12. BÖLÜM / CHAPTER 12

GENETİK PROGRAMLAMAMANIN TEMELLERİ: GENETİK PROGRAMLAMA VE EKONOMETRİ İLİŞKİSİ BAĞLAMINDA BİR İNCELEME

BASICS OF GENETIC PROGRAMMING: AN OVERVIEW IN THE CONTEXT OF THE RELATIONSHIP BETWEEN GENETIC PROGRAMMING AND ECONOMETRICS

Seda KARAKAŞ GEYİK*, Mehmet Hakan SATMAN**

*Dr. Öğr. Üyesi, İstanbul Üniversitesi, İktisat Fakültesi, Ekonometri Bölümü, İstanbul, Türkiye
E-mail: kseda@istanbul.edu.tr

**Prof. Dr., İstanbul Üniversitesi, İktisat Fakültesi, Ekonometri Bölümü, İstanbul, Türkiye
E-mail: mhsatman@istanbul.edu.tr

DOI: 10.26650/B/SS10.2021.013.12

ÖZ

Bu çalışmanın temel amacı genetik programlamanın temelleri ve ekonometri alanındaki uygulamaları hakkında bir değerlendirme sunmaktır. Genetik programlama, sağlanan girdiler ve çıktılar arasında matematiksel gösterimler veya bilgisayar programları üretmeyi amaçlayan evrim temelli bir arama tekniğidir. Çalışmada genetik programlama-ekonometri ilişkisi bağlamında okuyucuya derin bir kavrayış kazandırma motivasyonu ile öncelikle genetik programlamanın tanımı ve gelişimine ilişkin kapsamlı bir literatür taramasına yer verilmiş ve ardından genetik programlamanın çalışma prensipleri tanıtılmıştır. Genetik programlamanın mühendislik, elektronik, görüntü işleme, işaret işleme, işletme, ekonomi, finansal zaman serileri, tıp, biyoloji, eğlence ve bilgisayar oyunları gibi pek çok alanda insanla rekabet edebilir nitelikte, yenilikçi çözümler üretme yeteneği ortaya konulmuştur. Ekonometrik problemlerin nasıl bir genetik programlama problemi olarak düşünülebileceğine ilişkin örnekler ile ekonometrik uygulamalar ve genetik programlama arasındaki ilişkiler özellikle sembolik regresyon çerçevesinde tartışılmıştır. Ayrıca model seçimi, değişken seçimi, finansal zaman serilerinin modellenmesi ve tahmini, finansal zaman serilerinde kural arama, parametreler için anlamlılık testleri konularında genetik programlamanın alternatif olarak çözüm sunabileceği durumlar da ortaya konulmuştur. Çalışmanın sonucunda tartışılan kavramlar bir arada değerlendirildiğinde, ekonometrik modelleme ve tahmin süreçlerinde araştırmacıların modelin fonksiyonel biçimi ve (veya) değişkenleri hakkında bilgisinin olmadığı ya da yetersiz olduğu durumlarda genetik programlamanın geleneksel yöntemlere alternatif olarak bir çözüm sunma konusunda başarılı olabileceği ifade edilmiştir.

Anahtar Kelimeler: Genetik Programlama, Evrimsel Programlama, Sembolik Regresyon, Yapay Zekâ, Model Seçimi, Değişken Seçimi

Extended Abstract

This study provides a short introduction and an overview of the basics of genetic programming (GP) and its applications in the context of econometrics. In other words, the aim of this study is to gain a deep insight into the relationship between GP and econometrics. Accordingly, first, the basic definition and development of GP are introduced. After the literature review, GP applications in the areas of engineering, electronics, image processing, signal processing, business, economics, financial time series, medicine, biology, entertainment, and computer games are presented. In this part, we aim to highlight the results of GP applications that produce human-competitive results and new inventions, followed by working principles of GP (the basic representation schemes, initialization, and operators used in GP). Then, we discuss how econometric problems can be handled as a GP problem. We showed that GP can successfully solve several econometric problems by producing new inventions and human-competitive results.

