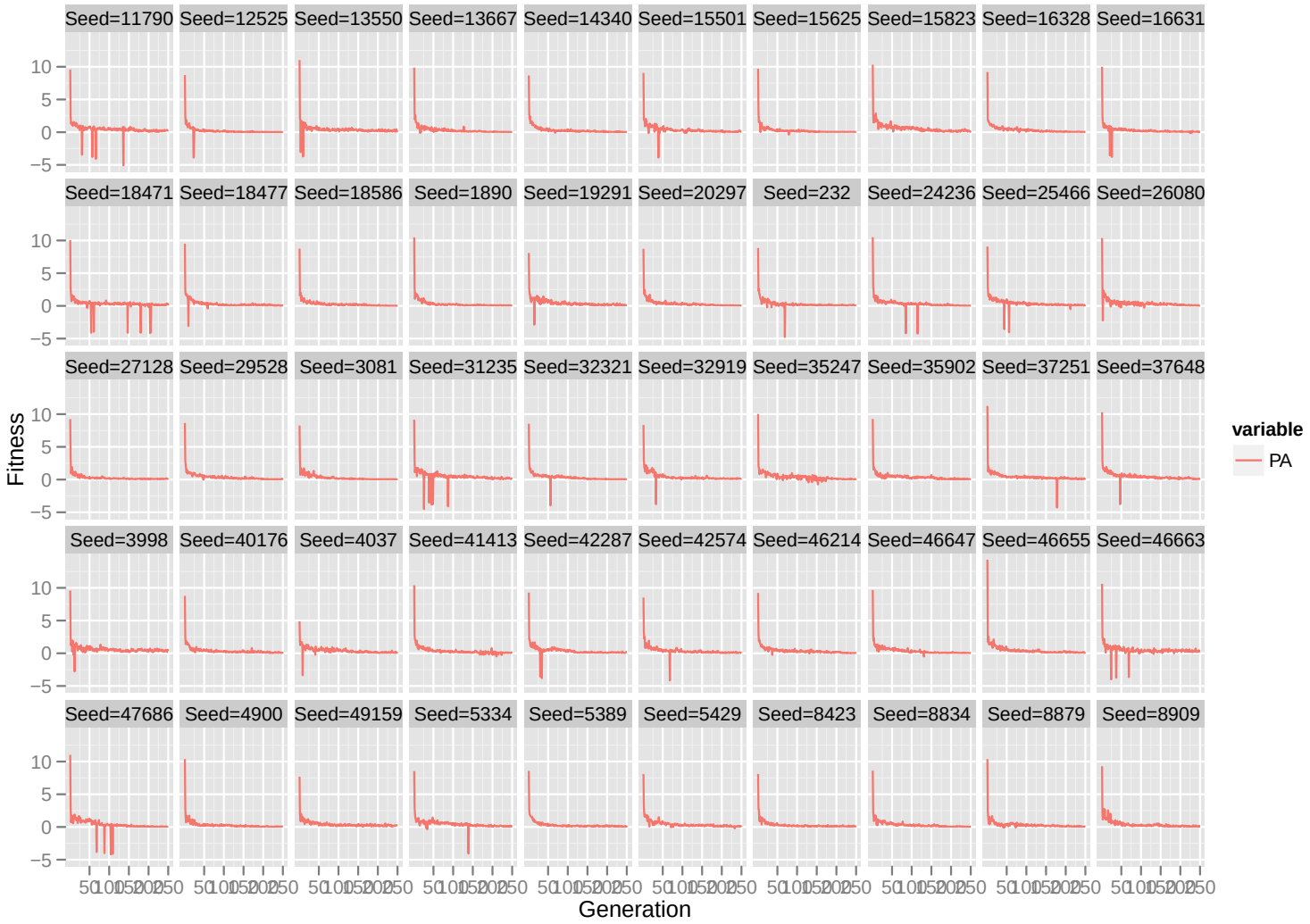


Şekil 4.9: 20 Günlük Test Periyodu İçin Elde Edilen 50 Modele İlişkin t Testi

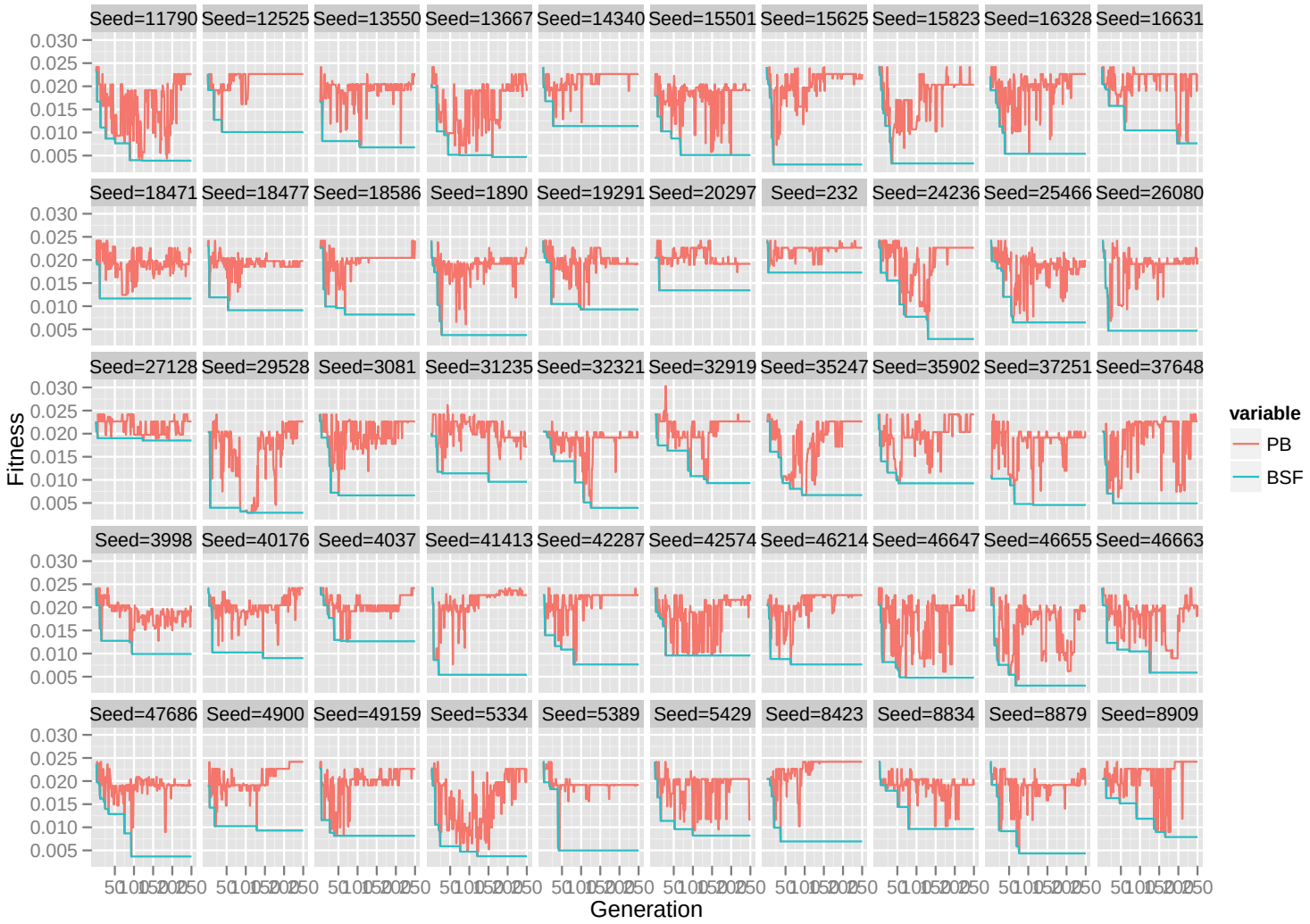
4.2.2.2. ***Test periyodu: 1 ay (20 iş günü)***

Bu deneyde sistem 1 Ocak 2000 tarihinden 16 Şubat 2012 (dahil) tarihine kadar olan 3057 iş günü verisi üzerinde eğitilerek, eğitim süresinin bittiği tarihi izleyen ilk 20 iş gününde test edilmiştir. Test dönemi başlangıç tarihi 17 Şubat 2012, bitiş tarihi ise 15 Mart 2012'dir.

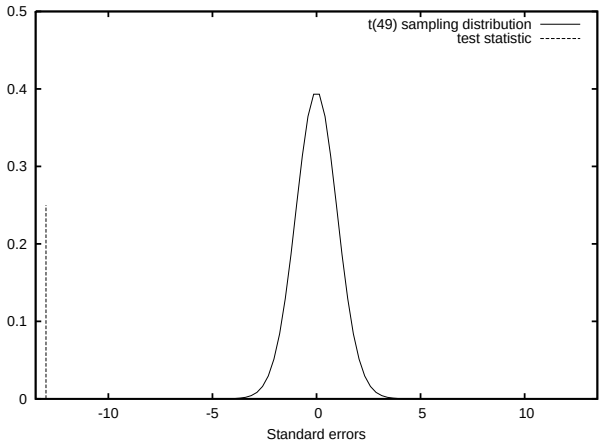
Sistem, eğitim döneminde, 50 değişik rassal besleyici verilerek çalıştırılmıştır. Bu çalıştırmaların her biri için uyum fonksiyonunun popülasyon ortalamasının nesiller boyunca değişimi Şekil 4.10'de verilmiştir. Görülmektedir ki; popülasyon ortalaması, beklendiği gibi sıfıra yakınsamaktadır. Yine bu çalıştırmalar esnasında elde edilen her nesilde bulunan popülasyon en iyisi ve o ana kadar elde edilen en iyi bireyin uyum fonksiyonu değerleri Şekil 4.11'de verilmiştir.



Şekil 4.10: 20 Günlük Test Periyodu İçin Ortalama Uyum Değerleri



Şekil 4.11: 20 Günlük Test Periyodu İçin Popülasyon En İyisi Ve Bulunan En İyi Bireyin Uyum Değerleri

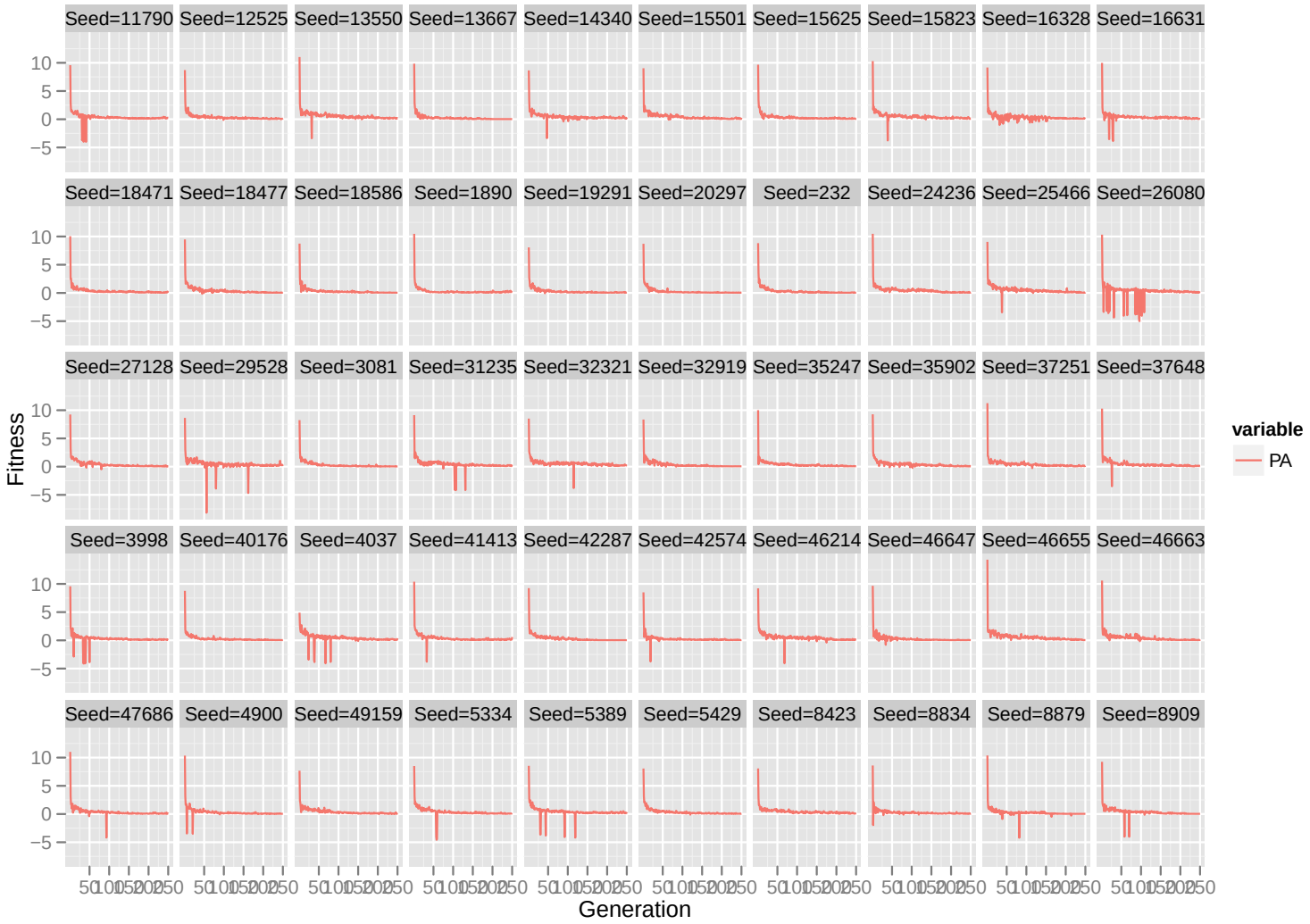


Şekil 4.12: 60 Günlük Test Periyodu İçin Elde Edilen 50 Modele İlişkin t Testi

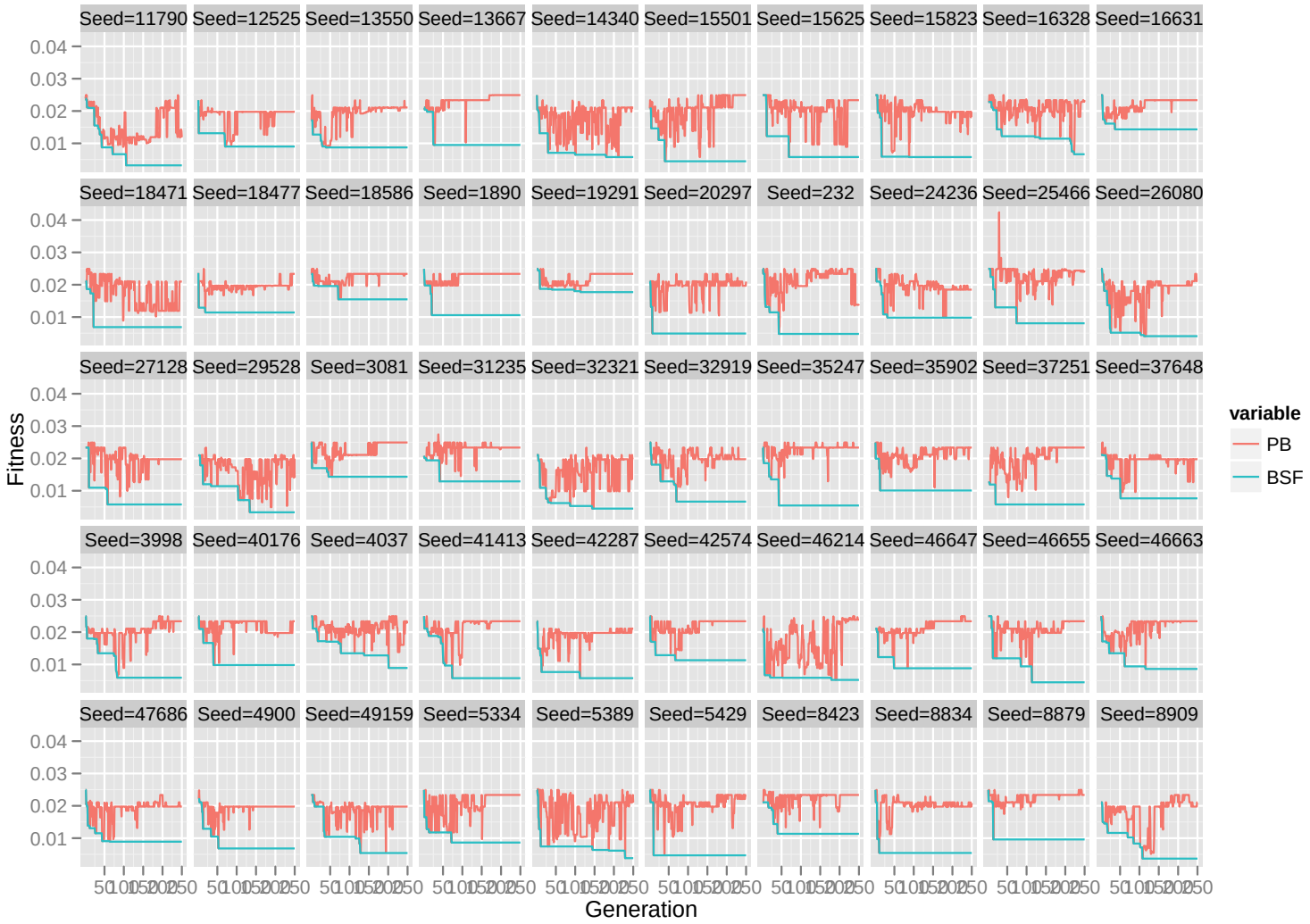
4.2.2.3. ***Test periyodu: 3 ay (60 iş günü)***

Bu deneyde sistem 1 Ocak 2000 tarihinden 22 Aralık 2011 (dahil) tarihine kadar olan 3017 iş günü verisi üzerinde eğitilerek, eğitim süresinin bittiği tarihi izleyen ilk 60 iş gününde test edilmiştir. Test dönemi başlangıç tarihi 23 Aralık 2011, bitiş tarihi ise 15 Mart 2012'dir.

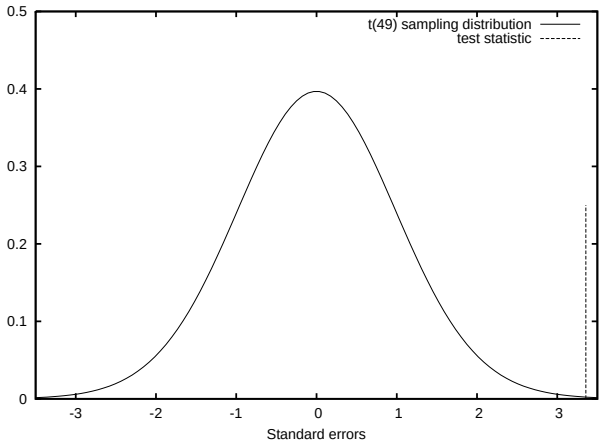
Sistem, eğitim döneminde, 50 değişik rassal besleyici verilerek çalıştırılmıştır. Bu çalıştırmaların her biri için uyum fonksiyonunun popülasyon ortalamasının nesiller boyunca değişimi Şekil 4.13'de verilmiştir. Görülmektedir ki; popülasyon ortalaması, beklendiği gibi sıfıra yakınsamaktadır. Yine bu çalıştırmalar esnasında elde edilen her nesilde bulunan popülasyon en iyisi ve o ana kadar elde edilen en iyi bireyin uyum fonksiyonu değerleri Şekil 4.14'de verilmiştir.