GP is a family of searching tools in which the basic aim is to generate mathematical expressions or computer programs between the inputs fed and output. In other words, GP explores the rules in terms of computer programs using given input and output data via evolutionary optimization. GP starts with a population of randomly chosen mathematical expressions or computer programs. The quality of an expression is measured by the fitness or cost, which indicates the difference between the generated and real outputs. Then, the best solutions are recombined to generate offspring. An offspring is a cut and paste copy of the two best candidate solutions. A good offspring is expected by recombining two fittest parents. After generating new candidates, mutations are applied on the constants or functions with small probabilities. Mutations result in small and significant changes depending on the location of the mutated elements in expressions. Small changes perform a local search, whereas significant changes are essential for searching other sides of the search space in terms of optimization. After several steps, generated expressions form a new population. The search process terminates after the meeting the stopping criteria.

GP develops computer programs by optimization, whereas genetic algorithms are generally used for problems in which the main objective is to find a global extremum of a function. Genetic algorithms encode decision variables using binary, integer, permutation, or floating-point encoding–decoding schemes, whereas GP generally uses abstract syntax trees, postfix and prefix representations, Java bytecodes, or assembly language opcodes. Because GP develops mathematical expressions and programs, GP tasks are computer-aided, and an interpreter is always needed to evaluate generated offspring for calculating fitness values in runtime.

In econometrics; the tasks of estimating model parameters, testing the significance of parameters, and performing forecasts are nontrivial. In numerous cases, researchers face difficulties in selecting the best model, functional structure, correct combination of lagged values or independent variables, etc. The problems of model and variable selections are handled separately or together in the literature. When the number of independent variables in variable-pool increases, selecting the k correct variables among K variables results in a combinatorial problem, where $k \leq K$. Furthermore, when the functional form is undefined by theory, nothing can be done, except trying all possible forms with unknown variables. This is a “searching the needle in the haystack” type of problem. GP is broadly used in such problems under the title of symbolic regression. In general terms, the main question of symbolic regression is: “what is the correct functional form with correct independent variables that explains a dependent variable well?”. Besides the estimation process and techniques used, numerous model suggestions can satisfy a model selection criteria used in the classic econometric literature.

Econometric models are generally represented by mathematical expressions. In programming languages, mathematical expressions are also computer programs. For example, piecewise functions are basically represented using if-then-else statements or expressions in several programming languages. Similarly, piecewise regressions, nonlinear time-series models, as well as buy and hold strategies in finance are combinations of mathematical expressions including decisions as computer programs. Developing the best strategy and determining the best time for buying a stock is a well-studied topic in the GP literature.

GP is neither the end of the whole story nor the magic that solves problems. First, GP tends to generate more than sufficient expressions as solutions; several researchers have tried deciphering this phenomenon. Second, an automatic process can almost always find an overfitted set of solutions describing the relations between variables. As in other econometric studies, researchers should be attentive to select interpretable solutions to the concerned problem. In other words, the model that minimizes selection criteria may not be the proper model. Moreover, GP is successful in model and variable selections, rule detection, discovering new location-comparison tests, and inventing patentable things in a human-competitive way.

Keywords: Genetic Programming, Evolutionary Programming, Symbolic Regression, Artificial Intelligence, Model Selection, Variable Selection

1. Giriş

Bilgisayarların problemleri otomatik olarak çözmesini sağlama fikri yapay zekâ, makine öğrenmesi ve Alan Turing tarafından *makine zekâsı* (*machine intelligence*) olarak adlandırılan geniş kapsamlı araştırma alanlarının merkezinde yer almaktadır (Koza, 2010: 252). Tarihte yapay zekâ ve makine öğrenmesi ile ilgili ilk örnekler Yunan mitolojisinde teknoloji ve artisan tanrısı Hephaestus ile mimar Pygmalion'un akıllı robotu olan Talos'tan (İnce, 2017: 16), El-Cezeri'nin (1206) kitabında bilgisayarsız olarak tasarlanmış örneklerine rastlanan programlı robotlara kadar (Adalı, 2017: 11) eskilere dayansa da insanoğlunun yapay zekâ kavramına karşı algısı zaman içerisinde teknolojik gelişmeler ışığında sürekli değişime uğramıştır.

Günümüzde kullandığımız modern bilgisayarlar ve yapay zekâ düşüncesinin öncüsü olarak ise İngiliz bilim insanı Alan Turing'in çalışmalarının kabul edildiği yönünde literatürde yaygın bir fikir birliği vardır. Turing, 1936 yılında *London Mathematical Society* bildirilerinde “*On Computable Numbers, With An Application to the Entscheidungsproblem*” başlıklı çalışmasını yayımlamıştır. Bu çalışmada hesaplama kavramının tanımını yaparken günümüzde *Turing Makinesi* olarak adlandırılacak soyut bir hesaplama aracı olan kavramın da kuramsal temellerini atmıştır (Turing, 1936). Turing, 1948 yılında yayımladığı “*Intelligent Machinery*” adlı çalışmada akıllı makineler elde etmek için evrim ve doğal seleksiyonun kullanılabileceği fikrini ortaya attı. Bu çalışmada akıllı bilgisayar programları yaratmak için üç yaklaşım tanımlamıştır. Bu yaklaşımlardan ilki mantık temelli, ikincisi bilgi temelli iken üçüncü yaklaşım genetik ve evrimsel araştırmalar temeline dayanmaktadır (Koza, Keane, Yu, Bennet ve Mydlowec, 2000: 121-122).

Turing'in 1950 yılında *Mind* dergisinde yayımladığı “*Makineler düşünebilir mi?*” gibi çarpıcı bir söylem ile başlayan “*Computing Machinery and Intelligence*” başlıklı çalışması belki de günümüz dijital bilgisayarlarının ve yapay zekâ fikrinin temellerinin atıldığı devrimci bir metindir. Turing (1950), “*Computing Machinery and Intelligence*” başlıklı bu çalışmada akıllı bilgisayar programlarının otomatik olarak oluşturulmasında evrim ve doğal seleksiyon süreçlerinin nasıl kullanılacağını ortaya koymuştur (Koza v.d., 2000:122). Akıllı makinelerin tasarımı için yetişkin bir insanın modellenmesi yerine öncelikle bir çocuğun modellenmesini uygun bulan Turing, bu tasarımı “çocuk program” ve “öğrenme süreci” olarak iki sürece bölerek tanımlar. İlk denemede elde edilecek sonucun en iyisi olmadığını ve belli bir öğrenme süreci yardımı ile daha iyi ya da kötü sonuçların da denenmesini vurgular. Söz konusu öğrenme süreci ile evrim arasında aşağıda verilen tanımlar çerçevesinde ilişkiler olduğunu ileri sürer (Turing, 1950: 456).

Çocuk Makinenin Yapısı ↔ Kalıtsal Materyal

Çocuk Makinedeki Değişimler ↔ Mutasyon

Doğal Seleksiyon ↔ Araştırmacının Yargıları

Böylece 1948 ve 1950 yılındaki araştırmaları ile Alan Turing *makine zekâsı* kavramının bir bilgisayar programının (*kalıtsal materyalin*) doğal seleksiyon güdümünde aşamalı bir değişime uğraması (*mutasyon*) şeklinde evrimsel bir süreç ile elde edilebileceğini ortaya koymuştur. Ancak bu evrimsel süreç içerisinde bireylerden oluşan bir topluluğu ya da bireylerin herhangi bir çaprazlama (*genlerin kombinasyonu*) benzeşimini (*sexual recombination*) öngörmemiştir (Koza, 2010: 252).