Şekil 4.13: 60 Günlük Test Periyodu İçin Ortalama Uyum Değerleri



Şekil 4.14: 60 Günlük Test Periyodu İçin Popülasyon En İyisi Ve Bulunan En İyi Bireyin Uyum Değerleri

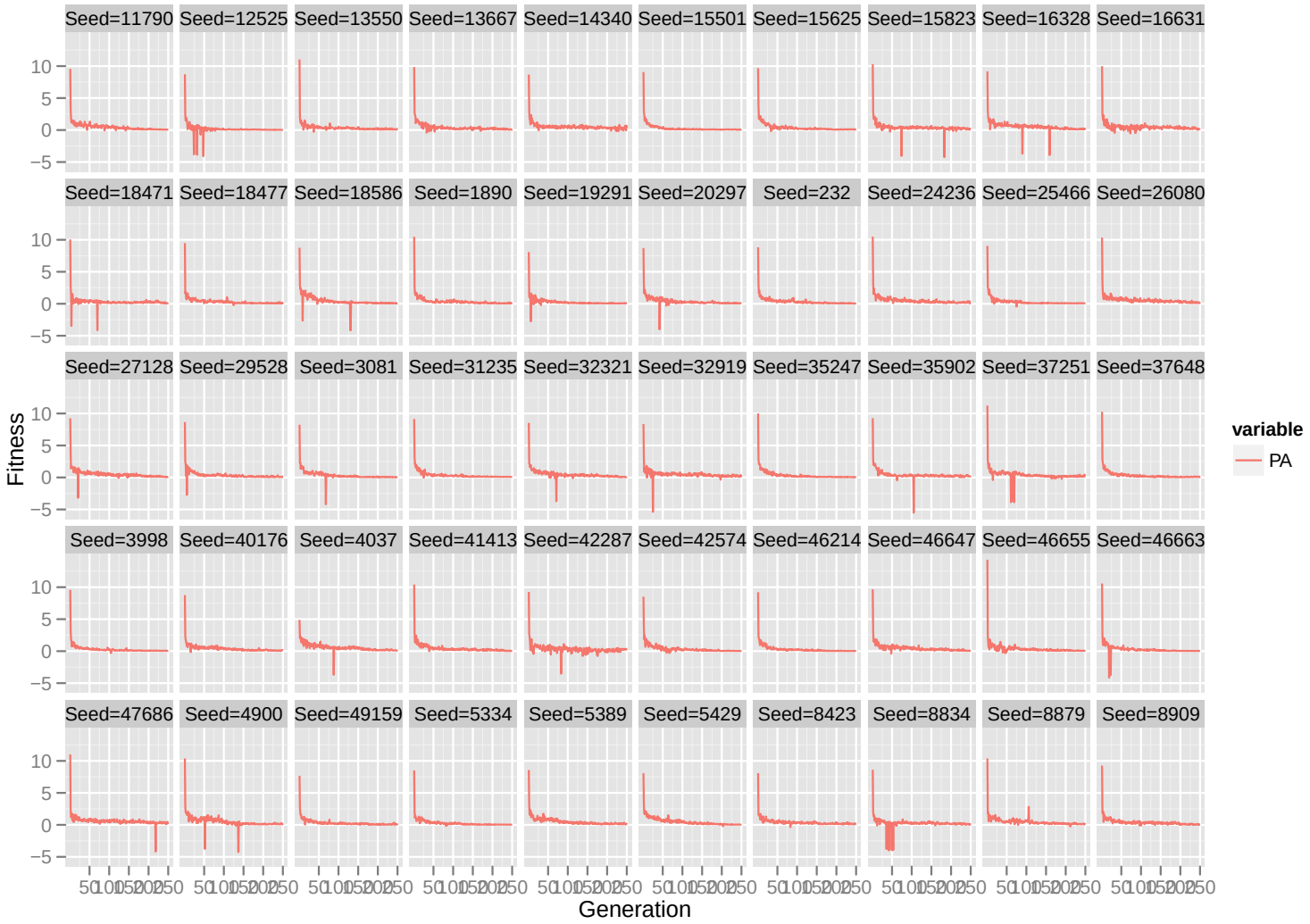


Şekil 4.15: 120 Günlük Test Periyodu İçin Elde Edilen 50 Modele İlişkin t Testi

4.2.2.4. ***Test periyodu: 6 ay (120 iş günü)***

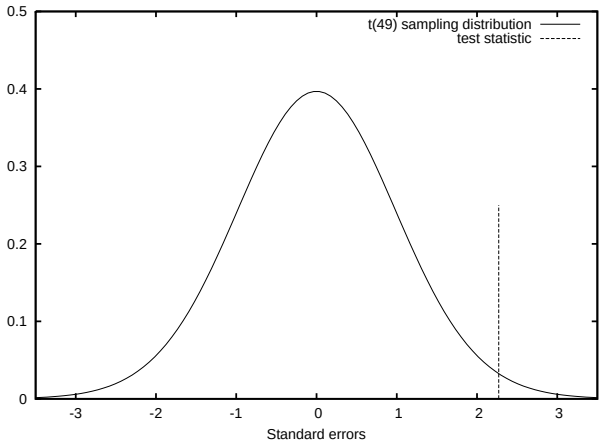
Bu deneyde sistem 1 Ocak 2000 tarihinden 26 Eylül 2011 (dahil) tarihine kadar olan 2957 iş günü verisi üzerinde eğitilerek, eğitim süresinin bittiği tarihi izleyen ilk 120 iş gününde test edilmiştir.Test dönemi başlangıç tarihi 27 Eylül 2011, bitiş tarihi ise 15 Mart 2012'dir.

Sistem, eğitim döneminde, 50 değişik rassal besleyici verilerek çalıştırılmıştır. Bu çalıştırmaların her biri için uyum fonksiyonunun popülasyon ortalamasının nesiller boyunca değişimi Şekil 4.16'de verilmiştir. Görülmektedir ki; popülasyon ortalaması, beklendiği gibi sıfıra yakınsamaktadır. Yine bu çalıştırmalar esnasında elde edilen her nesilde bulunan popülasyon en iyisi ve o ana kadar elde edilen en iyi bireyin uyum fonksiyonu değerleri Şekil 4.17'de verilmiştir.



Şekil 4.16: 120 Günlük Test Periyodu İçin Ortalama Uyum Değerleri

Şekil 4.17: 120 Günlük Test Periyodu İçin Popülasyon En İyisi Ve Bulunan En İyi Bireyin Uyum Değerleri

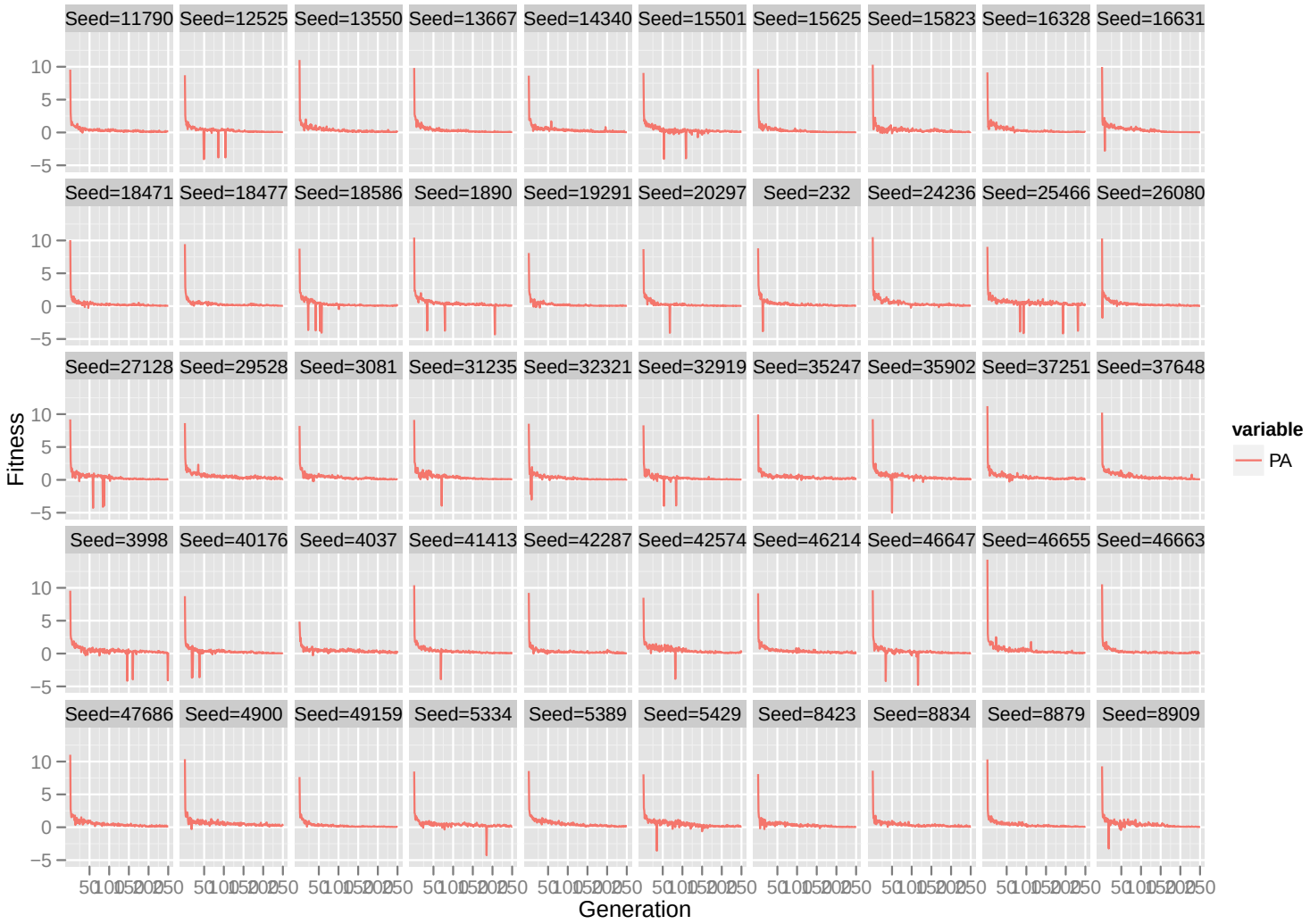


Şekil 4.18: 240 Günlük Test Periyodu İçin Elde Edilen 50 Modele İlişkin t Testi

4.2.2.5. ***Test periyodu: 12 ay (240 iş günü)***

Bu deneyde sistem 1 Ocak 2000 tarihinden 5 Nisan 2011 (dahil) tarihine kadar olan 2837 iş günü verisi üzerinde eğitilerek, eğitim süresinin bittiği tarihi izleyen ilk 240 iş gününde test edilmiştir.Test dönemi başlangıç tarihi 6 Nisan 2011, bitiş tarihi ise 15 Mart 2012'dir.

Sistem, eğitim döneminde, 50 değişik rassal besleyici verilerek çalıştırılmıştır. Bu çalıştırmaların her biri için uyum fonksiyonunun popülasyon ortalamasının nesiller boyunca değişimi Şekil 4.19'de verilmiştir. Görülmektedir ki; popülasyon ortalaması, beklendiği gibi sıfıra yakınsamaktadır. Yine bu çalıştırmalar esnasında elde edilen her nesilde bulunan popülasyon en iyisi ve o ana kadar elde edilen en iyi bireyin uyum fonksiyonu değerleri Şekil 4.20'de verilmiştir.



Şekil 4.19: 240 günlük test dönemi için ortalama uyum değerleri

Şekil 4.20: 240 Günlük Test Periyodu İçin Popülasyon En İyisi Ve Bulunan En İyi Bireyin Uyum Değerleri

5. Sonuç, Öneriler ve Tartışma

Finansal sistemler, sürekli olarak gerçek zamanlı değeri olan büyük miktarlarda veri üreten bir sistemlerdir. Dahası, piyasa oyuncularınca bu veriye eksiksiz ve zamanında erişimin kritik derecede önemli olduğu çok dinamik bir mekanizmadır. Veriye erişimin zamanlamasının son derece önem taşıması, finansal verilerin toplanıp saklanması ve sunulmasında teknolojinin geldiği son noktadaki bütün imkanların seferber edildiği bir finansal hizmet pazarını ortaya çıkarmıştır. Söz konusu finansal hizmetler, bireysel yatırımcılarda gerçek zamanlı veriye erişimi sağlayan teknik analiz paket programları ve piyasa takip platformları olarak karşımıza çıkmaktadır. Öte yandan, bu hizmetler kurumsal piyasa oyuncularında ise güçlü fiziksel donanımlar üzerinde çalışan gerçek zamanlı ve otonom karar verme mekanizmalarının yardımıyla spekülasyon yeteneğine sahip son derece karmaşık yazılımlar şeklinde vucut bulmaktadır.

Genel olarak "algoritmik alım-satım sistemleri" *(algorithmic -automated- trading systems)* olarak adlandırılmakta olan bu tip sistemler; alım-satım emirlerinin zamanlama, miktar ve fiyat gibi niteliklerinin insan müdahalesi olmaksızın bir algoritma yardımıyla belirlenip eyleme geçirilmesini sağlamaktadır. Algoritmik alım-satım sistemlerinin bir alt dalı olarak değerlendirilen ve genellikle yüksek sıklıkta al-sat işlemi yapan sistemler *(high-frequency trading - HFT)* olarak sınıflandırılan sistemler de son zamanlarda öne çıkmaktadır. HFT sistemleri sermaye piyasalarında neredeyse tamamen otomatik olarak çok dar zaman aralıklarında çok sayıda işlem yapabilmekte ve dahası çok küçük miktardaki fiyat hareketlerinden dahi spekulatif kazanç sağlayabilmektedir. Bu sistemleri kullanan fonların ve özel yatırım şirketlerinin piyasalara likidite anlamında olumlu etkileri olduğu iddia edilse de, çoğu uzman tarafından anlamsız bulunan ani ve sert fiyat hareketleri yaratabildikleri ve piyasalarda volatiliteyi arttırdıkları da öne sürülmektedir. Örneğin; 6 Mayıs 2010 tarihinde yaşanan ve Dow Jones endeksinin bir günde yaklaşık 1000 puan

(% 9) düşmesinin arkasında bu sistemlerin olduğu iddia edilmektedir [1].

Bu durumda, bireysel yatırımcıların kurumsal yatırımcılar karşısında geleneksel yöntemlere dayanan stratejilerle yola devam etmeleri oldukça güçtür. Öyle ki; kurumsal yatırımcılar nitelikli veriye erişim olanakları, işlem hızı ve maliyetleri bakımından bireysel yatırımcılardan çok daha üstün imkanlarla donanmış olduğundan birkaç adım önde yarışa başlamaktadır.

Öte yandan, günümüz kişisel bilgisayarları bile çok yakın zaman öncesine kadar ancak sunucu veya özel amaçlarla yapılandırılmış donanımlardan elde edilen performansı yakalamıştır. İşlemciler giderek daha performanslı hale gelmekte olup çok sayıda çekirdekle piyasaya sürülmektedir. Donanım seviyesinde paralel mimari sıradan hale gelmiş olup bellek ve harici sabit disk maliyetleri ise göz ardı edilebilir seviyelere düşmüştür.

Yazılım boyutunda ise donanımdaki gelişmelerin hızının gerisinde kalmayacak nitelikte gelişmeler yaşanmaktadır. Açık kaynak kodlu ve özgür yazılımların yakaladığı ivme çok üst seviyelere ulaşmıştır. Öyle ki; geliştiriciler için çok sayıda serbest lisanslı Unix türevi işletim sistemi doğal geliştirme ortamı haline gelmiştir. Bunların üzerinde çalışan eski veya yeni onlarca programlama dili yorumlayıcısını, derleyicisini ve çalışma zamanı ortamını kullanmak için herhangi bir ücret ödenmediği gibi bunların serbest lisanslı olması ayrı bir sinerji doğurmaktadır. Örneğin bu sinerji; veritabanı yönetim sistemleri, grafik kullanıcı arabirim araçları veya değişik amaçlarla geliştirilmiş kütüphaneler gibi çok sayıda yardımcı yazılımın da türlü programlama dilleri için arayüzlerini ortaya çıkarmaktadır.

Ülkemizde finansal konularda yapılan araştırmalarda bile araştırmacılarca veritabanı oluşturma ve kullanımına rastlamak neredeyse imkansızdır. Bunun yerine, geçici ve üstünkörü bir biçimde eldeki probleme ilişkin analizlerin talep ettiği kadar verinin organize olmayan bir biçimde edinildiği gözlemlenmektedir. Oysa ki; gelişen web teknolojileri altyapısı ve zenginleşen web hizmetleri ikincil finansal veri kaynaklarına ulaşmada araştırmacılara büyük olanaklar sunmaktadır.

Mevcut şartlar altında; bireysel yatırımcıların, yatırım kararlarını verirken bilgisayarlı

[1] Daha fazla bilgi için bkz.:

`http://en.wikipedia.org/wiki/2010_Flash_Crash`

(Erişim tarihi: 05.04.2012)

karar destek sistemlerine başvurmaları şimdilik çok olası görünmemektedir. Bunun yerine teknik analiz yapmayı mümkün kılan ve anlık veri desteğine sahip piyasa takip programlarının yoğun kullanımı gözlemlenmektedir. Halbuki; yukarıda değinilen donanım ve yazılım teknolojilerindeki gelişmeler; küçük yatırımcılara da büyük kurumsal yatırımcıların karşısında yatırım kararlarını veririken istatistiksel tekniklerden ya da değişik yapay zeka (makine öğrenimi) tekniklerinden yararlanabilecekleri karar destek sistemlerini geliştirebilme ve kullanabilme şansını vermektedir.

Bu çalışmayla günlük verilere dayalı al-sat sinyalleri üreten bir sistemin veri ihtiyacının ikincil veri kaynaklarından rahatlıkla karşılanabileceği gösterilmiş olup uzun vadede GP gibi bir evrimsel algoritma yardımıyla al-sat kararları veren otomatik bir sistemin al-bekle stratejesinden sürekli olarak üstün stratejiler geliştirebileceği ortaya konulmuştur. Ancak kısa vadeli al-sat kararlarında böyle bir çıkarsamayı destekleyecek sonuçlara ulaşılamamıştır.

Teknik analiz bilimsel bir disiplin sayılmamasına karşın, yatırım kararlarının alınmasında profesyonellerce kullanım yaygınlığı son derece üst seviyelerde olan bir araçtır. Zaman içerisinde farklı teknik analiz göstergeleri ortaya çıkmış olup bunların bir kısmı sektör uzmanlarınca geniş kabul görmüştür. Temelde birçok teknik analiz göstergesi geçmiş fiyat ve hacim bilgisine dayalı olarak hesaplanmaktadır. Teknik analiz mevcut fiyatların aşırı alım veya satım bölgesinde olup olmadıklarını veya piyasa uzmanlarınca benimsenmiş olan kısa / orta / uzun vade uzunlukları için hesaplanan hareketli ortalamaların birbirlerine göre durumlarını veya belirli bir zaman çerçevesi içerisindeki yerel yüksek/düşük fiyatlara göre fiyatın nerede bulunduğunu belirlemeye çalışan gösterge ve osilatörlerin değerlerinin belirli yorumlarına göre yapılmaktadır. Osilatörlerin geri kalan göstergelerden farkı sabit bir alt ve üst sınır arasında salınmalarıdır. Osilatör olmayan göstergeler ise böyle bir durumda olmayıp değişik değerler alabilmekte ve genellikle kendisinden türetilen başka bir tetikleyici değer ile karşılaştırılarak yorumlanmaktadır. Sonuçta, teknik analiz geçmiş fiyatlardan hareketle menkul kıymetin mevcut fiyatının bulunduğu seviyelerin yatırım kararlarına uygun olup olmadığı kararının verilmesinde yatırımcıya yardımcı olmaktadır.

Her ne kadar teknik analiz bilimsel bir disiplin olarak sayılmasa da; bilimsel tekniklerle gerçek uzmanların başvurduğu bu aracı kullanarak uzman bilgisini taklit edip

mantıklı kararlar üretmeye çalışan sistemler tasarlamak ve hayata geçirmek; uygulamalı bilimler anlamında bilimsel bir çabadır.

Bu bağlamda elde edilen sonuçlar, geliştirilen sistemin herhangi bir ön bilgi olmadan elde ettiği performansın gerek Türkiye gerekse uluslararası menkul kıymet piyasalarında kullanılabileceğini ve geliştirilmeye açık bir sistem olduğunu göstermektedir.

Uygulama kapsamında elde edilen sonuçlar gözden geçirildiğinde, geliştirilen sistemin 5, 20 ve 60 günlük gibi kısa sayılabilecek vadelerde al-bekle stratejisinden her zaman daha kötü stratejiler geliştirdiği; buna karşın 6 aylık ve 12 aylık gibi uzun vadeli stratejilerde sürekli olarak al-bekle stratejisinden üstün sonuçlar bulabildiği ortaya konulmuştur. Bu bulgu; sistemin görece uzun vadelerde fiyat farklılıkları yaratan aşağı ve yukarı yönlü trendleri doğru zamanlamalarla yakalayabildiğini ortaya koymaktadır. Sistemin başarısız olduğu vadeler, günlük veriler için kısa sayılabilirken, gözlem sayısının çok daha fazla olacağı 60 dakikalık veya daha kısa periyotlu veri kümelerinde uzun vade sayılabilmektedir. Bu durum; sistemin söz konusu vadelerde, daha kısa periyotlu ve sık frekanslı verilerle başarılı olabileceği anlamına gelmektedir.

İkincil kaynaklardan elde edilen günlük verilerde karşılaşılan sorunların başında, özellikle de hisse senedi verilerinde, hacim ve açılış fiyatı verilerinin sağlıklı olmaması gelmektedir. Hacim ve açılış verilerinin eksik ya da hatalı oluşu bunlara ihtiyaç duyan bazı teknik analiz göstergelerinin hesaplanmasını olanaksız hale getirmektedir. Halbuki böyle bir olanak elde edilebilirse uygulama kapsamında geliştirilen sistemdeki teknik analiz göstergelerinden oluşan terminal çeşitliliği artırılabilir. Hacim verisinin bünyesinde barındıran terminaller hacim artış ve azalışlarını yaratan etkenlerin tekrarlandığı durumlarda ortaya çıkan fiyat hareketlerinin tekrarı durumunda sistemin doğru pozisyonlar önermesini sağlayabilir. Benzer şekilde açılış fiyatının (gün içi en yüksek, gün içi en düşük ve kapanış gibi) diğer fiyat bilgilerine göre aldığı değeri temsil eden göstergelerin terminal olarak eklenmesi fiyat boşluklarının kararlarda etkili olabileceği bir altyapı doğurabilir.

Birincil ve gerçek zamanlı verinin otomatikleştirilmiş al-sat sistemlerine sağlayacağı üstünlük son derece üst seviyededir. Ancak, bu gibi veri beslemelerine sahip olmayan küçük yatırımcıların görece daha uzun vadede isabetli yatırım kararları alabilen sistemleri kullanmaları, algoritmik alım-satım yapan kurumsal sistemlerin karşısında en azından al-bekle stratejisinden daha iyi imkanlar sağlayacaktır.

A. GP Kütüphanelerine İlişkin Ekler

A.1. Common Lisp ile Yazılmış Bir GP Sisteminde Birey Gösterimleri ve Büyüme (Grow) Yöntemiyle Popülasyon Oluşturulması

Aşağıda verilen program, Common Lisp kullanılarak ağaç tabanlı bir GP sisteminin kolaylıkla geliştirilebileceğini göstermek amacıyla çalışmanın yazarı tarafından geliştirilmiştir. Bu program tam ve fonksiyonel bir GP sistemi olmamakla beraber belli bir terminal ve fonksiyon kümesi yardımıyla rassal program ağaçlarından oluşan bir popülasyonu *büyüme (grow)* yöntemini kullanarak oluşturmaktadır.

```
;=============================================================================
;;;; gpsys.lisp
;;;; A future GP implementation in Common  Lisp.
;;;; Written by Özgür İcan

;;; Utility functions
(defun random-select(lst)
  (let ((len (length lst)))
    (nth (random len) lst)))

(defun tree-height(lst)
  (if (consp lst)
      (tree-height-1 lst 1)
      (tree-height-1 lst 0)))

(defun tree-height-1(tree height)
  (cond ((null tree) height)
        ((atom tree) height)
        ((consp (car tree))
         (max (tree-height-1 (car tree) (1+ height))
              (tree-height-1 (cdr tree) height)))
        (t (tree-height-1 (cdr tree) height))))

;;; Function Set - Terminal Set Definitions
(defun add(x y)
  (+ x y))
```

```
(defun sub(x y)
  (- x y))

(defun mul(x y)
  (* x y))

(defun div(x y)
  ; Protected division.
  (if (eql y 0)
      1
      (funcall #'/ x y)))

(defstruct fun-node
  val
  (arity 0))

(defstruct term-node
  val)

(defvar *add* (make-fun-node :val 'add :arity 2))
(defvar *sub* (make-fun-node :val 'sub :arity 2))
(defvar *mul* (make-fun-node :val 'mul :arity 2))
(defvar *div* (make-fun-node :val 'div :arity 2))
(defvar *exp* (make-fun-node :val 'expt :arity 2))
(defvar *log* (make-fun-node :val 'log :arity 1))

(defvar *arg1* (make-term-node :val 'x))
(defvar *arg2* (make-term-node :val 'ran))

(defvar *arithmetic-fun-set* (list *add* *sub* *mul* *div*))
(defvar *ext-arithmetic-fun-set* (list *add* *sub* *mul* *div* *exp* *log*))
(defvar *basic-terminal-set* (list *arg1* *arg2*))

;;; Fundamental structures
(defstruct individual
  data
  (raw-fitness 0)
  (adj-fitness 0)
  (hits 0))

(defstruct population
  data
  best-ind-so-far
  (average-raw-fitness 0)
  (average-adj-fitness 0)
  (best-ind-raw-fitness 0)
```

```
  (best-ind-adj-fitness 0)
  (raw-fitness-std-dev 0)
  (adj-fitness-std-dev 0))

(defstruct gp
  objective-function
  (function-set *arithmetic-fun-set*)
  (terminal-set *basic-terminal-set*)
  selection-tech
  (initial-pop-generation-tech #'grow)
  (pop-size 100)
  (ind-init-depth 5)
  (ind-max-depth 50)
  (termination-criteria "max-iter"))


;;; Individual and population initialization
(defun init-population(pop-obj gp-obj)
  (let ((size (gp-pop-size gp-obj)))
    (setf (population-data pop-obj)
          (loop for i from 0 below size collect
               (init-individual (make-individual) gp-obj)))))

(defun init-individual(ind-obj gp-obj)
  (let ((f (gp-initial-pop-generation-tech gp-obj)))
    (setf (individual-data ind-obj) (funcall f gp-obj))))

(defun grow(gp-obj)
  (let* ((max-depth (gp-ind-init-depth gp-obj))
         (fs (gp-function-set gp-obj))
         (ts (gp-terminal-set gp-obj))
         (mix-set (append fs ts)))
    (grow1 fs ts mix-set 0 max-depth)))

(defun grow1(fs ts ms cd md)
   (cond ((eql cd md) nil)
         ((eql cd 0)
          (let* ((f-node (create-fun-node fs))
                 (f-obj (fun-node-val f-node))
                 (f-arity (fun-node-arity f-node))
                 (expr (loop for i from 0 below f-arity collect
                             (grow1 fs ts ms (1+ cd) md))))
            (cons f-obj expr)))
         ((eql cd (1- md))
          (let* ((t-node (create-term-node ts))
                 (t-obj (term-node-val t-node)))
            t-obj))
```

```
          (t
           (let* ((node (create-any-node ms))
                  (t? (term-node-p node)))
             (if t? (term-node-val node)
                 (let* ((f-obj (fun-node-val node))
                        (f-arity (fun-node-arity node))
                        (expr (loop for i from 0 below f-arity collect
                                    (grow1 fs ts ms (1+ cd) md))))
                   (cons f-obj expr)))))))

(defun create-fun-node(fs)
  (random-select fs))

(defun create-term-node(ts)
  (random-select ts))

(defun create-any-node(ms)
  (random-select ms))

;(defun full(ind-obj gp-obj))
;(defun ramped-half-and-half(ind-obj gp-obj))

;;; TEST

(defvar *gp* (make-gp))
(defvar *pop* (make-population))
(init-population *pop* *gp*)
;; (defvar *ind* (make-individual))
;; (init-individual *ind* *gp*)
;; (format t "~A" (loop for i in (gp-function-set *gp*) collect
;;                (funcall (fun-node-val i) 3 4)))
;; (format t "~&~A~&" (individual-data *ind*))
```

A.2. Little LISP Kullanılarak Hipotetik Veriye Uygun Bir Polinom Bulunması

Aşağıda açıklanan program, Little LISP GP kütüphanesi kullanılarak geliştirilmiş bir sembolik regresyon uygulaması örneğidir. Bu örnekte; iki bağımsız değişkenden oluşan hipotetik $z = x^2y + xy^2$ polinomu temel alınarak x ve y bağımsız değişkenleri ve bunların rassal değerlerinden türetilmiş $z = f(x,y)$ fonksiyonunun bu değişkenlerin ilgili değerlerine karşılık gelen değeri hedef olmak üzere $\hat{f}(x,y)$ fonksiyonunun matematiksel formunu tahmin etmeye çalışan bir GP uygulamasıdır.

Bu uygulamada, 50 x ve y türetilmiş gözlem çifti ve $z = f(x,y)$ değeri veri kümesi olarak kullanılmıştır. Bu uygulama için *standartlaştırılmış* ve *ham uyum fonksiyonu* aynı olup z 'nin gerçek ve tahmin edilen değerleri arasındaki farkın mutlak toplamıdır.

```
;============================================================================
;;; Discovering a formula for the polynom x^2.y + x.y^2

(defvar y)
(defvar x)

(defun define-terminal-set-for-POLYNOM ()
  (values '(x y :floating-point-random-constant)))

(defun define-function-set-for-POLYNOM ()
  (values '(+ - * %)
          '(2 2 2 2)))

(defstruct POLYNOM-fitness-case
  independent-variable-1
  independent-variable-2
  target)

(defun define-fitness-cases-for-POLYNOM ()
  (let (fitness-cases x y this-fitness-case)
    (setf fitness-cases (make-array *number-of-fitness-cases*))
    (format t "~%Fitness cases")
    (dotimes (index *number-of-fitness-cases*)
      (setf x (float (/ index *number-of-fitness-cases*)))
      (setf y (* (mod (/ index *number-of-fitness-cases*) 0.05) 10e4))
      (setf this-fitness-case (make-POLYNOM-fitness-case))
      (setf (aref fitness-cases index) this-fitness-case)
      (setf (POLYNOM-fitness-case-independent-variable-1 this-fitness-case) x)
      (setf (POLYNOM-fitness-case-independent-variable-2 this-fitness-case) y)
```

```
      (setf (POLYNOM-fitness-case-target this-fitness-case)
            (+ (* (* x y) x) (* (* x y) y)))
      (format t "~% ~D ~D ~D ~D"
              index
              (float x)
              (float y)
              (POLYNOM-fitness-case-target this-fitness-case)))
    (values fitness-cases)))

(defun POLYNOM-wrapper (result-from-program)
  (values result-from-program))

(defun evaluate-standardized-fitness-for-POLYNOM  (program fitness-cases)
  (let (raw-fitness hits standardized-fitness x y target-value
                    difference value-from-program this-fitness-case)
    (setf raw-fitness 0.0)
    (setf hits 0)
    (dotimes (index *number-of-fitness-cases*)
      (setf this-fitness-case (aref fitness-cases index))
      (setf x
              (POLYNOM-fitness-case-independent-variable-1
               this-fitness-case))
      (setf y
              (POLYNOM-fitness-case-independent-variable-2
               this-fitness-case))
      (setf target-value
              (POLYNOM-fitness-case-target
               this-fitness-case))
      (setf value-from-program
              (POLYNOM-wrapper (eval program)))
      (setf difference (abs (- target-value
                               value-from-program)))
      (incf raw-fitness difference)
      (when (< difference 0.01) (incf hits)))
    (setf standardized-fitness raw-fitness)
    (values standardized-fitness hits)))

(defun define-parameters-for-POLYNOM ()
  (setf *number-of-fitness-cases* 50)
  (setf *max-depth-for-new-individuals* 6)
  (setf *max-depth-for-individuals-after-crossover* 17)
  (setf *fitness-proportionate-reproduction-fraction* 0.1)
  (setf *crossover-at-any-point-fraction* 0.2)
  (setf *crossover-at-function-point-fraction* 0.2)
  (setf *max-depth-for-new-subtrees-in-mutants* 4)
  (setf *method-of-selection* :fitness-proportionate)
  (setf *method-of-generation* :ramped-half-and-half)
```

```
  (values))

(defun define-termination-criterion-for-POLYNOM
       (current-generation
        maximum-generations
        best-standardized-fitness
        best-hits)
  (declare (ignore best-standardized-fitness))
  (values
   (or (>= current-generation maximum-generations)
       (>= best-hits *number-of-fitness-cases*))))

(defun POLYNOM ()
  (values 'define-function-set-for-POLYNOM
          'define-terminal-set-for-POLYNOM
          'define-fitness-cases-for-POLYNOM
          'evaluate-standardized-fitness-for-POLYNOM
          'define-parameters-for-POLYNOM
          'define-termination-criterion-for-POLYNOM))
```

Bu programın çalıştırılabilmesi için, Little LISP sistemi derlenip Common Lisp çalışma zamanı ortamına yüklendikten sonra, programı içeren dosya aynı ortama derlenip yüklenmelidir. Çalıştırmak içinse, örneğin 17 rassal sayı üreteci için besleyici değeri, 20 maksimum kuşak sayısı ve 100 de popülasyon büyüklüğü olmak üzere Common Lisp ortamında izleyen komut verilmiştir:

```
CL-USER> (run-genetic-programming-system 'POLYNOM 17 20 100)
```

Bu örnek çalıştırma sonucunda, en iyi birey 14. kuşakta bulunmuştur. Standartlaştırılmış uyum fonksiyonu 3.82 olup 39 *isabet (hit)* değerine ulaştığı görülmüştür. Başka bir ifadeyle, sistemin bulduğu matematiksel modelin ürettiği tahminlerden 39'unun gerçek fonksiyon değerine çok yakın olduğu gözlenmiştir. Açıklamak gerekirse, 50 gözlem çiftinin neredeyse %80'inde bu bireyin matematiksel formunun ürettiği tahmin değerlerinin, hedef değerlerin $\pm$%0.01 çevresinde olduğu program tarafından rapor edilmektedir.

Sistem çalıştıktan sonra, genetik programın parametrelerine ve *uyum vakalarına (fitness case)* ilişkin aşağıdaki mesajı vermektedir:

```
 Parameters used for this run.
==============================
Maximum number of Generations:                  20
Size of Population:                             100
```

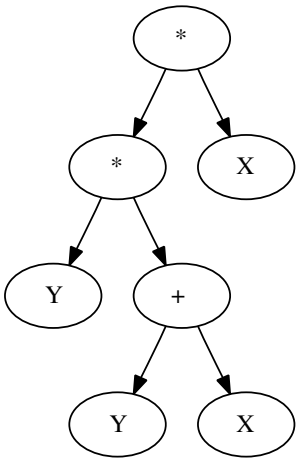
```
Maximum depth of new individuals:                6
Maximum depth of new subtrees for mutants:       4
Maximum depth of individuals after crossover:    17
Fitness-proportionate reproduction fraction:     0.1
Crossover at any point fraction:                 0.2
Crossover at function points fraction:           0.2
Number of fitness cases:                         50
Selection method:                                FITNESS-PROPORTIONATE
Generation method:                               RAMPED-HALF-AND-HALF
Randomizer seed:                                 17.0d0
Fitness cases
 0 0.0 0.0 0.0
 1 0.02 2000.0 80000.8
 2 0.04 4000.0 640006.4
 3 0.06 999.9998 60003.58
 4 0.08 2999.9998 720019.06
 5 0.1 0.0 0.0
 6 0.12 1999.9996 480028.63
 7 0.14 4000.0 2240078.5
 8 0.16 999.999 160025.28
 9 0.18 3000.0 1620097.3
 10 0.2 0.0 0.0
 11 0.22 1999.9996 880096.44
 12 0.24 3999.9993 3840229.0
 13 0.26 999.999 260067.06
 14 0.28 3000.0 2520235.3
 15 0.3 0.0 0.0
 16 0.32 1999.998 1280202.3
 17 0.34 3999.9993 5440460.5
 18 0.36 1000.002 360131.06
 19 0.38 3000.0 3420433.3
 20 0.4 0.0 0.0
 21 0.42 1999.998 1680349.4
 22 0.44 3999.9993 7040771.5
 23 0.46 999.999 460210.72
 24 0.48 2999.997 4320682.5
 25 0.5 0.0 0.0
 26 0.52 1999.998 2080536.5
 27 0.54 4000.0022 8641176.0
 28 0.56 999.999 560312.5
 29 0.58 2999.997 5220999.0
 30 0.6 0.0 0.0
 31 0.62 1999.998 2480764.0
 32 0.64 3999.996 1.0241618e7
 33 0.66 999.999 660434.4
 34 0.68 2999.997 6121375.0
 35 0.7 0.0 0.0
```

```
36 0.72 2000.004 2881048.5
37 0.74 4000.0022 1.1842203e7
38 0.76 999.999 760576.1
39 0.78 2999.997 7021811.0
40 0.8 0.0 0.0
41 0.82 1999.998 3281338.3
42 0.84 3999.996 1.3442795e7
43 0.86 999.999 860737.94
44 0.88 2999.997 7922307.0
45 0.9 -0.0059604645 -0.0047960016
46 0.92 1999.998 3681685.8
47 0.94 3999.996 1.5043505e7
48 0.96 999.999 960919.7
49 0.98 3000.0032 8822900.0
```

Bunu takiben ise her bir kuşakla ilgili istatistikleri, yani popülasyona ait standartlaştırılmış uyum fonksiyonu değeri ve söz konusu popülasyonun en iyi bireyine ait standartlaştırılmış uyum fonksiyonu değeriyle bu bireyin *"hit"* sayısını içeren bir mesaj vermektedir.

Sistemin ürettiği nihai mesaj ve bu mesajda yer alan bu *çalıştırmadaki (run)* en düşük uyum değeri fonksiyonuna sahip bireye ait matematiksel form aşağıdaki şekilde olup çalıştırma sonunda elde edilen bireyin ağaç şeklinde gösterimi Şekil A.1 'de verilmiştir.

```
The best-of-run individual program for this run was found on
generation 14 and had a standardized fitness measure of 3.8164063 and 39 hits.
It was:
(* (* Y (+ Y X)) X)
```



Şekil A.1: En İyi Bireyin (Programın) Ağaç Şeklinde Gösterimi

Sonuç olarak; oldukça düşük sayıdaki gözleme ve bağımsız değişkenler arasında hususi olarak yaratılan ölçek sorununa rağmen; GP, doğrusal olmayan bir yapıdaki bu veri

kümesine uygun olarak asıl polinomun matematiksel olarak eşdeğerini, hatta daha etkileyici bir biçimde sadeleştirilmiş halini bulmuştur. Bu basit deney gerçek hayat uygulamalarında doğrusal olmayan yapıya sahip veri kümelerine doğru uydurmada, GP yardımıyla pratikte başarılı olunabileceği ümidini vermektedir.

B. Finansal Veritabanına İlişkin Ekler

B.1. Şirket Bilgileri Veritabanını Oluşturan SQL Programı

```
BEGIN TRANSACTION;
CREATE TABLE Sector
       (ID INTEGER PRIMARY KEY,
        Name TEXT UNIQUE NOT NULL);

CREATE TABLE Market
       (ID INTEGER PRIMARY KEY,
        Name TEXT UNIQUE NOT NULL);

CREATE TABLE Company
       (ID INTEGER PRIMARY KEY,
        Name TEXT UNIQUE NOT NULL,
        SectorID INTEGER NOT NULL);
        -- FOREIGN KEY(SectorID) REFERENCES Sector(ID));

CREATE TABLE Auction
       (Type VARCHAR(1) PRIMARY KEY,
       MorningOpen TIME NOT NULL,
       MorningClose TIME NOT NULL,
       NoonOpen TIME NOT NULL,
       NoonClose TIME NOT NULL);

CREATE TABLE Indx
       (ID INTEGER PRIMARY KEY,
       Quote TEXT UNIQUE NOT NULL,
       Context TEXT NOT NULL DEFAULT 'Sectoral',
       HistDataTableName TEXT UNIQUE NOT NULL,
       RealTimeDataTableName TEXT UNIQUE NOT NULL);

CREATE TABLE YahooForeignIndx
       (ID INTEGER PRIMARY KEY,
       Quote TEXT UNIQUE NOT NULL,
       Comment TEXT UNIQUE NOT NULL,
       Region TEXT NOT NULL,
       Country TEXT NOT NULL,
       Composition TEXT NOT NULL,
```

```
        HistDataTableName TEXT UNIQUE NOT NULL,
        RealTimeDataTableName TEXT UNIQUE NOT NULL);

CREATE TABLE Stock
        (ID INTEGER PRIMARY KEY,
        Quote TEXT UNIQUE NOT NULL,
        CompanyID INTEGER NOT NULL,
        -- FOREIGN KEY(CompanyID) REFERENCES Company(ID),
        MarketID INTEGER NOT NULL,
        -- FOREIGN KEY(MarketID) REFERENCES Market(ID),
        AuctionType VARCHAR(1) NOT NULL DEFAULT 'A',
        Extension TEXT NOT NULL DEFAULT 'E',
        HistDataTableName TEXT UNIQUE NOT NULL,
        RealTimeDataTableName TEXT UNIQUE NOT NULL);

CREATE TABLE Stock_Indx
        (ID INTEGER PRIMARY KEY,
         StockID INTEGER NOT NULL,
         IndexID INTEGER NOT NULL);

CREATE TABLE Warrant
        (ID INTEGER PRIMARY KEY,
        Quote TEXT UNIQUE NOT NULL,
        IssuerCompanyID INTEGER NOT NULL,
        MarketID INTEGER NOT NULL,
        AuctionType VARCHAR(1) NOT NULL DEFAULT 'A',
        Extension TEXT NOT NULL DEFAULT 'V',
        HistDataTableName TEXT UNIQUE NOT NULL,
        RealTimeDataTableName TEXT UNIQUE NOT NULL,
        -- Below this line comes distinctive properties
        -- of warrants that do not exist for stocks.
        UnderlyingAssetID INTEGER NOT NULL DEFAULT 1,
        -- FOREIGN KEY(UnderlyingAssetID) REFERENCES WarrantableCommodity(ID),
        OptionCategory TEXT NOT NULL DEFAULT 'Call',
        OptionType TEXT NOT NULL DEFAULT 'European',
        Settlement TEXT NOT NULL DEFAULT 'Cash',
        LastTradingDate DATE NOT NULL DEFAULT 'YYYYMMDD',
        ExerciseDate DATE NOT NULL DEFAULT 'YYYYMMDD',
        StrikePrice REAL NOT NULL DEFAULT 0.0,
        Multiplier REAL NOT NULL DEFAULT 0.001);

CREATE TABLE WarrantableCommodity
        (ID INTEGER PRIMARY KEY,
         Quote TEXT UNIQUE NOT NULL);
COMMIT;

-- Create temporary tables for some housekeepin'
```

```
BEGIN TRANSACTION;
CREATE TABLE Temp_B (Quote TEXT UNIQUE NOT NULL);
CREATE TABLE Temp_C (Quote TEXT UNIQUE NOT NULL);
CREATE TABLE Temp_Magn (Quote TEXT UNIQUE NOT NULL);
CREATE TABLE Temp_Geo (Quote TEXT UNIQUE NOT NULL);
COMMIT;

-- Populate Auction table.
BEGIN TRANSACTION;
INSERT INTO Auction VALUES('A', '09:50:00', '12:30:00', '14:20:00', '17:30:00');
INSERT INTO Auction VALUES('B', '09:50:00', '12:30:00', '14:20:00', '17:30:00');
INSERT INTO Auction VALUES('C', '12:25:00', '12:30:00', '17:25:00', '17:30:00');
COMMIT;

-- Populate WarrantableCommodity table.
BEGIN TRANSACTION;
INSERT INTO WarrantableCommodity VALUES(NULL, 'XU030');
INSERT INTO WarrantableCommodity VALUES(NULL, 'GARAN');
INSERT INTO WarrantableCommodity VALUES(NULL, 'ISCTR');
INSERT INTO WarrantableCommodity VALUES(NULL, 'YKBNK');
INSERT INTO WarrantableCommodity VALUES(NULL, 'AKBNK');
INSERT INTO WarrantableCommodity VALUES(NULL, 'KCHOL');
INSERT INTO WarrantableCommodity VALUES(NULL, 'TCELL');
INSERT INTO WarrantableCommodity VALUES(NULL, 'THYAO');
INSERT INTO WarrantableCommodity VALUES(NULL, 'EKGYO');
INSERT INTO WarrantableCommodity VALUES(NULL, 'EREGL');
INSERT INTO WarrantableCommodity VALUES(NULL, 'USDTRY');
COMMIT;
```

B.2. Şirket Bilgileri Veritabanında Bazı Düzenlemeler Yapan Yardımcı SQL Programı

```
BEGIN TRANSACTION;

-- Seperate warrants from stocks and insert them into Warrant table.
-- Delete unnecessary rows from Stock table.
INSERT INTO Warrant
(ID, Quote, IssuerCompanyID, MarketID, HistDataTableName, RealTimeDataTableName)
SELECT ID, Quote, CompanyID, MarketID, HistDataTableName,
RealTimeDataTableName FROM Stock WHERE CompanyID IN
(SELECT ID FROM Company WHERE NAME LIKE "%varant%" OR NAME lIKE "%deutsche%");

DELETE FROM Stock WHERE CompanyID IN
(SELECT ID FROM Company WHERE NAME LIKE "%varant%" OR NAME lIKE "%deutsche%");

-- Update types of stocks which  have an AuctionType other than 'A'.
-- Those are supposed to be type 'B' and 'C'.
UPDATE Stock SET AuctionType='B' WHERE Quote IN (SELECT Quote FROM Temp_B);
UPDATE Stock SET AuctionType='C' WHERE Quote IN (SELECT Quote FROM Temp_C);

-- Drop temporary tables.
DROP TABLE Temp_B;
DROP TABLE Temp_C;

-- Update indexes, those of
-- which have Context other than "Sectoral"
UPDATE Indx SET Context="Magnitude" WHERE Quote IN (SELECT Quote FROM Temp_Magn);
UPDATE Indx SET Context="Geographic" WHERE Quote IN (SELECT Quote FROM Temp_Geo);

-- Drop temporary tables.
DROP TABLE Temp_Magn;
DROP TABLE Temp_Geo;

COMMIT;
```