Turing'in bu genetik temelli yaklaşımı son birkaç on yılda giderek popülerlik kazanan evrimsel hesaplama yöntemlerinin temel mantığı ile ortaktır. Bu yöntemler arasında *evrim stratejileri* (*evolution strategies*), *evrimsel programlama* (*evolutionary programming*), *genetik algoritmalar* (*genetic algorithms*), *genetik sınıflandırma sistemleri* (*genetic classifier systems*), *genetik programlama* (*genetic programming*) ve *evrimleşebilen donanımlar* (*evolvable hardware*) gibi konular sayılabilir (Koza v.d., 2000: 122). Bu çalışma, evrimsel hesaplama tekniklerinden genetik programlama konusuna odaklanmaktadır. Çalışmanın amacı genetik programlamanın temellerini ortaya koymak ve ekonometri alanındaki uygulamaları bağlamında bir değerlendirme sunmaktır.

2. Genetik Programlamanın Gelişimi

Doğadaki evrim metaforunu taklit eden bir stokastik arama yöntemi olarak tanımlanabilecek genetik algoritmaların teori ve pratiği konusunda öncü çalışma olarak John Holland'ın 1975 yılında yayımladığı "*Adaptation in Natural and Artificial Systems*" adlı eseri olarak gösterilmektedir. Holland (1975) bu eserde "*genlerin kombinasyonu*" kavramını resmileştirmiş ve bireylerden oluşan bir topluluk fikrini kullanmıştır (Koza, 2010: 252). Matematiksel problemleri ve mühendislikte optimizasyon sorunlarını çözmek için evrimsel sürecin bir benzeşiminin nasıl uygulanabileceğini ortaya koymuştur. Holland (1975) çalışmasına göre günümüzde genetik algoritmalar olarak anılan bu kavram genetik olarak bir dizi nesil sonucunda üremiş bireylerden oluşan topluluk arasından en iyi ya da en iyiye yakın çözümü bulmayı amaçlar. Burada topluluktaki her bir birey verilen problem için bir aday çözüm sunar. Genetik algoritmalar, bu aday çözümlerin (*bireylerin*) oluşturduğu başlangıç topluluğunu üreme, çaprazlama ve mutasyon işlemleri uygulayarak yeni bir topluluğa dönüştürür (Koza, 1997:1).

Genetik algoritmaların yaygın olarak kullanılmaya başlanması ise daha çok John Holland'ın doktora öğrencisi olan David E. Goldberg'in (1989) genetik algoritmaların pratik uygulamalarına yer verdiği eseri "*Genetic Algorithms in Search, Optimisation and Machine Learning*" sonrasında gerçekleşmiştir. Genetik algoritmaların gelişimine katkı sağlayan başlıca çalışmalar arasında Michalewicz'in (1992) "*Genetic Algorithms + Data Structures = Evolution Programs*" başlıklı ve Mitchell'in (1996) "*An Introduction to Genetic Algorithms*" başlıklı eserleri sayılabilir. Aynı zamanda genetik algoritmaların finansal uygulamadaki yeri için Bauer (1994), görüntü segmentasyonu konusunda Bhanu ve Lee (1994), örüntü tanımlama için Pal ve Wang (1996), simülasyon ve modelleme uygulamaları için Stender, Hillebrand ve Kingdon (1994), kontrol ve işaret işleme alanındaki uygulamalar için Man, Tang, Kwong ve Halang (1997) ve mühendislik dizaynı için Gen ve Cheng (1997) çalışmaları incelenebilir.

Genetik algoritmaların bir uzantısı olarak kabul edilebilecek genetik programlama bilgisayar programlarının evrimleştirilmesi yolu ile bir problemi çözmeyi ya da yaklaşık çözümünü elde etmeyi amaçlayan alandan bağımsız bir problem çözme yaklaşımıdır. Bu yaklaşım genetik algoritmalar ile benzer şekilde çaprazlama ve mutasyon gibi doğal olarak meydana gelen genetik işlemlerin benzeşimine ve bu işlemlerin sonucunda Darwin'in ortama uyum sağlayanın yüksek olasılıkla hayatta kalması ilkesine dayanan bir yöntemdir (Koza, 1997:1).