B.3. Firefox iMacros Eklentisi İçin Makro Kodlarını Üreten Program

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# This script creates an iMacro script for Firefox
# in order to get company information from İMKB's (ISE-Istanbul Stock Exchange)
# site given as a tabbed table at the following URL:
# http://www.imkb.gov.tr/sirketler/sirketler.aspx

import string

IMACRO_SCRIPT_FILE = "get_IMKB_company_info.iim"
OUTPUT_FILE = "IMKB_company_info.txt"
PAUSE = 5 # Waiting duration.

letters = list(string.ascii_uppercase)
# These are non-Turkish chars and plus J,
# with which there is no company name begins with
non_turkish_chars = ['J', 'Q', 'W', 'X']
# These are special chars that have related Turkish
# sister chars also exist in alphabet
sc = ['C', 'I', 'O', 'S', 'U']

# Remove non Turkish characters from
# ascci letters set.
for i in non_turkish_chars:
    letters.remove(i)

# Insert placeholder characters into
# letters set, whose index is identified
# by specific_chars which are related to some chars
# specific to Turkish alphabet.

map(letters.insert,
    map((lambda x, y: x + y), map(letters.index, sc), xrange(1, sc.__len__()+1)),
    map((lambda x: x + '1'), sc))

# Companies that start with the letters in the below dictionary
# have been listed on multiple tab table.
# IMPORTANT! For now you should have to check ISE's site
# and update the below dictionary yourself.
d = { 'A':6, 'B':3, 'D':10, 'E':3, 'F':2, 'G':2,
      'I1':3, 'K':3, 'M':3, 'P':2, 'S':2, 'T':5}

# iMacro script variables
def put_header():
```

```
    return "VERSION BUILD=7021019 RECORDER=FX\nTAB T=1\n\
URL GOTO=http://www.imkb.gov.tr/sirketler/sirketler.aspx\n\
SET !EXTRACT_TEST_POPUP NO\n"

def put_extract_command():
    return "TAG POS=1 TYPE=TABLE ATTR=TXT:* EXTRACT=TXT\n"

def put_save_command(out_file):
    return "SAVEAS TYPE=EXTRACT FOLDER=* FILE=%s\n" % out_file

def put_wait_command(secs):
    return "WAIT SECONDS=%s\n" % secs

def change_main_tab(char, ismultipage):
    w=put_wait_command(PAUSE+1)
    s1="TAG POS=1 TYPE=A ATTR=ID:ctl00_cphContent_ctl00_lbtn%s\n" % char
    s1=s1+w
    s2="TAG POS=1 TYPE=A ATTR=TXT:1\n"
    if (ismultipage):
        s1=s1+'\n'+s2
    return s1

def change_child_tab(char):
    return "TAG POS=1 TYPE=SPAN ATTR=TXT:%s\n" % char

def put_common_tasks(a_file):
    a_file.write(put_wait_command(PAUSE))
    a_file.write(put_extract_command())
    a_file.write(put_save_command(OUTPUT_FILE))
    a_file.write('\n\n')

f = file(IMACRO_SCRIPT_FILE, 'w')
f.write(put_header())
f.close()

# Main loop
def main():
    f = file(IMACRO_SCRIPT_FILE, 'a')
    for i in letters:
        if i in d:
            f.write(change_main_tab(i, d[i]))
            put_common_tasks(f)
            for j in xrange(d[i]-1):
                f.write(change_child_tab(j+2))
                put_common_tasks(f)
        else:
            f.write(change_main_tab(i, False))
```

```
            put_common_tasks(f)
    f.close()

# Call main loop
main()
```

B.4. Şirket Bilgilerinin Veritabanına Doğrudan Okunulabilmesini Sağlayan Program

```
#!/usr/bin/env python
import sys
from my_utility import finalize_abc_list
sys.path.append('/home/thorin/subversion/oican/repos/trunk/coding/Python')
from  replace_turkish_chars import *

DIR="/home/thorin/iMacros/Downloads/"
FILE=DIR+"IMKB_company_info.txt"

f = open(FILE, 'r')
s=f.readlines()
f.close()

s=s[1:]

new_list=[]
for i in s:
  if i.isspace():
    None
  elif '\r\n' in i:
    i=i.replace('\r\n', '')
    new_list.append(i)
  elif '\n' in i:
    i=i.replace('\n', '')
    new_list.append(i)
  else:
    new_list.append(i)

i=0
for i in xrange(len(new_list)):
  new_list[i]=replace_turkish_chars(new_list[i])
  i+=1

s=new_list
l=len(s)
out=[]
i=0

while(i < l):
  sub1=[]
  j = 0
  while(j < 4):
    sub1.append(s[i])
```

```
    i+=1
    j+=1
  sub2=[]
  while((i < l) and (s[i][0] is 'X')):
    sub2.append(s[i])
    i+=1
  out.append(sub1+sub2)

# Dump data to files
# according to columns
companies=[]
stocks=[]
markets=[]
sectors=[]
indexes=[]
for i in out:
  companies.append(i[0])
  stocks.append(i[1])
  markets.append(i[2])
  sectors.append(i[3])
  indexes.extend(i[4:])

# Remove duplicates
s_companies=set(companies)
s_stocks=set(stocks)
s_markets=set(markets)
s_sectors=set(sectors)
s_indexes=set(indexes)

# Back to lists removed from duplicates and sorted
companies=list(s_companies)
stocks=list(s_stocks)
markets=list(s_markets)
sectors=list(s_sectors)
indexes=list(s_indexes)

# After this line come the procedures
# which composes necessary text files under "data_files" folder
# to be imported directly into the sqlite database.
f=open('./data_files/markets.txt', 'w')
markets.sort()
i=0
for i in xrange(len(markets)):
  f.write('%s:' % (i+1))
  f.write(markets[i])
  f.write('\n')
  i+=1
```

```
f.close()

f=open('./data_files/sectors.txt', 'w')
i=0
sectors.sort()
for i in xrange(len(sectors)):
  f.write('%s:' % (i+1))
  f.write(sectors[i])
  f.write('\n')
  i+=1
f.close()

def retrieve(key, target_lst):
  return target_lst.index(key)

f=open('./data_files/companies.txt', 'w')
i=0
companies.sort()
for i in xrange(len(companies)):
  f.write('%s:' % (i+1))
  f.write(companies[i]+':')
  for j in out:
    if(companies[i]==j[0]):
      f.write('%s' % (retrieve(j[3], sectors) + 1))
      break
  f.write('\n')
  i+=1
f.close()

f=open('./data_files/stocks.txt', 'w')
i=0
stocks.sort()
for i in xrange(len(stocks)):
  f.write('%s:' % (i+1))
  f.write(stocks[i]+':')
  for j in out:
    if(stocks[i]==j[1]):
      f.write('%s:' % (retrieve(j[0], companies) + 1))
      f.write('%s:' % (retrieve(j[2], markets) + 1))
      break
  f.write('A:E:%s' % (stocks[i]+'_HT:'+stocks[i]+'_RT\n'))
  i+=1
f.close()

f=open('./data_files/indexes.txt', 'w')
i=0
indexes.sort()
```

```
for i in xrange(len(indexes)):
  f.write('%s:' % (i+1))
  f.write(indexes[i]+':Sectoral:')
  f.write('%s' % (indexes[i]+'_HT:'+indexes[i]+'_RT\n'))
  i+=1
f.close()

f=open('./data_files/stock_indx.txt', 'w')
i=0
counter=0
while(i < len(stocks)):
  for j in out:
    if(stocks[i]==j[1] and not(len(j[4:])==0)):
      for k in j[4:]:
        f.write('%s:%s:' % (counter+1, i+1))
        f.write('%s\n' % (retrieve(k, indexes)+1))
        counter+=1
  i+=1
f.close()

t=finalize_abc_list() # Tuple contains name of stocks that fall under C and B list.
f=open('./data_files/c_list.txt', 'w') # C list
for i in t[0]:
  f.write('%s\n' % i)
f.close()
f=open('./data_files/b_list.txt', 'w') # B list
for i in t[1]:
  f.write('%s\n' % i)
f.close()
```

B.5. Şirket Bilgileri Veritabanının Oluşturan Kabuk Betiği

```
#!/bin/bash
[ $# != 1 ] && echo "Usage is: $0 filename" && exit 0
SQL_SOURCE_FILE="info.sql"
DOT_COMMANDS_FILE="info_dot_commands.sqlite3"
UPDATE_FILE="info_update.sql"
FILE=$1

function newdb() {
    sqlite3 $FILE < $SQL_SOURCE_FILE
    sqlite3 $FILE <<EOF
.read $DOT_COMMANDS_FILE
.read $UPDATE_FILE
EOF
#    sqlite3 -init $DOT_COMMANDS_FILE $FILE
}


function existing() {
    rm $FILE
    newdb
}

if [ -e $FILE ]; then
    echo "File named \"$FILE\" exists."
    read -p "Do you want to replace the existing SQLite3 DB file? (y/n)"
    if [ $REPLY == 'y' ]; then
        existing
    elif [ $REPLY == 'n' ]; then
        echo "Exiting without any transactions."
    else
        echo "Try again with sensible answers to questions."
    fi
else
    echo "File named \"$FILE\" does NOT exist."
    read -p "Do you want to create a new SQLite3 DB file? (y/n)"
    if [ $REPLY == 'y' ]; then
        newdb
    elif [ $REPLY == 'n' ]; then
        echo "Exiting without any transactions."
    else
        echo "Try again with sensible answers to questions."
    fi
fi

exit 0
```

B.6. Tarihi ve Diğer İlgili Verileri MynetFinans Portalından Elde Eden Rutinleri Sağlayan Python Modülü

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
#  Copyright (c) 2010, Özgür İcan (ozgurican@gmail.com)
#
#  license: GNU GPLv3
#
#  This library is Free Software; you can redistribute it and/or
#  modify it under the terms of the GNU General Public
#  License as published by the Free Software Foundation; either
#  GPLv3 or later versions.
#
#  This is "mynetfinans" module.
#
#  This module provides access to Mynet Finans
#  the Turkish counterpart of Yahoo! Finance.
#  It is inspired from Corey Goldberg's "ystockquote"
#  Python module which retrieves stock data from
#  Yahoo! Finance.

import urllib as U
import numpy as N
import string as S
import re

HIST_MAIN_DIR = "http://teknikanaliz.mynet.com/pages/lastdatacsv.csv?"
DAIL_MAIN_DIR = "http://finans.mynet.com/hisse/"
IND_HIST_MAIN_DIR = HIST_MAIN_DIR
IND_DAIL_MAIN_DIR = "http://finans.mynet.com/endeksler.aspx"

turkish_chars = {'\xc4\xb0' : 'I',
                 '\xc3\xbc' : 'u',
                 '\xc3\x9c' : 'U',
                 '\xc5\x9e' : 'S'}

def historical_data_url(symbol, start_date, end_date):
    """
    Compose URL of Mynet Finans in order to
    get historical prices for the given
    'symbol' between "start_date"
    and "end_date" given in the 'YYYYMMDD' form.
    """
    return (HIST_MAIN_DIR + \
```

```
            "&tar1=%s" % start_date + \
            "&tar2=%s" % end_date + \
            "&sembol=%s" % symbol)

def get_historical_data(symbol, start_date, end_date):
    """
    Get historical prices for the given
    commodity ticker symbol between "start_date"
    and "end_date" given in the 'YYYYMMDD' form.

    Returns a nested list of daily data.
    """
    url=historical_data_url(symbol, start_date, end_date)
    days=U.urlopen(url).readlines()
    data=[day[:-2].split(',') for day in days]
    return data

def pull_daily_data(symbol):
    """
    Pull daily stock data plus company info
    from Mynet Finans. Slice the whole source
    of relevant web page in a managable way.
    This result slice includes both company info and
    last trade price.

    Returns only necessary part of the page's source
    as a string. This will be the main input
    for the functions:

    "get_daily_data", "get_stock_properties" and "get_weekly_data".
    """
    url=DAIL_MAIN_DIR + symbol
    url_object=U.urlopen(url)
    raw_string=url_object.read()
    url_object.close()
    first_anchor = 'ctl00_IDMenu_DWStockDetail'
    second_anchor = '>Haberler'
    subind1 = raw_string.find(first_anchor)
    subind2 = raw_string.find(second_anchor)
    return raw_string[subind1:subind2]

def retrieve_num_chars_ahead_for_once(s, i):
    empty=''
    while(not(s[i].isdigit() or s[i]=='-')):
        i+=1
    while(s[i].isdigit() or s[i]==',' or s[i]=='-'):
        empty+=s[i]
```

```
        i+=1
    return empty

def get_daily_data(raw_string):
    """
    Disassembly the string returned by "pull_daily_data"
    and returns them as a dict containing DAILY data.
    """
    identifiers=['Son', 'Fark %', 'En D&#252;\xc5\x9f&#252;k',
                 'En Y&#252;ksek', 'Ortalama', '&#214;nceki G&#252;n']
    positions = map(raw_string.find, identifiers)
    positions = N.array(positions) + N.array(map(len, identifiers))
    values=map(retrieve_num_chars_ahead_for_once,
               [raw_string]*identifiers.__len__(), positions)
    values=map(S.atof, (S.replace( i, ',', '.') for i in values))
    out_tags=['LastPrice', 'Diff%', 'Low', 'High',
              'Average', 'PreviousDayClose']
    return dict(zip(out_tags,values))

def get_weekly_data(raw_string):
    """
    Disassembly the string returned by "pull_daily_data"
    and returns them as a dict containing WEEKLY data.
    """
    identifiers=['Haftal\xc4\xb1k En Y&#252;ksek',
                 'Haftal\xc4\xb1k En D&#252;\xc5\x9f&#252;k',
                 'Aylik En Y&#252;ksek', 'Ayl\xc4\xb1k D&#252;\xc5\x9f&#252;k',
                 'Y\xc4\xb1ll\xc4\xb1k En Y&#252;ksek',
                 'Y\xc4\xb1ll\xc4\xb1k En D&#252;\xc5\x9f&#252;k']
    positions = map(raw_string.find, identifiers)
    positions = N.array(positions) + N.array(map(len, identifiers))
    values=map(retrieve_num_chars_ahead_for_once,
               [raw_string]*identifiers.__len__(), positions)
    values=map(S.atof, (S.replace( i, ',', '.') for i in values))
    out_tags=['WeeklyHigh', 'WeeklyLow', 'MonthlyHigh',
              'MonthlyLow', 'YearlyHigh', 'YearlyLow']
    return dict(zip(out_tags,values))

def get_stock_properties(raw_string):
    """
    Returns a list of index names
    which the stock belongs to.
    """
    first_anchor = "id123"
    second_anchor = "ctl00_IDMenu_DWStockDetailSumm"
    subind1 = raw_string.find(first_anchor)
    subind2 = raw_string.find(second_anchor)
```

```
sub_string = raw_string[subind1:subind2]
prop=re.findall('">.*</span>',  sub_string)
for i in xrange(len(prop)):
    prop[i] = prop[i].strip('"</span>')
    for j in turkish_chars.keys():
        prop[i]=prop[i].replace(j, turkish_chars[j])
return prop
```

B.7. Geçmiş Günlük Fiyatlar Veritabanını Oluşturan Python Programı

```
#!/usr/bin/env python

# This is hist_create_db.py
# This script will create historical prices databases
# for local stocks, local indices and foreign indices
# for a fixed period between START and END dates.

from mynetfinans import *
from my_utility import *
import getopt, sys, sqlite3, os
import ystockquote as Y

# Info DB
INFO_DB='info.db'

# Historical prices database for local stocks, local
# indices and foreign indices.
HP_LS_DB='hp_ls.db'
HP_LI_DB='hp_li.db'
HP_FI_DB='hp_fi.db'

# Start and end dates for creating databases.
START='20000101'
END='20110625'

info_conn = sqlite3.connect(INFO_DB)
info_curs = info_conn.cursor()

hp_ls_conn = sqlite3.connect(HP_LS_DB)
hp_ls_curs = hp_ls_conn.cursor()

hp_li_conn = sqlite3.connect(HP_LI_DB)
hp_li_curs = hp_li_conn.cursor()

hp_fi_conn = sqlite3.connect(HP_FI_DB)
hp_fi_curs = hp_fi_conn.cursor()

# Populator function for historical data
# database.
def populate_mynet(infodb_cursor, histdb_cursor):
    for row in infodb_cursor:
        # Create tables
        print "Creating table %s ..." % row[1]
        histdb_cursor.execute('CREATE TABLE %s (Day DATE, Open REAL, High REAL,\
                              Low REAL, Close REAL, Volume INTEGER)' % row[1])
```

```
            print "Fetching data for %s..." % row[0]
            lst=get_historical_data(row[0], START, END)
            case=invalid_mynet_p(lst)
            if(case is 1):
                print "THERE IS NO HISTORICAL DATA ON MYNET SERVERS FOR THE GIVEN\
                PERIOD FOR %s!" % row[0]
                continue
            elif(case is -1):
                print "Populating table %s ..." % row[1]
                lst.pop(0) # The first element is empty list. Remove it.
                print "Starting to insert data into table %s." % row[1]
                for i in lst:
                    i=helper_mynet(i)
                for t in lst:
                    with_table_name='INSERT INTO %s VALUES(?, ?, ?, ?, ?, ?)' % row[1]
                    histdb_cursor.execute(with_table_name, t)
                print "Done..."
            else:
                print "THERE IS AN UNKNOWN ERROR OCCURRED WHILE TRYING TO\
                FETCH DATA FOR %s" % row[0]
                continue

def populate_yahoo(infodb_cursor, histdb_cursor):
    for row in infodb_cursor:
        # Create tables
        print "Creating table %s ..." % row[1]
        histdb_cursor.execute('CREATE TABLE %s (Day DATE, Open REAL, High REAL, Low REAL,\
                              Close REAL, Volume INTEGER, AdjClose REAL)' % row[1])
        print "Fetching data for %s..." % row[0]
        lst = Y.get_historical_prices(row[0], START, END)
        case=invalid_yahoo_p(lst)
        if(case is 1):
            print "THERE IS NO HISTORICAL DATA ON YAHOO SERVERS FOR THE\
            GIVEN PERIOD FOR %s!" % row[0]
            continue
        elif(case is 2):
            print "YAHOO RETURNED 404 ERROR! BYPASSING %s" % row[0]
            continue
        elif(case is -1):
            print "Populating table %s ..." % row[1]
            lst.pop(0) # The first element contains headers. Remove it.
            lst.reverse()
            print "Starting to insert data for %s." % row[0]
            for i in lst:
                i=helper_yahoo(i)
            for t in lst:
                with_table_name='INSERT INTO %s VALUES(?, ?, ?, ?, ?, ?, ?)' % row[1]
```

```
                histdb_cursor.execute(with_table_name, t)
            print "Done..."
        else:
            print "THERE IS AN UNKNOWN ERROR OCCURRED WHILE TRYING TO\
            FETCH DATA FOR %s" % row[0]
            continue

def populate(infodb_cursor, histdb_cursor, source):
    if(source=="mynet"):
        populate_mynet(infodb_cursor, histdb_cursor)
    elif(source=="yahoo"):
        populate_yahoo(infodb_cursor, histdb_cursor)
    else:
        print("Select an appropriate data source.")
        sys.exit(1)

# Create tables for Stocks that are in XU100 index,
# for  all IMKB indices and foreign indices and populate them
local_stocks=info_curs.execute('''SELECT Quote, HistDataTableName
FROM Stock WHERE ID IN (SELECT StockID FROM Stock_Indx
WHERE IndexID IN (SELECT ID FROM Indx WHERE Quote='XU100'))''')
populate(local_stocks, hp_ls_curs, "mynet")

local_indices=info_curs.execute('''SELECT Quote, HistDataTableName
FROM Indx''')
populate(local_indices, hp_li_curs, "mynet")

foreign_indices=info_curs.execute('''SELECT Quote, HistDataTableName
FROM YahooForeignIndx''')
populate(foreign_indices, hp_fi_curs, "yahoo")

info_conn.commit()
info_curs.close()

hp_ls_conn.commit()
hp_ls_curs.close()

hp_li_conn.commit()
hp_li_curs.close()

hp_fi_conn.commit()
hp_fi_curs.close()
```