Genetik programlama bağlamında ilk çalışmalar 1980'lerin başlarına dayanmaktadır. Smith'in (1980) yayımladığı genetik algoritmalar temelli bir öğrenme sistemi, Forsyth'in (1981) İngiltere polisi için önerdiği adli bilimlerde karar kuralları üreten bir bilgisayar programı paketi bu çalışmalara örnek gösterilebilir. 1985 yılında Michael L. Cramer'in sunduğu "*A Representation for the Adaptive Generation of Simple Sequential Programs*" başlıklı eserin ise genetik programlama alanındaki ilk makale olarak kabul edildiği yönünde yaygın bir görüş birliği vardır. Cramer (1985) bu çalışmasında ağaç benzeri gösterimi ve genetik algoritmalarla programları çalıştırmada kullanılan operatörleri tanımlamıştır. Bu alandaki öncü çalışmalardan belki de en önemlisi ise John R. Koza tarafından 1992 yılında yayımlanan "*Genetic Programming: On the Programming of Computers by Means of Natural Selection*" başlıklı eserdir.

Koza (1992) bu temel eserinde, yapay zekâ, makine öğrenmesi, uyarlamalı sistemler, otomatik öğrenme alanlarındaki bütün problemlerin aslında birer uygun bilgisayar programı arama süreci olarak görülebileceğini ve genetik programlamanın çözüm için ortak bir yöntem sunacağını göstermiştir. Bu yöntem potansiyel çözümlerin LISP programlama dilinde ağaç

gösterimi ile çalıştırılabilir yapılar olarak ifade edilmesi ile genetik algoritmalar ve diğer evrimsel algoritmalarından farklılaşmaktadır (Koza, 1992; Wu ve Banzhaf, 1998). Genetik algoritmalar ile genetik programlama arasındaki temel fark temsil ettikleri yapının gösterimi ve bu gösterimin anlamıdır. Genetik algoritmalar genellikle sabit uzunluktaki ikili dizi topluluğunda çalışırken genetik programlama genellikle bilgisayar programlarınınca temsil edilen bir *ayırışma ağacı* (*parse tree*) topluluğunda çalışır (Wong ve Leung, 2002: 41). Bu özelliği sayesinde genetik programla çok geniş bir uygulama alanına sahiptir.

Koza (1992)'nin öncü çalışmasının yanı sıra Banzhaf, Nordin, Keller ve Francone (1997) "*Genetic Programming – An Introduction: On the automatic evolution of computer programs and its applications*", Langdon (1998) "*Genetic Programming and Data Structures*", Langdon ve Poli (2002), "*Foundations of Genetic Programming*", Poli, Langdon ve McPhee (2008) "*A field guide to genetic programming*" eserleri genetik programlama alanında diğer temel eserlerdendir. Bu temel eserlerden günümüze genetik programlama alanında çalışmalar önemli ölçüde artış göstermiştir. Aralarında genetik programlama teori ve uygulamalarındaki önemli gelişmelerin takip edebileceği *Genetic Programming and Evolvable Machines*' in de bulunduğu pek çok sayıda akademik dergi yayımlanmaya başlanmıştır. Diğer örnekleri için; *Transactions on Evolutionary Computation (IEEE)*, *Evolutionary Computation (MIT Press)* yayınları incelenebilir. Aynı zamanda genetik programlama ve evrimsel algoritmalar alanında düzenlenen *Genetic and Evolutionary Computation Conference (GECCO)*, *Congress on Evolutionary Computation (CEC)*, *Parallel Problem Solving from Nature (PPSN)*, *European Conference on Genetic Programming (EuroGP)* gibi önemli konferanslar da bu alandaki gelişmelere olanak sağlamaktadır (Affenzeller, Winkler, Wagner ve Beham, 2009: 63).

3. Genetik Programlamanın Kullanım Alanları

Genetik programlama, değişkenler arası karşılıklı ilişkiler bilinmediğinde ya da tam olarak anlaşılmadığı durumlarda veya bilinen mevcut kavrayışın yanlış olabileceğinden şüphelenildiği durumlarda kullanılmaktadır. Nihai çözümün büyüklüğünün ve şeklinin tespit edilmesi problemin önemli bir parçası olması halinde çözüm için genetik programlamayı seçmek uygun olacaktır. Veri ve modelin matematiksel analizlere elverişli olmadığı ve analitik bir çözüm sağlanamadığı durumlar ise yine genetik programlamanın kullanım alanlarından bazıları olarak karşımıza çıkmaktadır (Poli, v.d., 2008: 111-113).

1980'li yıllardan bu yana bilgisayarların işlemci hızları önemli miktarda artış göstermiştir. Bu durum hesaplama yoğun bir teknik olan genetik programlamanın uygulama alanlarının da giderek genişlemesine neden olmaktadır. Günümüzde genetik programlama veri temelli

modelleme, elektronik tasarımlar, görüntü ve işaret işleme, ekonomik modelleme ve zaman serileri tahminleme, endüstriyel süreç kontrolü, tıp biyoloji ve biyoinformatik, eğlence ve bilgisayar oyunları gibi alanlarda insanla rekabet edebilir çözümler sunmaktadır. Yapay zekâ ve makine öğrenmesi gibi kavramların odağında makinelerin insan benzeri çözümler üretmesi vardır. Koza, Bennett ve Stiffelman (1999) ise *zekâ (intelligence)* kavramının insanla rekabet edebilirlik (*human competitiveness*) kavramına evrilmesini önermişlerdir. Aynı zamanda elde edilen bir sonucun insan ile rekabet edilebilir (*human competitive*) olarak kabul edilmesi için aralarında patentlenebilir bir buluş niteliğinde olması, hakemli bir dergide yayımlanmış olması, alanında yeni bir sonuç olarak nitelendirilmesi ve eski çalışmalardan ayrılmış olması gibi özelliklerin yer aldığı bazı özellikleri taşıması gerektiği tanımlanmıştır. Bu alanda ilk kez 2004 yılında Seattle'daki *Genetic and Evolutionary Computation Conference (GECCO-2004)* konferansında "*Humies*" adı verilen ödüllü bir yarışma düzenlenmiştir.

2004'ten günümüze "*Human Competitive Awards*" olarak adlandırılan ödüllü yarışma genetik ve evrimsel hesaplama alanında elde edilen sonuçların insan ile rekabet edebilir olma özelliği kapsamında değerlendirildiği bir oluşumdur. Söz konusu yarışmada yarışan ve ödülleri kazanan araştırmaların önemli bir oranı genetik programlama konusunda uygulamalardan oluşmaktadır. Örneğin; 2004 yılında altın madalya, Lohn, Hornby ve Linden'in (2004) NASA'nın Uzay Teknolojisi 5 (NASA's Space Technology 5 - ST5) misyonu için genetik programlama aracılığı ile otomatik bir x-bant anten dizaynı tasarladıkları çalışmaya verilmiştir. 2005 yılında ise dört madalya yine genetik programlama uygulamalarına verilmiştir. Bunlar; optik lens sistemlerinin icadı (Koza, Al-Sakran ve Jones, 2005), quantum Fourier algoritmasının evrimi (Massey, Clark ve Stepney, 2005), Core War için evrimleşen Assembly programları (*Assembly programs*) (Corno, Sanchez ve Squillero, 2005), tavla, satranç ve robocode gibi oyunlar için yüksek performanslı oyuncular (Azaria ve Sipper, 2005a; Hauptman ve Sipper, 2005; Shichel, Ziserman ve Sipper, 2005) çalışmalarıdır. 2006 yılında yine genetik programlama yardımı ile görüntü analizi konusunda bir araştırma olan Trujillo ve Olague (2006) altın madalyaya layık görülmüştür. 2004-2010 yılları arasındaki ödül alan çalışmaların detaylı bir literatür taraması için Koza (2010) çalışması incelenebilir. Koza (2010) bu çalışmada insan ile rekabet edebilir olma niteliklerine (*human competitive*) sahip 76 çalışmayı incelemiş ve bu yenilikçi çalışmalar arasından azımsanamayacak oranda bir kısmının genetik programlama aracılığı ile elde edildiği sonucuna varmıştır.