B.8. Geçmiş Günlük Fiyatlar Veritabanını Güncelleyen Python Programı

```
#!/usr/bin/env python

# This is hist_update_db.py
# This script will update historical prices databases
# created with hist_create_db.py

from mynetfinans import *
from my_utility import *
import sqlite3, datetime
import ystockquote as Y

# Info DB
INFO_DB='info.db'

# Historical prices database for local stocks, local
# indices and foreign indices.
HP_LS_DB='hp_ls.db'
HP_LI_DB='hp_li.db'
HP_FI_DB='hp_fi.db'

# Just in case if there is empty tables we need an
# initial start date for update procedures because
# they can not take that info from database.
DEFAULT_START_DATE='20000101'

info_conn = sqlite3.connect(INFO_DB)
info_curs = info_conn.cursor()

hp_ls_conn = sqlite3.connect(HP_LS_DB)
hp_ls_curs = hp_ls_conn.cursor()

hp_li_conn = sqlite3.connect(HP_LI_DB)
hp_li_curs = hp_li_conn.cursor()

hp_fi_conn = sqlite3.connect(HP_FI_DB)
hp_fi_curs = hp_fi_conn.cursor()

# Some necessary functions
def convert_to_date(s):
    if(s is None):
        # Just in case if there are empty tables in database
        return convert_to_date(DEFAULT_START_DATE)
    else:
        s=str(s) # Make sure it's a string not int.
        y=int(s[0:4])
```

```
        m=int(s[4:6])
        d=int(s[6:9])
        return datetime.date(y,m,d)

def immediate_weekend_p(possible_friday, supposedly_weekend_day):
    td=(supposedly_weekend_day - possible_friday)
    if((possible_friday.isoweekday()==5) and (td.days <= 2)):
        return True
    else:
        return False

def need_for_update_p(local_db_date):
    today=local_db_date.today()
    if(local_db_date==today):
        return False
    elif(immediate_weekend_p(local_db_date, today)):
        return False
    else:
        return True

def time_for_update_p(local_date, source):
    today=local_date.today()
    if(source=="mynet"):
        dt=datetime.datetime(today.year, today.month, today.day, 22)
        now=dt.now()
        current_hour = now.hour
        # Mynet generally updates its servers 10 P.M. (22) and
        # the difference between local db date and current date usually
        # becomes 1 day until it updates its servers.
        if((today-local_date==datetime.timedelta(1))
           and ( current_hour < dt.hour )):
            return False
        else:
            return True
    elif(source=="yahoo"):
        dt=datetime.datetime(today.year, today.month, today.day, 4)
        now=dt.now()
        current_hour = now.hour
        # Yahoo generally updates its servers 4 A.M. and
        # the difference between local db date and current date usually
        # becomes 2 days until it updates its servers.
        # Thus during this time there's o need for update
        if((today-local_date==datetime.timedelta(2))
           and ( current_hour < dt.hour )):
            return False
        elif(today-local_date==datetime.timedelta(1)):
            return False
```

```
        else:
            return True
    else:
        print "time_for_update_p did not recognize this source...\n"



def update_mynet(infodb_cursor, histdb_cursor):
    for row in infodb_cursor:
        # Update tables
        print "Attempting to update table %s ..." % row[1]
        date=histdb_cursor.execute('''SELECT Max(Day) FROM %s''' % row[1])
        t=date.fetchone() # tuple
        ld=convert_to_date(t[0]) # local date
        if(need_for_update_p(ld) and time_for_update_p(ld, "mynet")):
            today=ld.today()
            tda_ld=ld+datetime.timedelta(1)# the day after the local date
            tda_ld_str=tda_ld.strftime("%Y%m%d") # string representation of tda_ld
            today_str=today.strftime("%Y%m%d") # string representation of today
            print "There is a need for update %s" % row[0]
            print "Fetching data for %s ..." % row[0]
            lst=get_historical_data(row[0], tda_ld_str, today_str)
            case=invalid_mynet_p(lst)
            if(case is 1):
                print "THERE IS NO HISTORICAL DATA ON MYNET SERVERS\
                FOR THE GIVEN PERIOD FOR %s!\n" % row[0]
                continue
            elif(case is -1):
                print "Updating table %s ..." % row[1]
                lst.pop(0) # The first element is empty list. Remove it.
                print "Starting to insert data into table %s." % row[1]
                for i in lst:
                    i=helper_mynet(i)
                for t in lst:
                    with_table_name='INSERT INTO %s VALUES(?, ?, ?, ?, ?, ?)' % row[1]
                    histdb_cursor.execute(with_table_name, t)
                print "Done...\n"
            else:
                print "THERE IS AN UNKNOWN ERROR OCCURRED WHILE\
                TRYING TO FETCH DATA FOR %s\n" % row[0]
                continue
        else:
            print "%s is up to date ..." % row[0]

def update_yahoo(infodb_cursor, histdb_cursor):
    for row in infodb_cursor:
```

```
            # Update tables
            print "Attempting to update table %s ..." % row[1]
            date=histdb_cursor.execute('''SELECT Max(Day) FROM %s''' % row[1])
            t=date.fetchone() # tuple
            ld=convert_to_date(t[0]) # local date
            if(need_for_update_p(ld) and time_for_update_p(ld, "yahoo")):
                today=ld.today()
                tda_ld=ld+datetime.timedelta(1)# the day after the local date
                tda_ld_str=tda_ld.strftime("%Y%m%d") # string representation of tda_ld
                today_str=today.strftime("%Y%m%d") # string representation of today
                print "There is a need for update %s" % row[0]
                print "Fetching data for %s ..." % row[0]
                lst=Y.get_historical_prices(row[0], tda_ld_str, today_str)
                case=invalid_yahoo_p(lst)
                if(case is 1):
                    print "THERE IS NO HISTORICAL DATA ON YAHOO SERVERS\
                    FOR THE GIVEN PERIOD FOR %s\n!" % row[0]
                    continue
                elif(case is 2):
                    print "YAHOO RETURNED 404 ERROR! BYPASSING %s\n" % row[0]
                    continue
                elif(case is -1):
                    print "Updating table %s ..." % row[1]
                    lst.pop(0) # The first element contains headers. Remove it.
                    lst.reverse()
                    print "Starting to insert data for %s." % row[0]
                    for i in lst:
                        i=helper_yahoo(i)
                    for t in lst:
                        with_table_name='INSERT INTO %s VALUES\
                        (?, ?, ?, ?, ?, ?, ?)' % row[1]
                        histdb_cursor.execute(with_table_name, t)
                    print "Done...\n"
                else:
                    print "THERE IS AN UNKNOWN ERROR OCCURRED WHILE\
                    TRYING TO FETCH DATA FOR %s\n" % row[0]
                    continue
            else:
                print "%s is up to date ..." % row[0]


def update(infodb_cursor, histdb_cursor, source):
    if(source=="mynet"):
        update_mynet(infodb_cursor, histdb_cursor)
    elif(source=="yahoo"):
        update_yahoo(infodb_cursor, histdb_cursor)
    else:
```

```
        print("Select an appropriate data source.")
        sys.exit(1)

local_stocks=info_curs.execute('''SELECT Quote,
HistDataTableName FROM Stock
WHERE ID IN (SELECT StockID FROM Stock_Indx
WHERE IndexID IN (SELECT ID FROM Indx WHERE Quote='XU100'))''')
update(local_stocks, hp_ls_curs, "mynet")

local_indices=info_curs.execute('''SELECT Quote,
HistDataTableName FROM Indx''')
update(local_indices, hp_li_curs, "mynet")

foreign_indices=info_curs.execute('''SELECT Quote,
HistDataTableName FROM YahooForeignIndx''')
update(foreign_indices, hp_fi_curs, "yahoo")

info_conn.commit()
info_curs.close()

hp_ls_conn.commit()
hp_ls_curs.close()

hp_li_conn.commit()
hp_li_curs.close()

hp_fi_conn.commit()
hp_fi_curs.close()
```

C. Finansal Zaman Serilerinde GP Kullanımına İlişkin Ekler

C.1. Garanti Bankası A.Ş. Hisse Senedi Sembolik Regresyon Uygulaması Kaynak Kodları

```
;;;; icll/gp/examples/symreg.lisp

(in-package :icll.gp.examples.symreg)

(defparameter *ls-db* (connect
                       '( "/home/thorin/subversion/oican/repos/trunk/coding/Python/financial_db/hp_ls.db")
                               :pool t :database-type :sqlite3))
(defparameter *li-db* (connect
                       '( "/home/thorin/subversion/oican/repos/trunk/coding/Python/financial_db/hp_li.db")
                               :pool t :database-type :sqlite3))

(defun to-array(l)
  (make-array (length l) :initial-contents (mapcar #'car l)))

(execute-command
"ATTACH '/home/thorin/subversion/oican/repos/trunk/coding/Python/financial_db/hp_ls.db' AS LS;"
:database *li-db*)

(execute-command
 "CREATE TABLE TEMP_GRN_TIMELINE AS SELECT Day FROM NAT_TIMELINE INTERSECT SELECT Day FROM LS.GARAN_HT;"
 :database *li-db*)

(defvar *garan* (to-array (query "select Close from GARAN_HT where Day in TEMP_GRN_TIMELINE"
                                 :database *li-db*)))
(defvar *xu030* (to-array (query "select Close from XU030_HT where Day in TEMP_GRN_TIMELINE"
                                 :database *li-db*)))
(defvar *xu100* (to-array (query "select Close from XU100_HT where Day in TEMP_GRN_TIMELINE"
                                 :database *li-db*)))

(execute-command
 "DROP TABLE TEMP_GRN_TIMELINE;" :database *li-db*)

(disconnect :database *ls-db*)
(disconnect :database *li-db*)
```

```
(defvar garan)
(defvar xu030)
(defvar xu100)

(defun define-terminal-set-for-SYMREG ()
  (values '(garan xu030 xu100 :floating-point-random-constant)))

(defun define-function-set-for-SYMREG ()
  (values '(+ - * %)
          '(2 2 2 2)))

(defun % (numerator denominator)
  "The Protected Division Function"
  (values (if (= 0 denominator) 1 (/ numerator denominator))))

(defstruct SYMREG-fitness-case
  independent-variable-1 ; garan
  independent-variable-2 ; xu030
  independent-variable-3 ; xu100
  target) ; garan_(t+1)

(defun define-fitness-cases-for-SYMREG ()
  (let (fitness-cases garan xu030 xu100 this-fitness-case)
    (setf fitness-cases (make-array *number-of-fitness-cases*))
    (format t "~%Fitness cases")
    (dotimes (index *number-of-fitness-cases*)
      (setf garan (aref *garan* index))
      (setf xu030 (aref *xu030* index))
      (setf xu100 (aref *xu100* index))
      (setf this-fitness-case (make-SYMREG-fitness-case))
      (setf (aref fitness-cases index) this-fitness-case)
      (setf (SYMREG-fitness-case-independent-variable-1 this-fitness-case) garan)
      (setf (SYMREG-fitness-case-independent-variable-2 this-fitness-case) xu030)
      (setf (SYMREG-fitness-case-independent-variable-3 this-fitness-case) xu100)
      (setf (SYMREG-fitness-case-target this-fitness-case) (aref *garan* (1+ index)))
      (format t "~% ~D ~D ~D ~D ~D"
              index
              (float garan)
              (float xu030)
              (float xu100)
              (SYMREG-fitness-case-target this-fitness-case)))
    (values fitness-cases)))

(defun SYMREG-wrapper (result-from-program)
  (values result-from-program))
```

```
(defun evaluate-standardized-fitness-for-SYMREG  (program fitness-cases)
  (let ((raw-fitness 0.0)
        (hits 0)
        standardized-fitness
        garan
        xu030
        xu100
        target-value
        difference
        value-from-program
        this-fitness-case)
    (dotimes (index *number-of-fitness-cases*)
      (setf this-fitness-case (aref fitness-cases index))
      (setf garan
              (SYMREG-fitness-case-independent-variable-1
               this-fitness-case))
      (setf xu030
              (SYMREG-fitness-case-independent-variable-2
               this-fitness-case))
      (setf xu100
              (SYMREG-fitness-case-independent-variable-3
               this-fitness-case))
      (setf target-value
              (SYMREG-fitness-case-target
               this-fitness-case))
      (setf value-from-program
              (SYMREG-wrapper (eval program)))
      (setf difference (abs (- target-value
                               value-from-program)))
      (incf raw-fitness difference)
      (when (< difference 0.01) (incf hits)))
    (setf standardized-fitness raw-fitness)
    (values standardized-fitness hits)))

(defun define-parameters-for-SYMREG ()
  (setf *number-of-fitness-cases* 1000)
  (setf *max-depth-for-new-individuals* 7)
  (setf *max-depth-for-new-subtrees-in-mutants* 4)
  (setf *max-depth-for-individuals-after-crossover* 17)
  (setf *fitness-proportionate-reproduction-fraction* 0.1)
  (setf *crossover-at-any-point-fraction* 0.2)
  (setf *crossover-at-function-point-fraction* 0.2)
  (setf *method-of-selection* :fitness-proportionate)
  (setf *method-of-generation* :ramped-half-and-half)
  (values))

(defun define-termination-criterion-for-SYMREG
```

```
        (current-generation
         maximum-generations
         best-standardized-fitness
         best-hits)
  (declare (ignore best-standardized-fitness))
  (values
   (or (>= current-generation maximum-generations)
       (>= best-hits *number-of-fitness-cases*))))

(defun SYMREG ()
  (values 'define-function-set-for-SYMREG
          'define-terminal-set-for-SYMREG
          'define-fitness-cases-for-SYMREG
          'evaluate-standardized-fitness-for-SYMREG
          'define-parameters-for-SYMREG
          'define-termination-criterion-for-SYMREG))
```

C.2. Ulusal 30 Endeksi Sembolik Regresyon Uygulaması Kaynak Kodları

```
;;;; icll/gp/examples/symreg_u30.lisp
;;;; Written by Ozgur ICAN

(in-package :icll.gp.examples.symreg_u30)

(defparameter *li-db* (connect
                       '( "/home/thorin/subversion/oican/repos/trunk/coding/Python/financial_db/hp_li.db")
                       :pool t :database-type :sqlite3))
(defparameter *fi-db* (connect
                       '( "/home/thorin/subversion/oican/repos/trunk/coding/Python/financial_db/hp_fi.db")
                       :pool t :database-type :sqlite3))

(defun to-array(l)
  (make-array (length l) :initial-contents (mapcar #'car l)))

(execute-command
"ATTACH '/home/thorin/subversion/oican/repos/trunk/coding/Python/financial_db/hp_fi.db' AS FI;"
:database *li-db*)

(execute-command
 "CREATE TABLE TEMP_UNI_TIMELINE AS SELECT Day FROM NAT_TIMELINE INTERSECT SELECT Day FROM FI.INT_TIMELINE;"
 :database *li-db*)

(defvar *xu030* (to-array
                 (query "select Close from XU030_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
(defvar *xu100* (to-array
                 (query "select Close from XU100_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
(defvar *n225*  (to-array
                 (query "select Close from FI.N225_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
(defvar *aord*  (to-array
                 (query "select Close from FI.AORD_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
(defvar *gdaxi* (to-array
                 (query "select Close from FI.GDAXI_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
(defvar *fchi*  (to-array
                 (query "select Close from FI.FCHI_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
(defvar *ftse*  (to-array
                 (query "select Close from FI.FTSE_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
(defvar *dji*   (to-array
                 (query "select Close from FI.DJI_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
(defvar *ixic*  (to-array
                 (query "select Close from FI.IXIC_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
(defvar *gspc*  (to-array
                 (query "select Close from FI.GSPC_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
(defvar *bvsp*  (to-array
                 (query "select Close from FI.VSP_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
```

```
(defvar *xoi*   (to-array
                 (query "select Close from FI.XOI_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
(defvar *xau*   (to-array
                 (query "select Close from FI.XAU_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))
(defvar *vix*   (to-array
                 (query "select Close from FI.VIX_HT where Day in TEMP_UNI_TIMELINE" :database *li-db*)))

(execute-command
 "DROP TABLE TEMP_UNI_TIMELINE;" :database *li-db*)

(disconnect :database *li-db*)
(disconnect :database *fi-db*)

(defvar xu030)
(defvar xu100)
(defvar n225)
(defvar aord)
(defvar hsi)
(defvar bsesn)
(defvar gdaxi)
(defvar fchi)
(defvar ftse)
(defvar dji)
(defvar ixic)
(defvar gspc)
(defvar bvsp)
(defvar xoi)
(defvar xau)
(defvar vix)

(defun define-terminal-set-for-SYMREG_U30 ()
  (values '(xu030
            xu100
            n225
            aord
            hsi
            bsesn
            gdaxi
            fchi
            ftse
            dji
            ixic
            gspc
            bvsp
            xoi
            xau
            vix
```

```
            :floating-point-random-constant)))

(defun define-function-set-for-SYMREG_U30 ()
  (values '(+ - * %)
          '(2 2 2 2)))

(defun % (numerator denominator)
  "The Protected Division Function"
  (values (if (= 0 denominator) 1 (/ numerator denominator))))

(defstruct SYMREG_U30-fitness-case
  var-1 ;  xu030
  var-2 ;  xu100
  var-3 ;  n225
  var-4 ;  aord
  var-5 ;  hsi
  var-6 ;  bsesn
  var-7 ;  gdaxi
  var-8 ;  fchi
  var-9 ;  ftse
  var-10 ; dji
  var-11 ; ixic
  var-12 ; gspc
  var-13 ; bvsp
  var-14 ; xoi
  var-15 ; xau
  var-16 ; vix
  target) ; xu030_(t+1)

(defun define-fitness-cases-for-SYMREG_U30 ()
  (let (fitness-cases
        xu030
        xu100
        n225
        aord
        hsi
        bsesn
        gdaxi
        fchi
        ftse
        dji
        ixic
        gspc
        bvsp
        xoi
        xau
        vix
```

```
        this-fitness-case)
    (setf fitness-cases (make-array *number-of-fitness-cases*))
    (format t "~%Fitness cases")
    (dotimes (index *number-of-fitness-cases*)
      (setf xu030 (aref *xu030* index))
      (setf xu100 (aref *xu100* index))
      (setf n225  (aref *n225* index))
      (setf aord  (aref *aord* index))
      (setf hsi   (aref *hsi* index))
      (setf bsesn (aref *bsesn* index))
      (setf gdaxi (aref *gdaxi* index))
      (setf fchi  (aref *fchi* index))
      (setf ftse  (aref *ftse* index))
      (setf dji   (aref *dji* index))
      (setf ixic  (aref *ixic* index))
      (setf gspc  (aref *gspc* index))
      (setf bvsp  (aref *bvsp* index))
      (setf xoi   (aref *xoi* index))
      (setf xau   (aref *xau* index))
      (setf vix   (aref *vix* index))
      (setf this-fitness-case (make-SYMREG_U30-fitness-case))
      (setf (aref fitness-cases index) this-fitness-case)
      (setf (SYMREG_U30-fitness-case-var-1  this-fitness-case) xu030)
      (setf (SYMREG_U30-fitness-case-var-2  this-fitness-case) xu100)
      (setf (SYMREG_U30-fitness-case-var-3  this-fitness-case) n225 )
      (setf (SYMREG_U30-fitness-case-var-4  this-fitness-case) aord )
      (setf (SYMREG_U30-fitness-case-var-5  this-fitness-case) hsi  )
      (setf (SYMREG_U30-fitness-case-var-6  this-fitness-case) bsesn)
      (setf (SYMREG_U30-fitness-case-var-7  this-fitness-case) gdaxi)
      (setf (SYMREG_U30-fitness-case-var-8  this-fitness-case) fchi )
      (setf (SYMREG_U30-fitness-case-var-9  this-fitness-case) ftse )
      (setf (SYMREG_U30-fitness-case-var-10 this-fitness-case) dji  )
      (setf (SYMREG_U30-fitness-case-var-11 this-fitness-case) ixic )
      (setf (SYMREG_U30-fitness-case-var-12 this-fitness-case) gspc )
      (setf (SYMREG_U30-fitness-case-var-13 this-fitness-case) bvsp )
      (setf (SYMREG_U30-fitness-case-var-14 this-fitness-case) xoi  )
      (setf (SYMREG_U30-fitness-case-var-15 this-fitness-case) xau  )
      (setf (SYMREG_U30-fitness-case-var-16 this-fitness-case) vix  )
      (setf (SYMREG_U30-fitness-case-target this-fitness-case) (aref *xu030* (1+ index)))
      ;(format t "~% ~A ~A" index this-fitness-case)
      )
    (values fitness-cases)))

(defun SYMREG_U30-wrapper (result-from-program)
  (values result-from-program))

(defun evaluate-standardized-fitness-for-SYMREG_U30  (program fitness-cases)
```

```
(let ((raw-fitness 0.0)
      (hits 0)
      standardized-fitness
      xu030
      xu100
      n225
      aord
      hsi
      bsesn
      gdaxi
      fchi
      ftse
      dji
      ixic
      gspc
      bvsp
      xoi
      xau
      vix
      target-value
      difference
      value-from-program
      this-fitness-case)
  (dotimes (index *number-of-fitness-cases*)
    (setf this-fitness-case (aref fitness-cases index))
    (setf xu030 (SYMREG_U30-fitness-case-var-1  this-fitness-case))
    (setf xu100 (SYMREG_U30-fitness-case-var-2  this-fitness-case))
    (setf n225  (SYMREG_U30-fitness-case-var-3  this-fitness-case))
    (setf aord  (SYMREG_U30-fitness-case-var-4  this-fitness-case))
    (setf hsi   (SYMREG_U30-fitness-case-var-5  this-fitness-case))
    (setf bsesn (SYMREG_U30-fitness-case-var-6  this-fitness-case))
    (setf gdaxi (SYMREG_U30-fitness-case-var-7  this-fitness-case))
    (setf fchi  (SYMREG_U30-fitness-case-var-8  this-fitness-case))
    (setf ftse  (SYMREG_U30-fitness-case-var-9  this-fitness-case))
    (setf dji   (SYMREG_U30-fitness-case-var-10 this-fitness-case))
    (setf ixic  (SYMREG_U30-fitness-case-var-11 this-fitness-case))
    (setf gspc  (SYMREG_U30-fitness-case-var-12 this-fitness-case))
    (setf bvsp  (SYMREG_U30-fitness-case-var-13 this-fitness-case))
    (setf xoi   (SYMREG_U30-fitness-case-var-14 this-fitness-case))
    (setf xau   (SYMREG_U30-fitness-case-var-15 this-fitness-case))
    (setf vix   (SYMREG_U30-fitness-case-var-16 this-fitness-case))
    (setf target-value (SYMREG_U30-fitness-case-target this-fitness-case))
    (setf value-from-program (SYMREG_U30-wrapper (eval program)))
    (setf difference (abs (- target-value value-from-program)))
    (incf raw-fitness difference)
    (when (<= (/ difference target-value) 0.0025)
      (incf hits)))
```

```
    (setf standardized-fitness raw-fitness)
    (values standardized-fitness hits)))

(defun define-parameters-for-SYMREG_U30 ()
  (setf *number-of-fitness-cases* 2000)
  (setf *max-depth-for-new-individuals* 9)
  (setf *max-depth-for-new-subtrees-in-mutants* 4)
  (setf *max-depth-for-individuals-after-crossover* 24)
  (setf *fitness-proportionate-reproduction-fraction* 0.1)
  (setf *crossover-at-any-point-fraction* 0.2)
  (setf *crossover-at-function-point-fraction* 0.2)
  (setf *method-of-selection* :fitness-proportionate)
  (setf *method-of-generation* :ramped-half-and-half)
  (values))

(defun define-termination-criterion-for-SYMREG_U30
       (current-generation
        maximum-generations
        best-standardized-fitness
        best-hits)
  (declare (ignore best-standardized-fitness))
  (values
   (or (>= current-generation maximum-generations)
       (>= best-hits *number-of-fitness-cases*))))

(defun SYMREG_U30 ()
  (values 'define-function-set-for-SYMREG_U30
          'define-terminal-set-for-SYMREG_U30
          'define-fitness-cases-for-SYMREG_U30
          'evaluate-standardized-fitness-for-SYMREG_U30
          'define-parameters-for-SYMREG_U30
          'define-termination-criterion-for-SYMREG_U30))
```