Hidroloji bilimi alanında da genetik programlama uygulamalarına sıklıkla rastlanmaktadır. Bu alandaki uygulamalara, Drecourt'un (1999) yağış akışının modellenmesinde, Arganis v.d.'nin (2009) İspanya'da Ebro ırmağının su sıcaklığının tahminlenmesinde, Chadalawada,

Havlicek ve Babovic'in (2017) havza akış dinamiklerinin belirlenmesinde ve Aryafar, Khosravi, Zarepourfard ve Rooki' nin (2019) Doğu İran Khezri ovası yeraltı suyu kalite parametrelerinin tahmininde genetik programlamayı kullandıkları çalışmaları örnek gösterilebilir.

Genetik programlamanın bir diğer uygulama alanı da finansal ekonomi ve ekonomik modelleme çalışmalarıdır. Genetik programlama, finansal zaman serileri gibi dinamik davranış gösteren yapıların davranışlarını karakterize eden sembolik bir yapısal model bulmakta tercih edilen yöntemler arasında olup bu nedenle finansal zaman serilerinin tahmini ve modellenmesi konusunda yaygın olarak kullanılmaktadır.

Finansal zaman serileri alanında genetik programlamanın kullanılması ile ilgili ilk çalışmalar ABD merkez bankasından araştırmacılar Neely, Weller ve Dittmar (1997), Neely ve Weller (1999, 2001), Neely (2003) çalışmalarına dayanır. Söz konusu çalışmalarda genetik programlama kullanılarak döviz piyasalarında gün içi işlemlerde piyasa üzeri getiri elde etmek için herhangi bir kanıt olmadığı ve döviz piyasasının etkin olduğu sonucuna ulaşılmıştır. Bu çalışmanın sonuçları literatürde Marney, Miller, Fyfe ve Tarbert (2001) tarafından eleştirilmiştir. Daha sonra Neely, Weller ve Ulrich (2006) kendilerini eleştiren çalışmaları destekleyecek nitelikte sonuç elde etmiş olup bulgularının etkin piyasalar hipotezinden daha çok Lo'nun adaptif piyasalar hipotezi (Lo, 2004) ile tutarlı olduğu sonucuna ulaşmışlardır. Kaboudan (1999) hisse senedi getirilerinin, Kaboudan (2000) hisse senedi fiyatlarının, Kaboudan (2005) uluslararası döviz kuru verilerinin tahminlenmesinde genetik programlamayı kullanmış ve üstünlüklerini ortaya koymuştur. Mousavi, Esfahanipour ve Zarandi (2014) ise portföy getirisini maksimize etme ve riski minimuma indirmeyi amaçlayan dinamik portföy işlem sistemi geliştirilmesinde genetik programlamayı kullanmıştır.

Mikroekonomi ve oyun teorisi alanında da genetik programlama uygulamalarına rastlamak mümkündür. Oyun teorisi, ekonomistlerin piyasaların yapısını anlamak için kullandıkları araçlar arasında yer almakta ve giderek artan biçimde sosyal sistemlerin simülasyonu ile desteklenmektedir. Söz konusu sosyal sistem simülasyonlarının bir parçası olarak da genetik programlamanın oyun teorisi alanındaki uygulamaları literatürde yer almaktadır. Chen, Duffy ve Yeh (2002), Jin (2005), Jin ve Tsang (2006) çalışmaları incelenebilir.

Genel anlamda evrimsel hesaplama yöntemlerinin ve özelinde genetik programlamanın veri bilimlerinin içerisinde de farklı alanlarda yaygın şekilde kullanımına rastlanmaktadır. Veri sınıflama ve veri madenciliği çalışmaları (Freitas, 2008), parametre tahmini (Dorsey ve Mayer, 1995), model keşfetme (Fischer ve Leung, 1998) alanları bu uygulamalara örnektir.