C.3. Teknik Analiz Kütüphanesinin Kaynak Kodları

```
;;;; icll/finance/finance.lisp

(in-package :icll.finance)

(eval-when (:compile-toplevel)
  (declaim (optimize (speed 3) (debug 0) (safety 1) (space 0))))

(defun frame (seq i n)
  (declare (fixnum i n))
  ; Returns time frame i.e. subsequence
  (if (< i n)
      nil
      (subseq seq (- i n) i)))

(defun ma-sim (seq n)
  ; Simple moving everage
  (declare (fixnum n))
  (loop for i from 1 to (len seq) collect
       (if (< i n)
           nil
           (avg (frame seq i n)))))

(defun ma-exp (seq n &optional alpha)
  ; Exponential moving average
  (declare (fixnum n))
  (let* ((prev 0.0)
         (alpha (if (null alpha)
                    (/ 2.0 (+ 1 n))
                    alpha)))
    (loop for i from 1 to (len seq) collect
         (if (< i n)
             nil
             (if (= i n)
                 (progn (setf prev (avg (frame seq i n))) prev)
                 (progn (setf prev (+ (* alpha (seq-nth seq (1- i))) (* (- 1 alpha) prev))) prev))))))

(defun mstd (seq n)
  ; Moving standard deviation for later use in Bolinger Bands etc.
  (declare (fixnum n))
  (loop for i from 1 to (len seq) collect
       (if (< i n)
           nil
           (stdev (frame seq i n)))))

(defun ignoring-nils (fn seq &rest args)
```

```
  ; Some sequences especially the arguments in financial calculations
  ; may have some initial elements of nils. This function is necessary
  ; for computing moving averages and alike functions.
  (do ((i 0 (incf i))
       (result nil))
      ((not (null (nth i seq)))
       (append result (apply fn (subseq seq i) args)))
    (push nil result)))

(defun williams-formula (close-today high-ndays low-ndays)
  (declare (double-float close-today high-ndays low-ndays))
  (the double-float
    (* 100.0 (/ (- close-today high-ndays) (- high-ndays low-ndays)))))

(defun williams-%r (c h l &optional (n 14))
  ; Williams %R for a period of n.
  (loop for i from 1 to (len c) collect
       (if (< i n)
           nil
           (let* ((fr-high (frame h i n))
                  (fr-low (frame l i n))
                  (close-today (seq-nth c (1- i)))
                  (hn (seq-max fr-high))
                  (ln (seq-min fr-low)))
             (williams-formula close-today hn ln)))))

(defun stochastics-formula (close-today high-ndays low-ndays)
  (declare (double-float close-today high-ndays low-ndays))
  (the double-float
    (* 100.0 (/ (- close-today low-ndays) (- high-ndays low-ndays)))))

(defun stochastics (c h l &optional (n 14) (smooth 3))
  ; Stochastics %K, %D and %D-Slow for aperiod of n.
  (let* ((per-k
          (loop for i from 1 to (len c) collect
               (if (< i n)
                   nil
                   (let* ((fr-high (frame h i n))
                          (fr-low (frame l i n))
                          (close-today (seq-nth c (1- i)))
                          (hn (seq-max fr-high))
                          (ln (seq-min fr-low)))
                     (stochastics-formula close-today hn ln)))))
         (per-d (ignoring-nils #'ma-exp per-k smooth))
         (per-d-slow (ignoring-nils #'ma-exp per-d smooth)))
    (mapcar #'list per-k per-d per-d-slow)))
```

```
(defun dif (c)
  ; Compute and return difference of a serie wrt. the day before
  ; This serie will always be 1 observation shorter than than original serie.
  ; All the indicators based on this will also be shorter by one.
  (loop for i from 1 below (len c) collect
       (- (seq-nth c i) (seq-nth c (1- i)))))

(defun  u-d (dif)
  ; Compute and return (U . D) values given a dif of a serie.
  ; Return as a list of dotted pair. Required for RSI.
  (mapcar #'(lambda (x) (cond ((null x) nil)
                              ((= x 0) (cons 0 0))
                              ((pos-p x) (cons x 0))
                              ((neg-p x) (cons 0 (abs x)))))
          dif))

(defun rs-formula (emau emad)
  (mapcar #'(lambda (x y)
              (if (or (null x) (null y))
                  nil
                  (/ x y)))
                     emau emad))

(defun rsi (seq &optional (n 14))
  ; Relative Strength Index of a given historical series.
  ; Returned RSI serie should contain 1 less element than
  ; the original one and also the first (n-1) values should be nil.
  (let* ((df (dif seq))
         (ud (u-d df))
         (u (mapcar #'car ud))
         (d (mapcar #'cdr ud))
         (emau (ma-exp u n))
         (emad (ma-exp d n))
         (rs (rs-formula emau emad)))
    (mapcar #'(lambda (x)
                (if (null x) nil
                (- 100 (/ 100 (1+ x))))) rs)))

(defun bollinger-bands (seq &optional (k 2) (n 20))
  ; Returns a list of lists containing 5 elements
  ; bollinger bands (i.e. lbb, mean and ubb),
  ; %b and bandWith indicators
  (let* ((mean (ma-sim seq n))
         (std (mstd seq n))
         (lbb (mapcar-nils #'(lambda (m s) (- m (* k s))) mean std))
         (ubb (mapcar-nils #'(lambda (m s) (+ m (* k s))) mean std))
         (range (mapcar-nils #'(lambda (l u) (- u l)) lbb ubb))
```

```
        (bandwidth (mapcar-nils #'(lambda (r m) (/ r m)) range mean))
        (seq (if (arrayp seq) (setf seq (arr-to-lst seq))))
        (percent-b (mapcar-nils #'(lambda (c l r) (/ (- c l) r)) seq lbb range)))
    (mapcar #'list lbb mean ubb percent-b bandwidth)))

(defun macd (seq &optional (fast 12) (slow 26) (sign 9))
  ; Returns MACD indicator and its triggering signal
  (let* ((f (ma-exp seq fast))
         (s (ma-exp seq slow))
         (macd (mapcar-nils #'- f s))
         (sig (ignoring-nils #'ma-exp macd sign)))
    (mapcar #'cons macd sig)))
```

C.4. Ulusal 30 Endeksi Teknik Analiz Göstergelerine Dayalı Al-Sat Sinyalleri Üreten Kural Ağacı Uygulaması Kaynak Kodları

```
;;;; icll/gp/examples/tarules.lisp
;;;; Written by Ozgur ICAN

(in-package :icll.gp.examples.tarules)

(eval-when (:compile-toplevel)
  (declaim (optimize (speed 3) (debug 0) (safety 1) (space 0))))

;;; Below code is specific to evaluation of TA signals
(defun encode (signals)
  (declare (type (array boolean) signals))
  (let* ((len (len signals))
         (arr (make-array len :element-type 'fixnum))
         prev)
    (loop for i from 0 below len do
         (if (= i 0)
             (progn
               (case (aref signals i)
                 ('nil (setf (aref arr i) -1))
                 (t (setf (aref arr i) 1)))
               (setf prev (aref signals i)))
             (if (eql (aref signals i) prev) (setf (aref arr i) 0)
                 (progn
                   (case (aref signals i)
                     ('nil (setf (aref arr i) -1))
                     (t (setf (aref arr i) 1)))
                   (setf prev (aref signals i))))))
    arr))

(defstruct portfolio
  status
  cash
  position
  open-position
  asset-price
  value)

(defun port-value (port)
  (declare (portfolio port))
  (let* ((cash (portfolio-cash port))
         (pos (portfolio-position port))
         (op (portfolio-open-position port))
         (price (portfolio-asset-price port)))
```

```
    (+ cash (* pos price) (* (- op) price))))

(defun buy (p asset-price)
  (declare (portfolio p))
  (declare (double-float asset-price))
  (assert (string-equal (portfolio-status p) "liquid"))
  (setf (portfolio-status p) "long")
  (setf (portfolio-position p) (/ (portfolio-cash p) asset-price))
  (setf (portfolio-cash p) 0)
  (setf (portfolio-asset-price p) asset-price)
  (setf (portfolio-value p) (port-value p)))

(defun update-portfolio (p asset-price)
  (declare (portfolio p))
  (declare (double-float asset-price))
  (setf (portfolio-asset-price p) asset-price)
  (setf (portfolio-value p) (port-value p)))

(defun sell (p asset-price)
  (declare (portfolio p))
  (declare (double-float asset-price))
  (assert (string-equal (portfolio-status p) "long"))
  (setf (portfolio-status p) "liquid")
  (setf (portfolio-cash p) (* (portfolio-position p) asset-price))
  (setf (portfolio-position p) 0)
  (setf (portfolio-asset-price p) asset-price)
  (setf (portfolio-value p) (port-value p)))

(defun short-sell (p asset-price)
  ; Assuming that you can not short-sell when you've already short-sold or you're
  ; in long position. Thus one can only short-sell when completely liquid.
  ; Also one should close an open position before getting into long again.
  ; WARNING: Amount of short-selling assets is limited with current equivalent of cash.
  (declare (portfolio p))
  (declare (double-float asset-price))
  (assert (string-equal (portfolio-status p) "liquid"))
  (setf (portfolio-status p) "short-sell")
  (setf (portfolio-open-position p) (/ (portfolio-cash p) asset-price))
  (setf (portfolio-asset-price p) asset-price)
  (setf (portfolio-cash p) (* 2 (portfolio-cash p)))
  (setf (portfolio-value p) (port-value p)))

(defun close-open-pos (p asset-price)
  (declare (portfolio p))
  (declare (double-float asset-price))
  (assert (string-equal (portfolio-status p) "short-sell"))
  (setf (portfolio-status p) "liquid")
```

```
  (setf (portfolio-cash p)
        (- (portfolio-cash p)
           (* (portfolio-open-position p) asset-price)))
  (setf (portfolio-open-position p) 0.0)
  (setf (portfolio-asset-price p) asset-price)
  (setf (portfolio-value p) (port-value p)))

(defun evaluate-signal (signal-arr price-arr)
  (declare (type (array boolean) signal-arr))
  (declare (type simple-vector price-arr))
  (let* (my-port
         result
         (init-capital 100.0)
         (encoded (encode signal-arr))
         (len (len signal-arr)))
    (setf my-port (make-portfolio
                   :status "liquid"
                   :cash init-capital
                   :position 0.0
                   :open-position 0.0
                   :asset-price 0.0
                   :value init-capital))
    (do* ((i 0 (incf i))
          (sig (aref encoded i))
          (price (aref price-arr i))
          (status (portfolio-status my-port))
          (val (portfolio-value my-port)))
         ; loop till the end of fitness cases or drain your init capital under 5 percent
         ((or (>= i len) (< val 5.0)))
      (setf sig (aref encoded i))
      (setf price (aref price-arr i))
      (setf status (portfolio-status my-port))
      (setf val (portfolio-value my-port))
      (if (= i 0)
          (if (= sig 1)
              (buy my-port price)
              (short-sell my-port price))
          (case sig
            (0 (update-portfolio my-port price))
            (1 (progn (close-open-pos my-port price)
                      (buy my-port price)))
            (-1 (progn (sell my-port price)
                       (short-sell my-port price)))))
      ; If you want step by step information uncomment
      ;(format t "~%~D~T~D~T~D~T~D~T~A" i sig price val status)
      (setf result val))
    ; if result is greater than init-cap return it otherwise return cap
```

```
    ; its because cap/cap will generate 1.0 in std. fitness computation.
    ; Thus the greater result the closer std. fitness to 0.
    result
    ))
;;;; End of domain specific code

;;; Open database connection
(defparameter *li-db* (connect
                       '( "/home/thorin/subversion/oican/repos/trunk/coding/Python/financial_db/hp_li.db")
                       :pool t :database-type :sqlite3))

;;; A utility function
(defun to-array (lst fn)
  (make-array (length lst) :initial-contents (mapcar fn lst)))

(defparameter *start* "'20000101'")
(defparameter *end* "'20120315'")

(defvar *query-string*
  (concatenate 'string
               "select Day, High, Low, Close from XU030_HT where Day between "
               *start* " and " *end* " order by Day asc;"))

;;; Pull XU030 data from DB
(defparameter *xu030*
   (query *query-string* :database *li-db*))

;;; Close database connection
(disconnect :database *li-db*)

;;; Lists returned from database query turned into arrays
(defvar *time-arr* (to-array *xu030* #'car))
(defvar *harr* (to-array *xu030* #'cadr))
(defvar *larr* (to-array *xu030* #'caddr))
(defvar *carr* (to-array *xu030* #'cadddr))

;;; Technical indicators computed.
(defvar *macd* (macd *carr*))
(defvar *bollinger-bands* (bollinger-bands *carr*))
(defvar *rsi* (rsi *carr*))
(defvar *stochastics* (stochastics *carr* *harr* *larr*))
(defvar *williams* (williams-%r *carr* *harr* *larr*))
;; (defvar *ma-sim-20* (ma-sim *carr* 20))
;; (defvar *ma-sim-60* (ma-sim *carr* 60))
;; (defvar *ma-sim-120* (ma-sim *carr* 120))
;; (defvar *ma-sim-200* (ma-sim *carr* 200))
;; (defvar *ma-exp-20* (ma-exp *carr* 20))
```

```
;; (defvar *ma-exp-60* (ma-exp *carr* 60))
;; (defvar *ma-exp-120* (ma-exp *carr* 120))
;; (defvar *ma-exp-200* (ma-exp *carr* 200))

(defparameter *clip* 50) ; For clipping nil values of TA arays

;;; Clip arrays from the beginning, for appropriately eliminating nil elements
;;; therefore making every array in the same size
(setf *macd* (subseq *macd* *clip*))
(setf *bollinger-bands* (subseq *bollinger-bands* *clip*))
(setf *rsi* (subseq *rsi* (1- *clip*))) ; Remember RSI has one least element.
(setf *stochastics* (subseq *stochastics* *clip*))
(setf *williams* (subseq *williams* *clip*))
(setf *time-arr* (subseq *time-arr* *clip*)) ; These last two are
(setf *carr* (subseq *carr* *clip*)) ; for later use.

(defvar *test-range* 60)
(defvar *final-length* (len *macd*))
(defvar *learning-range* (- *final-length* *test-range*)) ;*test-range* should be set before

;;; Rearrange arrays according to testing and learning and period
(defvar *macd-test* (last *macd* *test-range*))
(defvar *bollinger-bands-test* (last *bollinger-bands* *test-range*))
(defvar *rsi-test* (last *rsi* *test-range*))
(defvar *stochastics-test* (last *stochastics* *test-range*))
(defvar *williams-test* (last *williams* *test-range*))
(defvar *time-test* (arr-last *time-arr* *test-range*))
(defvar *carr-test* (arr-last *carr* *test-range*))

(setf *macd* (subseq *macd* 0 *learning-range*))
(setf *bollinger-bands* (subseq *bollinger-bands* 0 *learning-range*))
(setf *rsi* (subseq *rsi*  0 *learning-range*))
(setf *stochastics* (subseq *stochastics* 0 *learning-range*))
(setf *williams* (subseq *williams* 0 *learning-range*))
(setf *time-arr* (subseq *time-arr* 0 *learning-range*))
(setf *carr* (subseq *carr* 0 *learning-range*))

;;; Learning and testing periods' boundary dates and prices
(defvar *TRANGE-START-DATE* (arr-first *time-test*))
(defvar *TRANGE-END-DATE* (arr-last *time-test*))
(defvar *TRANGE-START-PRICE* (arr-first *carr-test*))
(defvar *TRANGE-END-PRICE* (arr-last *carr-test*))
(defvar *LRANGE-START-DATE* (arr-first *time-arr*))
(defvar *LRANGE-END-DATE* (arr-last *time-arr*))

; will be compared to its triggering signal macd-sig
(defstruct smacd ind sig)
```

```
(defvar macd)

; %b is enough for representing band idea, oscillates  0 <= %b <= 1 , thus
; will be compared to lower and upper triggers.
(defstruct sboll ind tr)
(defvar boll0)
(defvar boll1)
(defvar boll2)
(defvar boll3)
(defvar boll4)
(defvar boll5)
(defvar boll6)
(defvar boll7)
(defvar boll8)
(defvar boll9)
(defvar boll10)

; RSI oscillates 0 <= RSI <= 100 and will be compared to a random
; lower trigger and upper trigger.
(defstruct srsi ind tr)
(defvar rsi0)
(defvar rsi1)
(defvar rsi2)
(defvar rsi3)
(defvar rsi4)
(defvar rsi5)
(defvar rsi6)
(defvar rsi7)
(defvar rsi8)
(defvar rsi9)
(defvar rsi10)

; %K will be compared and %D-Slow. Note that we're not comparing
; %K according to an absolute level because probably %R will do this job.
(defstruct sstoch k ds)
(defvar stoch)

; %R oscillates between -100 <= %R <= 0 and will be compared to
; lower and upper triggers.
(defstruct swill ind tr)
(defvar will0)
(defvar will1)
(defvar will2)
(defvar will3)
(defvar will4)
(defvar will5)
(defvar will6)
```

```
(defvar will7)
(defvar will8)
(defvar will9)
(defvar will10)

(defun define-terminal-set-for-TARULES ()
  (values '(macd stoch
            boll0   rsi0   will0
            boll1   rsi1   will1
            boll2   rsi2   will2
            boll3   rsi3   will3
            boll4   rsi4   will4
            boll5   rsi5   will5
            boll6   rsi6   will6
            boll7   rsi7   will7
            boll8   rsi8   will8
            boll9   rsi9   will9
            boll10  rsi10  will10)))


(defun m<= (x)
  (cond ((smacd-p x) (<= (smacd-ind x) (smacd-sig x)))
        ((sboll-p x) (<= (sboll-ind x) (sboll-tr x)))
        ((srsi-p x) (<= (srsi-ind x) (srsi-tr x)))
        ((sstoch-p x) (<= (sstoch-k x) (sstoch-ds x)))
        ((swill-p x) (<= (swill-ind x) (swill-tr x)))
        (t (error "Illegal argument passed to m<= !!!"))))

(defun m> (x)
  (cond ((smacd-p x) (> (smacd-ind x) (smacd-sig x)))
        ((sboll-p x) (> (sboll-ind x) (sboll-tr x)))
        ((srsi-p x) (> (srsi-ind x) (srsi-tr x)))
        ((sstoch-p x) (> (sstoch-k x) (sstoch-ds x)))
        ((swill-p x) (> (swill-ind x) (swill-tr x)))
        (t (error "Illegal argument passed to m> !!!"))))

(defun ind-struct-p (x)
  (or (smacd-p x)
      (sboll-p x)
      (srsi-p x)
      (sstoch-p x)
      (swill-p x)))

(defun mand<= (x y)
  (cond ((and (ind-struct-p x) (ind-struct-p y))
         (and (m<= x)
              (m<= y)))
```

```
        ((ind-struct-p x) (and (m<= x) y))
        ((ind-struct-p y) (and x (m<= y)))
        (t (and x y))))

(defun mand> (x y)
  (cond ((and (ind-struct-p x) (ind-struct-p y))
         (and (m> x)
              (m> y)))
        ((ind-struct-p x) (and (m> x) y))
        ((ind-struct-p y) (and x (m> y)))
        (t (and x y))))

(defun mor<= (x y)
   (cond ((and (ind-struct-p x) (ind-struct-p y))
          (or (m<= x)
              (m<= y)))
         ((ind-struct-p x) (or (m<= x) y))
         ((ind-struct-p y) (or x (m<= y)))
         (t (or x y))))

(defun mor> (x y)
  (cond ((and (ind-struct-p x) (ind-struct-p y))
         (or (m> x)
             (m> y)))
        ((ind-struct-p x) (or (m> x) y))
        ((ind-struct-p y) (or x (m> y)))
        (t (or x y))))

(defun define-function-set-for-TARULES ()
  (values '(mor<= mor> mand<= mand>)
          '(2 2 2 2)))

(defstruct TARULES-fitness-case
  ; TA indicators as data structures
  macd stoch
  boll0   rsi0   will0
  boll1   rsi1   will1
  boll2   rsi2   will2
  boll3   rsi3   will3
  boll4   rsi4   will4
  boll5   rsi5   will5
  boll6   rsi6   will6
  boll7   rsi7   will7
  boll8   rsi8   will8
  boll9   rsi9   will9
  boll10  rsi10  will10)
```

```
(defun define-fitness-cases-for-TARULES ()
  (let* (fitness-cases
         this-fitness-case
         (macd-arr (to-array *macd* #'car))
         (len (arr-len macd-arr))
         (macd-sig-arr (to-array *macd* #'cdr))
         (bb-%b-arr (to-array *bollinger-bands* #'cadddr))
         (rsi-arr (make-array len :initial-contents *rsi*))
         (will-arr (make-array len :initial-contents *williams*))
         (stochk-arr (to-array *stochastics* #'car))
         (stochds-arr (to-array *stochastics* #'caddr))
         (m_ (make-array len))
         (s_ (make-array len))
         (b_0 (make-array len)) (b_1 (make-array len)) (b_2 (make-array len)) (b_3 (make-array len))
         (b_4 (make-array len)) (b_5 (make-array len)) (b_6 (make-array len)) (b_7 (make-array len))
         (b_8 (make-array len)) (b_9 (make-array len)) (b_10 (make-array len))
         (r_0 (make-array len)) (r_1 (make-array len)) (r_2 (make-array len)) (r_3 (make-array len))
         (r_4 (make-array len)) (r_5 (make-array len)) (r_6 (make-array len)) (r_7 (make-array len))
         (r_8 (make-array len)) (r_9 (make-array len)) (r_10 (make-array len))
         (w_0 (make-array len)) (w_1 (make-array len)) (w_2 (make-array len)) (w_3 (make-array len))
         (w_4 (make-array len)) (w_5 (make-array len)) (w_6 (make-array len)) (w_7 (make-array len))
         (w_8 (make-array len)) (w_9 (make-array len)) (w_10 (make-array len)))
    (setf fitness-cases (make-array *number-of-fitness-cases*))
    (format t "~%Fitness cases: ")
    (dotimes  (i *number-of-fitness-cases*)
      (setf (aref m_ i) (make-smacd :ind (aref macd-arr i) :sig (aref macd-sig-arr i)))
      (setf (aref s_ i) (make-sstoch :k (aref stochk-arr i) :ds (aref stochds-arr i)))
      (setf (aref b_0 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.0 ))
      (setf (aref b_1 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.10 ))
      (setf (aref b_2 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.20 ))
      (setf (aref b_3 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.30 ))
      (setf (aref b_4 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.40 ))
      (setf (aref b_5 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.50 ))
      (setf (aref b_6 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.60 ))
      (setf (aref b_7 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.70 ))
      (setf (aref b_8 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.80 ))
      (setf (aref b_9 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.90 ))
      (setf (aref b_10 i) (make-sboll :ind (aref bb-%b-arr i) :tr 1.0 ))
      (setf (aref r_0 i) (make-srsi :ind (aref rsi-arr i) :tr 0.0 ))
      (setf (aref r_1 i) (make-srsi :ind (aref rsi-arr i) :tr 10.0 ))
      (setf (aref r_2 i) (make-srsi :ind (aref rsi-arr i) :tr 20.0 ))
      (setf (aref r_3 i) (make-srsi :ind (aref rsi-arr i) :tr 30.0 ))
      (setf (aref r_4 i) (make-srsi :ind (aref rsi-arr i) :tr 40.0 ))
      (setf (aref r_5 i) (make-srsi :ind (aref rsi-arr i) :tr 50.0 ))
      (setf (aref r_6 i) (make-srsi :ind (aref rsi-arr i) :tr 60.0 ))
      (setf (aref r_7 i) (make-srsi :ind (aref rsi-arr i) :tr 70.0 ))
      (setf (aref r_8 i) (make-srsi :ind (aref rsi-arr i) :tr 80.0 ))
```

```
(setf (aref r_9 i) (make-srsi :ind (aref rsi-arr i) :tr 90.0 ))
(setf (aref r_10 i) (make-srsi :ind (aref rsi-arr i) :tr 100.0 ))
(setf (aref w_0 i) (make-swill :ind (aref will-arr i) :tr 0.0 ))
(setf (aref w_1 i) (make-swill :ind (aref will-arr i) :tr -10.0 ))
(setf (aref w_2 i) (make-swill :ind (aref will-arr i) :tr -20.0 ))
(setf (aref w_3 i) (make-swill :ind (aref will-arr i) :tr -30.0 ))
(setf (aref w_4 i) (make-swill :ind (aref will-arr i) :tr -40.0 ))
(setf (aref w_5 i) (make-swill :ind (aref will-arr i) :tr -50.0 ))
(setf (aref w_6 i) (make-swill :ind (aref will-arr i) :tr -60.0 ))
(setf (aref w_7 i) (make-swill :ind (aref will-arr i) :tr -70.0 ))
(setf (aref w_8 i) (make-swill :ind (aref will-arr i) :tr -80.0 ))
(setf (aref w_9 i) (make-swill :ind (aref will-arr i) :tr -90.0 ))
(setf (aref w_10 i) (make-swill :ind (aref will-arr i) :tr -100.0 ))
(setf this-fitness-case (make-TARULES-fitness-case))
(setf (aref fitness-cases i) this-fitness-case)
(setf (TARULES-fitness-case-macd this-fitness-case) (aref m_ i))
(setf (TARULES-fitness-case-stoch this-fitness-case) (aref s_ i))
(setf (TARULES-fitness-case-boll0 this-fitness-case) (aref b_0 i))
(setf (TARULES-fitness-case-boll1 this-fitness-case) (aref b_1 i))
(setf (TARULES-fitness-case-boll2 this-fitness-case) (aref b_2 i))
(setf (TARULES-fitness-case-boll3 this-fitness-case) (aref b_3 i))
(setf (TARULES-fitness-case-boll4 this-fitness-case) (aref b_4 i))
(setf (TARULES-fitness-case-boll5 this-fitness-case) (aref b_5 i))
(setf (TARULES-fitness-case-boll6 this-fitness-case) (aref b_6 i))
(setf (TARULES-fitness-case-boll7 this-fitness-case) (aref b_7 i))
(setf (TARULES-fitness-case-boll8 this-fitness-case) (aref b_8 i))
(setf (TARULES-fitness-case-boll9 this-fitness-case) (aref b_9 i))
(setf (TARULES-fitness-case-boll10 this-fitness-case) (aref b_10 i))
(setf (TARULES-fitness-case-rsi0 this-fitness-case) (aref r_0 i))
(setf (TARULES-fitness-case-rsi1 this-fitness-case) (aref r_1 i))
(setf (TARULES-fitness-case-rsi2 this-fitness-case) (aref r_2 i))
(setf (TARULES-fitness-case-rsi3 this-fitness-case) (aref r_3 i))
(setf (TARULES-fitness-case-rsi4 this-fitness-case) (aref r_4 i))
(setf (TARULES-fitness-case-rsi5 this-fitness-case) (aref r_5 i))
(setf (TARULES-fitness-case-rsi6 this-fitness-case) (aref r_6 i))
(setf (TARULES-fitness-case-rsi7 this-fitness-case) (aref r_7 i))
(setf (TARULES-fitness-case-rsi8 this-fitness-case) (aref r_8 i))
(setf (TARULES-fitness-case-rsi9 this-fitness-case) (aref r_9 i))
(setf (TARULES-fitness-case-rsi10 this-fitness-case) (aref r_10 i))
(setf (TARULES-fitness-case-will0 this-fitness-case) (aref w_0 i))
(setf (TARULES-fitness-case-will1 this-fitness-case) (aref w_1 i))
(setf (TARULES-fitness-case-will2 this-fitness-case) (aref w_2 i))
(setf (TARULES-fitness-case-will3 this-fitness-case) (aref w_3 i))
(setf (TARULES-fitness-case-will4 this-fitness-case) (aref w_4 i))
(setf (TARULES-fitness-case-will5 this-fitness-case) (aref w_5 i))
(setf (TARULES-fitness-case-will6 this-fitness-case) (aref w_6 i))
(setf (TARULES-fitness-case-will7 this-fitness-case) (aref w_7 i))
```

```
      (setf (TARULES-fitness-case-will8 this-fitness-case) (aref w_8 i))
      (setf (TARULES-fitness-case-will9 this-fitness-case) (aref w_9 i))
      (setf (TARULES-fitness-case-will10 this-fitness-case) (aref w_10 i)))
      ;(format t "~%~A~&~A" i this-fitness-case)
    (values fitness-cases)))

(defun define-fitness-cases-for-TARULES-TEST ()
  (let* (fitness-cases
         this-fitness-case
         (macd-arr (to-array *macd-test* #'car))
         (len (arr-len macd-arr))
         (macd-sig-arr (to-array *macd-test* #'cdr))
         (bb-%b-arr (to-array *bollinger-bands-test* #'cadddr))
         (rsi-arr (make-array len :initial-contents *rsi-test*))
         (will-arr (make-array len :initial-contents *williams-test*))
         (stochk-arr (to-array *stochastics-test* #'car))
         (stochds-arr (to-array *stochastics-test* #'caddr))
         (m_ (make-array len))
         (s_ (make-array len))
         (b_0 (make-array len)) (b_1 (make-array len)) (b_2 (make-array len)) (b_3 (make-array len))
         (b_4 (make-array len)) (b_5 (make-array len)) (b_6 (make-array len)) (b_7 (make-array len))
         (b_8 (make-array len)) (b_9 (make-array len)) (b_10 (make-array len))
         (r_0 (make-array len)) (r_1 (make-array len)) (r_2 (make-array len)) (r_3 (make-array len))
         (r_4 (make-array len)) (r_5 (make-array len)) (r_6 (make-array len)) (r_7 (make-array len))
         (r_8 (make-array len)) (r_9 (make-array len)) (r_10 (make-array len))
         (w_0 (make-array len)) (w_1 (make-array len)) (w_2 (make-array len)) (w_3 (make-array len))
         (w_4 (make-array len)) (w_5 (make-array len)) (w_6 (make-array len)) (w_7 (make-array len))
         (w_8 (make-array len)) (w_9 (make-array len)) (w_10 (make-array len)))
    (setf fitness-cases (make-array *test-range*))
    (dotimes  (i *test-range*)
      (setf (aref m_ i) (make-smacd :ind (aref macd-arr i) :sig (aref macd-sig-arr i)))
      (setf (aref s_ i) (make-sstoch :k (aref stochk-arr i) :ds (aref stochds-arr i)))
      (setf (aref b_0 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.0 ))
      (setf (aref b_1 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.10 ))
      (setf (aref b_2 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.20 ))
      (setf (aref b_3 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.30 ))
      (setf (aref b_4 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.40 ))
      (setf (aref b_5 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.50 ))
      (setf (aref b_6 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.60 ))
      (setf (aref b_7 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.70 ))
      (setf (aref b_8 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.80 ))
      (setf (aref b_9 i) (make-sboll :ind (aref bb-%b-arr i) :tr 0.90 ))
      (setf (aref b_10 i) (make-sboll :ind (aref bb-%b-arr i) :tr 1.0 ))
      (setf (aref r_0 i) (make-srsi :ind (aref rsi-arr i) :tr 0.0 ))
      (setf (aref r_1 i) (make-srsi :ind (aref rsi-arr i) :tr 10.0 ))
      (setf (aref r_2 i) (make-srsi :ind (aref rsi-arr i) :tr 20.0 ))
      (setf (aref r_3 i) (make-srsi :ind (aref rsi-arr i) :tr 30.0 ))
```

```
(setf (aref r_4 i) (make-srsi :ind (aref rsi-arr i) :tr 40.0 ))
(setf (aref r_5 i) (make-srsi :ind (aref rsi-arr i) :tr 50.0 ))
(setf (aref r_6 i) (make-srsi :ind (aref rsi-arr i) :tr 60.0 ))
(setf (aref r_7 i) (make-srsi :ind (aref rsi-arr i) :tr 70.0 ))
(setf (aref r_8 i) (make-srsi :ind (aref rsi-arr i) :tr 80.0 ))
(setf (aref r_9 i) (make-srsi :ind (aref rsi-arr i) :tr 90.0 ))
(setf (aref r_10 i) (make-srsi :ind (aref rsi-arr i) :tr 100.0 ))
(setf (aref w_0 i) (make-swill :ind (aref will-arr i) :tr 0.0 ))
(setf (aref w_1 i) (make-swill :ind (aref will-arr i) :tr -10.0 ))
(setf (aref w_2 i) (make-swill :ind (aref will-arr i) :tr -20.0 ))
(setf (aref w_3 i) (make-swill :ind (aref will-arr i) :tr -30.0 ))
(setf (aref w_4 i) (make-swill :ind (aref will-arr i) :tr -40.0 ))
(setf (aref w_5 i) (make-swill :ind (aref will-arr i) :tr -50.0 ))
(setf (aref w_6 i) (make-swill :ind (aref will-arr i) :tr -60.0 ))
(setf (aref w_7 i) (make-swill :ind (aref will-arr i) :tr -70.0 ))
(setf (aref w_8 i) (make-swill :ind (aref will-arr i) :tr -80.0 ))
(setf (aref w_9 i) (make-swill :ind (aref will-arr i) :tr -90.0 ))
(setf (aref w_10 i) (make-swill :ind (aref will-arr i) :tr -100.0 ))
(setf this-fitness-case (make-TARULES-fitness-case))
(setf (aref fitness-cases i) this-fitness-case)
(setf (TARULES-fitness-case-macd this-fitness-case) (aref m_ i))
(setf (TARULES-fitness-case-stoch this-fitness-case) (aref s_ i))
(setf (TARULES-fitness-case-boll0 this-fitness-case) (aref b_0 i))
(setf (TARULES-fitness-case-boll1 this-fitness-case) (aref b_1 i))
(setf (TARULES-fitness-case-boll2 this-fitness-case) (aref b_2 i))
(setf (TARULES-fitness-case-boll3 this-fitness-case) (aref b_3 i))
(setf (TARULES-fitness-case-boll4 this-fitness-case) (aref b_4 i))
(setf (TARULES-fitness-case-boll5 this-fitness-case) (aref b_5 i))
(setf (TARULES-fitness-case-boll6 this-fitness-case) (aref b_6 i))
(setf (TARULES-fitness-case-boll7 this-fitness-case) (aref b_7 i))
(setf (TARULES-fitness-case-boll8 this-fitness-case) (aref b_8 i))
(setf (TARULES-fitness-case-boll9 this-fitness-case) (aref b_9 i))
(setf (TARULES-fitness-case-boll10 this-fitness-case) (aref b_10 i))
(setf (TARULES-fitness-case-rsi0 this-fitness-case) (aref r_0 i))
(setf (TARULES-fitness-case-rsi1 this-fitness-case) (aref r_1 i))
(setf (TARULES-fitness-case-rsi2 this-fitness-case) (aref r_2 i))
(setf (TARULES-fitness-case-rsi3 this-fitness-case) (aref r_3 i))
(setf (TARULES-fitness-case-rsi4 this-fitness-case) (aref r_4 i))
(setf (TARULES-fitness-case-rsi5 this-fitness-case) (aref r_5 i))
(setf (TARULES-fitness-case-rsi6 this-fitness-case) (aref r_6 i))
(setf (TARULES-fitness-case-rsi7 this-fitness-case) (aref r_7 i))
(setf (TARULES-fitness-case-rsi8 this-fitness-case) (aref r_8 i))
(setf (TARULES-fitness-case-rsi9 this-fitness-case) (aref r_9 i))
(setf (TARULES-fitness-case-rsi10 this-fitness-case) (aref r_10 i))
(setf (TARULES-fitness-case-will0 this-fitness-case) (aref w_0 i))
(setf (TARULES-fitness-case-will1 this-fitness-case) (aref w_1 i))
(setf (TARULES-fitness-case-will2 this-fitness-case) (aref w_2 i))
```

```
      (setf (TARULES-fitness-case-will3 this-fitness-case) (aref w_3 i))
      (setf (TARULES-fitness-case-will4 this-fitness-case) (aref w_4 i))
      (setf (TARULES-fitness-case-will5 this-fitness-case) (aref w_5 i))
      (setf (TARULES-fitness-case-will6 this-fitness-case) (aref w_6 i))
      (setf (TARULES-fitness-case-will7 this-fitness-case) (aref w_7 i))
      (setf (TARULES-fitness-case-will8 this-fitness-case) (aref w_8 i))
      (setf (TARULES-fitness-case-will9 this-fitness-case) (aref w_9 i))
      (setf (TARULES-fitness-case-will10 this-fitness-case) (aref w_10 i)))
      ;(format t "~%~A~&~A" i this-fitness-case)
    (values fitness-cases)))

(defun TARULES-wrapper (result-from-program)
  (values result-from-program))

(defun evaluate-standardized-fitness-for-TARULES  (program fitness-cases)
  (let* (macd stoch
         boll0 boll1 boll2 boll3 boll4 boll5 boll6 boll7 boll8 boll9 boll10
         rsi0 rsi1 rsi2 rsi3 rsi4 rsi5 rsi6 rsi7 rsi8 rsi9 rsi10
         will0 will1 will2 will3 will4 will5 will6 will7 will8 will9 will10
         (raw-fitness 0.0)
         (hits 0)
         standardized-fitness
         (cap 100.0)
         this-fitness-case
         value-from-program
         (signals (make-array *learning-range* :element-type 'boolean)))
    (dotimes (index *number-of-fitness-cases*)
      (setf this-fitness-case (aref fitness-cases index))
      (setf macd (TARULES-fitness-case-macd this-fitness-case))
      (setf stoch (TARULES-fitness-case-stoch this-fitness-case))
      (setf boll0 (TARULES-fitness-case-boll0 this-fitness-case))
      (setf boll1 (TARULES-fitness-case-boll1 this-fitness-case))
      (setf boll2 (TARULES-fitness-case-boll2 this-fitness-case))
      (setf boll3 (TARULES-fitness-case-boll3 this-fitness-case))
      (setf boll4 (TARULES-fitness-case-boll4 this-fitness-case))
      (setf boll5 (TARULES-fitness-case-boll5 this-fitness-case))
      (setf boll6 (TARULES-fitness-case-boll6 this-fitness-case))
      (setf boll7 (TARULES-fitness-case-boll7 this-fitness-case))
      (setf boll8 (TARULES-fitness-case-boll8 this-fitness-case))
      (setf boll9 (TARULES-fitness-case-boll9 this-fitness-case))
      (setf boll10 (TARULES-fitness-case-boll10 this-fitness-case))
      (setf rsi0 (TARULES-fitness-case-rsi0 this-fitness-case))
      (setf rsi1 (TARULES-fitness-case-rsi1 this-fitness-case))
      (setf rsi2 (TARULES-fitness-case-rsi2 this-fitness-case))
      (setf rsi3 (TARULES-fitness-case-rsi3 this-fitness-case))
      (setf rsi4 (TARULES-fitness-case-rsi4 this-fitness-case))
      (setf rsi5 (TARULES-fitness-case-rsi5 this-fitness-case))
```

```
      (setf rsi6 (TARULES-fitness-case-rsi6 this-fitness-case))
      (setf rsi7 (TARULES-fitness-case-rsi7 this-fitness-case))
      (setf rsi8 (TARULES-fitness-case-rsi8 this-fitness-case))
      (setf rsi9 (TARULES-fitness-case-rsi9 this-fitness-case))
      (setf rsi10 (TARULES-fitness-case-rsi10 this-fitness-case))
      (setf will0 (TARULES-fitness-case-will0 this-fitness-case))
      (setf will1 (TARULES-fitness-case-will1 this-fitness-case))
      (setf will2 (TARULES-fitness-case-will2 this-fitness-case))
      (setf will3 (TARULES-fitness-case-will3 this-fitness-case))
      (setf will4 (TARULES-fitness-case-will4 this-fitness-case))
      (setf will5 (TARULES-fitness-case-will5 this-fitness-case))
      (setf will6 (TARULES-fitness-case-will6 this-fitness-case))
      (setf will7 (TARULES-fitness-case-will7 this-fitness-case))
      (setf will8 (TARULES-fitness-case-will8 this-fitness-case))
      (setf will9 (TARULES-fitness-case-will9 this-fitness-case))
      (setf will10 (TARULES-fitness-case-will10 this-fitness-case))
      (setf value-from-program (TARULES-wrapper (fast-eval program)))
      (setf (aref signals index) value-from-program))
    (setf raw-fitness (/ cap (evaluate-signal signals *carr*)))
    (setf standardized-fitness raw-fitness)
    (values standardized-fitness hits)))

(defun evaluate-standardized-fitness-for-TARULES-TEST (program fitness-cases)
  (let* (macd stoch
         boll0 boll1 boll2 boll3 boll4 boll5 boll6 boll7 boll8 boll9 boll10
         rsi0 rsi1 rsi2 rsi3 rsi4 rsi5 rsi6 rsi7 rsi8 rsi9 rsi10
         will0 will1 will2 will3 will4 will5 will6 will7 will8 will9 will10
         (raw-fitness 0.0)
         (hits 0)
         standardized-fitness
         (cap 100.0)
         this-fitness-case
         value-from-program
         (signals (make-array *test-range* :element-type 'boolean)))
    (dotimes (index *test-range*)
      (setf this-fitness-case (aref fitness-cases index))
      (setf macd (TARULES-fitness-case-macd this-fitness-case))
      (setf stoch (TARULES-fitness-case-stoch this-fitness-case))
      (setf boll0 (TARULES-fitness-case-boll0 this-fitness-case))
      (setf boll1 (TARULES-fitness-case-boll1 this-fitness-case))
      (setf boll2 (TARULES-fitness-case-boll2 this-fitness-case))
      (setf boll3 (TARULES-fitness-case-boll3 this-fitness-case))
      (setf boll4 (TARULES-fitness-case-boll4 this-fitness-case))
      (setf boll5 (TARULES-fitness-case-boll5 this-fitness-case))
      (setf boll6 (TARULES-fitness-case-boll6 this-fitness-case))
      (setf boll7 (TARULES-fitness-case-boll7 this-fitness-case))
      (setf boll8 (TARULES-fitness-case-boll8 this-fitness-case))
```

```
      (setf boll9 (TARULES-fitness-case-boll9 this-fitness-case))
      (setf boll10 (TARULES-fitness-case-boll10 this-fitness-case))
      (setf rsi0 (TARULES-fitness-case-rsi0 this-fitness-case))
      (setf rsi1 (TARULES-fitness-case-rsi1 this-fitness-case))
      (setf rsi2 (TARULES-fitness-case-rsi2 this-fitness-case))
      (setf rsi3 (TARULES-fitness-case-rsi3 this-fitness-case))
      (setf rsi4 (TARULES-fitness-case-rsi4 this-fitness-case))
      (setf rsi5 (TARULES-fitness-case-rsi5 this-fitness-case))
      (setf rsi6 (TARULES-fitness-case-rsi6 this-fitness-case))
      (setf rsi7 (TARULES-fitness-case-rsi7 this-fitness-case))
      (setf rsi8 (TARULES-fitness-case-rsi8 this-fitness-case))
      (setf rsi9 (TARULES-fitness-case-rsi9 this-fitness-case))
      (setf rsi10 (TARULES-fitness-case-rsi10 this-fitness-case))
      (setf will0 (TARULES-fitness-case-will0 this-fitness-case))
      (setf will1 (TARULES-fitness-case-will1 this-fitness-case))
      (setf will2 (TARULES-fitness-case-will2 this-fitness-case))
      (setf will3 (TARULES-fitness-case-will3 this-fitness-case))
      (setf will4 (TARULES-fitness-case-will4 this-fitness-case))
      (setf will5 (TARULES-fitness-case-will5 this-fitness-case))
      (setf will6 (TARULES-fitness-case-will6 this-fitness-case))
      (setf will7 (TARULES-fitness-case-will7 this-fitness-case))
      (setf will8 (TARULES-fitness-case-will8 this-fitness-case))
      (setf will9 (TARULES-fitness-case-will9 this-fitness-case))
      (setf will10 (TARULES-fitness-case-will10 this-fitness-case))
      (setf value-from-program (TARULES-wrapper (fast-eval program)))
      (setf (aref signals index) value-from-program))
    (setf raw-fitness (/ cap (evaluate-signal signals *carr-test*)))
    (setf standardized-fitness raw-fitness)
    (values standardized-fitness hits)))

(defun define-parameters-for-TARULES ()
  (setf *number-of-fitness-cases* *learning-range*)
  (setf *max-depth-for-new-individuals* 6)
  (setf *max-depth-for-new-subtrees-in-mutants* 3)
  (setf *max-depth-for-individuals-after-crossover* 12)
  (setf *fitness-proportionate-reproduction-fraction* 0.2)
  (setf *crossover-at-any-point-fraction* 0.2)
  (setf *crossover-at-function-point-fraction* 0.7)
  (setf *method-of-selection* :fitness-proportionate)
  (setf *method-of-generation* :ramped-half-and-half)
  (values))

(defun define-termination-criterion-for-TARULES
    (current-generation
     maximum-generations
     best-standardized-fitness
     best-hits)
```

```
  (declare (ignore best-hits best-standardized-fitness))
  (values
    (>= current-generation maximum-generations)))

(defun TARULES ()
  (values 'define-function-set-for-TARULES
          'define-terminal-set-for-TARULES
          'define-fitness-cases-for-TARULES
          'evaluate-standardized-fitness-for-TARULES
          'define-parameters-for-TARULES
          'define-termination-criterion-for-TARULES))
```

Kaynakça

Allen, F. ve Karjalainen, R. (1999). Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, *51* (2), 245–271. Retrieved from `http://linkinghub.elsevier.com/retrieve/pii/S0304405X9800052X`

Allen, H. ve Taylor, M. P. (1990, Supplemen). Charts, noise and fundamentals in the london foreign exchange market. *Economic Journal*, *100* (400), 49-59. Retrieved from `http://ideas.repec.org/a/ecj/econjl/v100y1990i400p49-59.html`

Andreoni, J. ve Miller, J. (1991). *Auctions with adaptive artificial agents* (Tech. Rep.). Technical Report 91-01-004, Santa Fe Institute, Santa Fe, New Mexico 87501.

Andreoni, J. ve Miller, J. (1993). Rational cooperation in the finitely repeated prisoner's dilemma: Experimental evidence. *The Economic Journal*, (), 570–585.

Andresen, S. (2001, jul-aug). Herbert a. simon: Ai pioneer. *Intelligent Systems, IEEE*, *16* (4), 71 - 72. doi: 10.1109/5254.941361

Angeline, P. J. (1997, 13-16). Subtree crossover: Building block engine or macromutation? In J. R. Koza vd. (Eds.), *Genetic programming 1997: Proceedings of the second annual conference* (pp. 9–17). Stanford University, CA, USA: Morgan Kaufmann.

Arifovic, J. (1994). Genetic algorithm learning and the cobweb model. *Journal of Economic dynamics and Control*, *18* (1), 3–28.

Arifovic, J. (2000). Evolutionary algorithms in macroeconomic models. *Macroeconomic Dynamics*, *4* (3), 373–414.

Arifovic, J. ve Maschek, M. (2006). Revisiting individual evolutionary learning in the cobweb model–an illustration of the virtual spite-effect. *Computational economics*, *28* (4), 333–354.

Austin, M. P.; Bates, G.; Dempster, M. A. H.; Leemans, V. ve Williams, S. N. (2004). Adaptive systems for foreign exchange trading. *Quantitative Finance*, *4* (5),

37-45. Retrieved from http://www.tandfonline.com/doi/abs/10.1080/14697680400008593 doi: 10.1080/14697680400008593

Bagchi, A. ve Saroop, A. (2003). Internet auctions: Some issues and problems. *Distributed Computing-IWDC 2003*, (), 836–836.

Banzhaf, W.; Nordin, P.; Keller, R. E. ve Francone, F. D. (1997). *Genetic programming : An introduction : On the automatic evolution of computer programs and its applications (the morgan kaufmann series in artificial intelligence)*. Morgan Kaufmann Publishers. Hardcover.

Bauer, R. J. (1994). *Genetic algorithms and investment strategies*. New York, NY, USA: John Wiley & Sons, Inc.

Bolland, P. ve Connor, J. (1997). A constrained neural network kalman filter for price estimation in high frequency financial data. *International Journal of Neural Systems*, *8* (04), 399–415.

Brameier, M. F. ve Banzhaf, W. (2006). *Linear genetic programming (genetic and evolutionary computation)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Brock, W.; Lakonishok, J. ve LeBaron, B. (1992, December). Simple technical trading rules and the stochastic properties of stock returns. *Journal of Finance*, *47* (5), 1731-64. Retrieved from http://ideas.repec.org/a/bla/jfinan/v47y1992i5p1731-64.html

Cai, W.; Pacheco-Vega, A.; Sen, M. ve Yang, K. (2006). Heat transfer correlations by symbolic regression. *International Journal of Heat and Mass Transfer*, *49* (23), 4352–4359.

Chen, S. (2002). *Evolutionary computation in economics and finance*. Physica-Verlag HD. Retrieved from http://books.google.de/books?id=VhgM33yDFoQC

Chen, S. ve Ni, C. (2000). Simulating the ecology of oligopolistic competition with genetic algorithms. *Knowledge and Information Systems*, *2* (3), 285–309.

Chen, S. ve Yeh, C. (1996). Bridging the gap between nonlinearity tests and the efficient market hypothesis by genetic programming. In *Computational intelligence for financial engineering, 1996., proceedings of the ieee/iafe 1996 conference on* (pp. 34–40).

Chen, S. ve Yeh, C. (2000). Simulating economic transition processes by genetic prog-

ramming. *Annals of Operations Research*, *97* (1), 265–286.

Chen, S.-H. (2005). Computational intelligence in economics and finance: Carrying on the legacy of herbert simon. *Information Sciences*, *170* (1), 121 - 131. Retrieved from http://www.sciencedirect.com/science/article/pii/S0020025503004444 doi: 10.1016/j.ins.2003.11.006

Churcher, C. (2007). *Beginning database design: From novice to professional*. Berkeley, CA, USA: Apress.

Colby, R. (2003). *The encyclopedia of technical market indicators*. McGraw-Hill. Retrieved from http://books.google.com/books?id=f82LUFQVGOUC

Dempster, M. ve Jones, C. (2002). Can channel pattern trading be profitably automated? *The European Journal of Finance*, *8* (3), 275–301.

Dempster, M. ve Leemans, V. (2006). An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications*, *30* (3), 543 - 552. Retrieved from http://www.sciencedirect.com/science/article/pii/S0957417405003015 (Intelligent Information Systems for Financial Engineering) doi: 10.1016/j.eswa.2005.10.012

Dempster, M. A. H. ve Jones, C. M. (2001). A real-time adaptive trading system using genetic programming. *Quantitative Finance*, *1* (4), 397-413. Retrieved from http://ideas.repec.org/a/taf/quantf/v1y2001i4p397-413.html

Dixon, H. (2001). *Surfing economics: Essays for the inquiring economist*. Palgrave Macmillan. Retrieved from http://books.google.com/books?id=aPhMJC_tC5wC

Dorsey, R. ve Mayer, W. (1995). Genetic algorithms for estimation problems with multiple optima, nondifferentiability, and other irregular features. *Journal of Business & Economic Statistics*, *13* (1), 53–66.

Duffy, J. ve Feltovich, N. (1999). Does observation of others affect learning in strategic environments? an experimental study. *International Journal of Game Theory*, *28* (1), 131–152.

Fama, E. F. (1970, May). Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, *25* (2), 383-417. Retrieved from http://ideas.repec.org/a/bla/jfinan/v25y1970i2p383-417.html

Fischer, M. M. ve Leung, Y. (1998). A genetic-algorithms based evolutionary computational neural network for modelling spatial interaction data. *The Annals of Regional Science*, *32* (3), 437–458.

Frantz, R. (2003). Herbert Simon. Artificial intelligence as a framework for understanding intuition. *Journal of Economic Psychology*, *24* (2), 265–277+. Retrieved from http://www.sciencedirect.com/science/article/B6V8H-47YR8Y1-1/2/1789665d8cf3ea6492d067409e62f128

Fyfe, C.; Marney, J. P. ve Tarbert, H. (2005). Risk adjusted returns from technical trading: a genetic programming approach. *Applied Financial Economics*, *15* (15), 1073–1077.

Gagné, C. ve Parizeau, M. (2002, July). Open BEAGLE: A new versatile C++ framework for evolutionary computation. In E. Cantú-Paz (Ed.), *Late breaking papers at the genetic and evolutionary computation conference (GECCO-2002)* (pp. 161–168). New York, NY: AAAI. Retrieved from http://vision.gel.ulaval.ca/en/publications/Id_43/PublDetails.php

Gagné, C. ve Parizeau, M. (2006). Genericity in evolutionary computation software tools: Principles and case-study. *International Journal on Artificial Intelligence Tools*, *15* (2), 173–194.

Gatti, D.; Guilmi, C.; Gaffeo, E.; Giulioni, G.; Gallegati, M. ve Palestrini, A. (2005). A new approach to business fluctuations: heterogeneous interacting agents, scaling laws and financial fragility. *Journal of Economic Behavior & Organization*, *56* (4), 489–512.

Gen, M. ve Cheng, R. (1999). *Genetic algorithms and engineering optimization* (Vol. 7). Wiley-interscience.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Gustafson, S.; Burke, E. K. ve Krasnogor, N. (2005). On improving genetic programming for symbolic regression. In *Evolutionary computation, 2005. the 2005 ieee congress on* (Vol. 1, pp. 912–919).

Harrald, P. ve Fogel, D. (1996). Evolving continuous behaviors in the iterated prisoner's dilemma. *Biosystems*, *37* (1), 135–145.

Hemberg, E.; Ho, L.; O'Neill, M. ve Claussen, H. (2011). A symbolic regression approach to manage femtocell coverage using grammatical genetic programming. In *Gecco'11: Proceedings of the 13th annual conference companion on genetic and evolutionary computation.*

Hennessy, J. ve Patterson, D. (2006). *Computer architecture : A quantitative approach* (4th ed.). Morgan Kaufmann.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence*. Cambridge, MA, USA: MIT Press.

Isasi, P.; Quintana, D.; Sáez, Y. ve Mochón, A. (2007). Applied computational intelligence for finance and economics. *Computational Intelligence*, *23* (2), 111-116.

JP Marney, H. T. D. M., Colin Fyfe. (2001, April). *Risk adjusted returns to technical trading rules: a genetic programming approach* (Computing in Economics and Finance 2001 No. 147). Society for Computational Economics. Retrieved from `http://ideas.repec.org/p/sce/scecf1/147.html`

Kaboudan, M. (1999). A measure of time series' predictability using genetic programming applied to stock returns. *Journal of Forecasting*, *18* (5), 345–357.

Kaboudan, M. (2000). Genetic programming prediction of stock prices. *Computational Economics*, *16* (3), 207–236.

Kaboudan, M. (2001). Compumetric forecasting of crude oil prices. In *Evolutionary computation, 2001. proceedings of the 2001 congress on* (Vol. 1, pp. 283–287).

Kaboudan, M. (2005). Extended daily exchange rates forecasts using wavelet temporal resolutions. *New Mathematics and Natural Computation*, *1* (01), 79–107.

Keijzer, M. (2004). Scaled symbolic regression. *Genetic Programming and Evolvable Machines*, *5* (3), 259–269.

Keijzer, M.; Merelo, J.; Romero, G. ve Schoenauer, M. (2002). Evolving objects: A general purpose evolutionary computation library. In P. Collet; C. Fonlupt; J.-K. Hao; E. Lutton ve M. Schoenauer (Eds.), *Artificial evolution* (Vol. 2310, p. 829-888). Springer Berlin / Heidelberg.

Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection (complex adaptive systems)*. MIT Press. Hardcover.

Kreibich, J. A. (2010). *Using sqlite* (1st ed.). O'Reilly Media, Inc.

Kronberger, G.; Wagner, S.; Kommenda, M.; Beham, A.; Scheibenpflug, A. ve Affenzeller, M. (2012). Knowledge discovery through symbolic regression with heuristiclab. *Machine Learning and Knowledge Discovery in Databases*, (), 824–827.

Kuhlmann, H. ve Hollick, M. (1995, May). *Genetic programming in C/C++*. Retrieved from `http://www.cis.upenn.edu/~hollick/genetic/paper2.html` (CSE99/CIS899 Final Report)

Langdon, W. B. ve Poli, R. (2002). *Foundations of genetic programming*. Springer-Verlag. Retrieved from `http://www.cs.ucl.ac.uk/staff/W.Langdon/FOGP/`

LeBaron, B. (2002). Building the santa fe artificial stock market. *Physica A*, (), .

Lettau, M. (1997). Explaining the facts with adaptive agents: The case of mutual fund flows. *Journal of Economic Dynamics and Control*, *21* (7), 1117–1147.

Levich, R. M. ve Thomas, L. I. (1993, October). The significance of technical trading-rule profits in the foreign exchange market: a bootstrap approach. *Journal of International Money and Finance*, *12* (5), 451-474. Retrieved from `http://ideas.repec.org/a/eee/jimfin/v12y1993i5p451-474.html`

Lew, T.; Spencer, A.; Scarpa, F.; Worden, K.; Rutherford, A. ve Hemez, F. (2006). Identification of response surface models using genetic programming. *Mechanical Systems and Signal Processing*, *20* (8), 1819–1831.

Li, J. ve Tsang, E. P. (1999). Improving technical analysis predictions: an application of genetic programming. In *Proceedings of the 12th international florida ai research society conference, orlando, florida* (pp. 108–112).

Lo, A. (2004). The adaptive markets hypothesis: Market efficiency from an evolutionary perspective. *Journal of Portfolio Management, Forthcoming*, (), .

Lui, Y.-H. ve Mole, D. (1998, June). The use of fundamental and technical analyses by foreign exchange dealers: Hong kong evidence. *Journal of International Money and Finance*, *17* (3), 535-545. Retrieved from `http://ideas.repec.org/a/eee/jimfin/v17y1998i3p535-545.html`

Luke, S. (2009). *Essentials of metaheuristics*. (available at http://cs.gmu.edu/∼sean/book/metaheuristics/)

Malliaris, M. ve Salchenberger, L. (1993). A neural network model for estimating option

prices. *Applied Intelligence*, *3* (3), 193–206.

Marimon, R.; McGrattan, E. ve Sargent, T. (1990). Money as a medium of exchange in an economy with artificially intelligent agents. *journal of Economic Dynamics and Control*, *14* (2), 329–373.

McCarthy, J. (1965). *LISP 1.5 programmer's manual*. The MIT Press.

McCorduck, P. (2004). *Machines who think : a personal inquiry into the history and prospects of artificial intelligence*. Natick, Mass: A.K. Peters.

Michell, M. (1998). *An introduction to genetic algorithms*. Mit Press. Retrieved from `http://books.google.com.tr/books?id=0eznlz0TF-IC`

Neely, C.; Weller, P. ve Dittmar, R. (1997, December). Is technical analysis in the foreign exchange market profitable? a genetic programming approach. *Journal of Financial and Quantitative Analysis*, *32* (04), 405-426. Retrieved from `http://ideas.repec.org/a/cup/jfinqa/v32y1997i04p405-426_00.html`

Neely, C. J. ve Weller, P. A. (1999). Technical trading rules in the european monetary system. *Journal of International Money and Finance*, *18* (3), 429–458.

Neely, C. J. ve Weller, P. A. (2001). Predicting exchange rate volatility: Genetic programming vs. garch and risk metrics™. *Federal Reserve Bank of St. Louis Working Paper Series*, (), .

Neely, C. J.; Weller, P. A. ve Ulrich, J. M. (2009). The adaptive markets hypothesis: evidence from the foreign exchange market. *Journal of Financial and Quantitative Analysis*, *44* (02), 467–488.

Newman, C. (2004). *Sqlite (developer's library)*. Indianapolis, IN, USA: Sams.

NYIF. (1989). *Technical analysis: A personal seminar*. New York Institute of Finance. Retrieved from `http://books.google.com/books?id=R57WAAAAMAAJ`

Owens, A. J.; Walsh, M. J. ve Fogel, L. J. (1966). *Artificial intelligence through simulated evolution*.

Palmer, R.; Brian Arthur, W.; Holland, J.; LeBaron, B. ve Tayler, P. (1994). Artificial economic life: a simple model of a stockmarket. *Physica D: Nonlinear Phenomena*, *75* (1), 264–274.

Persky, J. (1995). Retrospectives: The ethology of homo economicus. *The Journal of Economic Perspectives*, *9* (2), 221–231.

Poli, R.; Langdon, W. B. ve McPhee, N. F. (2008). *A field guide to genetic programming*. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk. Retrieved from http://www.gp-field-guide.org.uk (With contributions by J. R. Koza)

Poole, D. (1998). *Computational intelligence : a logical approach*. New York: Oxford University Press.

Rechenberg, I. (1973). Evolutionsstrategie–optimierung technisher systeme nach prinzipien der biologischen evolution.

Russell, S. ve Norvig, P. (2003). *Artificial intelligence : a modern approach*. Upper Saddle River, N.J: Prentice Hall/Pearson Education.

Santini, M. ve Tettamanzi, A. (2001). Genetic programming for financial time series prediction. In *Proceeding of eurogp 2001, lncs 2038. berlin: Spinger* (pp. 361–370).

Saroop, A. ve Bagchi, A. (2002). Artificial neural networks for predicting final prices in ebay auctions. In *Proc. wits-2002, twelfth annu. workshop on information technology and systems* (pp. 19–24).

Schwefel, H.-P. (1965). Kybernetische evolution als strategie der experimentellen forschung in der strömungstechnik. *Master's thesis, Technical University of Berlin*, (), .

Searson, D. P.; Leahy, D. E. ve Willis, M. J. (2010). Gptips: an open source genetic programming toolbox for multigene symbolic regression. In *Proceedings of the international multiconference of engineers and computer scientists* (Vol. 1).

Silverberg, G. ve Verspagen, B. (1994). Learning, innovation and economic growth: a long-run model of industrial dynamics. *Industrial and Corporate Change*, *3* (1), 199–223.

Sivanandam, S. ve Deepa, S. (2007). *Introduction to genetic algorithms*. Springer-Verlag Berlin Heidelberg. Retrieved from http://books.google.com.tr/books?id=wonrLjj2GagC

Smith, A. (1863). *An inquiry into the nature and causes of the wealth of nations*. A. and C. Black.

Svingen, B. (2006). When lisp is faster than c. In *Gecco '06: Proceedings of the 8th*

annual conference on genetic and evolutionary computation (pp. 957–958). New York, NY, USA: ACM. doi: http://doi.acm.org/10.1145/1143997.1144168

Tang, S.; Michel, C. ve Larouche, P. (2012). Development of an explicit algorithm for remote sensing estimation of chlorophyll a using symbolic regression. *Optics Letters*, *37* (15), 3165–3167.

Taylor, M. P. ve Allen, H. (1992, June). The use of technical analysis in the foreign exchange market. *Journal of International Money and Finance*, *11* (3), 304-314. Retrieved from `http://ideas.repec.org/a/eee/jimfin/v11y1992i3p304-314.html`

Tsang, E. ve Martinez-Jaramillo, S. (2004). Computational finance. *IEEE Computational Intelligence Society Newsletter*, *3* (8), .

Tsang, E. P. K.; Li, J. ve Butler, J. M. (1998). Eddie beats the bookies. *Software-Practice and Experience*, *28* (10), 1033–1044.

Turing, A. (1950). Computing machinery and intelligence. *Mind*, *59* (236), 433–460.

Verna, D. (2006). How to make lisp go faster than c. In S. I. Ao; J.-A. Lee; O. Castillo; P. Chaudhuri ve D. D. Feng (Eds.), *Imecs* (p. 815-820). Newswood Limited.

Vladislavleva, E. J.; Smits, G. F. ve Den Hertog, D. (2009). Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *Evolutionary Computation, IEEE Transactions on*, *13* (2), 333–349.

Von Neumann, J. ve Morgenstern, O. (1945). Theory of games and economic behavior. *Bull. Amer. Math. Soc*, *51* (), 498–504.

Vriend, N. (1995). Self-organization of markets: An example of a computational approach. *Computational Economics*, *8* (3), 205–231.