Ekonometri alanında ise *sembolik regresyon* (*symbolic regression*) genetik programlamanın temel uygulama alanlarından biridir. Sembolik regresyon, genetik programlamanın teorik temellerinin yer aldığı bir sonraki başlığın ardından beşinci bölümde detayları ile incelenecektir.

4. Genetik Programlamanın Temelleri

Genetik programlama, belirli bir problemi çözmek üzere rastlantısal oluşturulmuş matematiksel ifadelerin veya bilgisayar programlarının genetik operatörler ile değiştirilmesi prensibiyle çalışan bir arama yöntemidir. Bir topluluğu (*population*) oluşturan *aday çözümler* (*individuals, candidates*) birer matematiksel ifade veya bilgisayar programının belirli bir gösterimi şeklindedir. Söz konusu problemi çözme konusunda daha başarılı olan *aday çözümlerin seçilmesi* (*selection*), *çaprazlanması* (*crossover*) ve *mutasyona* (*mutation*) *uğratılması* ile problemi çözme konusunda daha başarılı olabilecek aday çözümlerin üretilmesi beklenir. Tekrarlanan nesiller (*generations*) sonucunda problemi çözen bir matematiksel ifadeye veya bir bilgisayar programına ulaşılması sağlanır.

Genetik programlamada işlem gören aday çözümler belirli sistemlere göre kodlanırlar. Söz konusu kodlama *prefix* (*ön-ek*), *postfix* (*son-ek*), Java bytecode (Orlov ve Sipper, 2009) veya makine dili olabilir. Biz insanlar olarak matematiksel ifadeleri *infix* (*orta-ek*) gösterimi ile kodlamayı tercih ediyoruz. Örneğin;

$$7 + 3 \times 5 \tag{4.1}$$

ifadesi orta-ek gösterimi ile yazılmıştır. Toplam ve çarpım operatörleri işlem görülecek operandların ortasında yer aldığı için bu isim kullanılmaktadır. Orta-ek gösteriminde işlem önceliği meselesi de önem kazanır. Örneğin eşitlik 4.1 ile verilen gösterimde öncelikle çarpım operatörü işlem görür. Çarpım operatörünün sonucu ise toplama operatörünün ikinci operandı olarak yer alır. Söz konusu ifade öncelikle $7 + 15$ ifadesine, sonrasında da 22 sayısına değerlendirilir. Parantezler kullanılarak işlem önceliği değiştirilebilir. Tüm bunların yanında orta-ek gösterimdeki işlemleri gerçekleştirmek bilgisayarlar için etkin olmaz. Bu yüzden *yorumlayıcı* (*interpreter*) ve *derleyiciler* (*compiler*) orta-ek gösterimini öncelikle ön-ek veya son-ek gösterimine dönüştürür. Son-ek gösterimi operandları takiben operatörlerin yazıldığı gösterime karşılık gelir. Eşitlik 4.1. ile verilen ifade

$$7\ 3\ 5\ \times\ + \tag{4.2}$$

şeklinde yazıldığında son-ek gösterimi söz konusu olur. Bu gösterimde işlemler soldan sağa doğru işlem önceliği gözetmeksizin gerçekleştirilir. Bunun sebebi, son-ek gösteriminin

işlem önceliğini tanımlı gereği içermesidir. Soldan sağa okuma sırasında görülen ilk operatör, kendinden önceki yeterli sayıdaki operanda uygulanır. Örneğin eşitlik 4.2 ile verilen ifadede soldan sağa okunduğunda ilk operatör çarpım olduğundan ifade $7 \ 15 +$ haline gelir. Son haliyle ifade soldan sağa okunduğunda ilk operatör toplam olduğundan öncesindeki iki operand ile çalıştırılır ve 22 sayısına ulaşılır. Matematiksel ifadelerin orta-ek gösteriminden son-ek gösterimine çevrilmesi için Shunting-yard gibi algoritmalara ihtiyaç duyulur (Dijkstra, 1961). Ayrıca *yığın (stack)* temelli yorumlayıcılarda bu gösterim etkin olarak uygulanır (Perkis, 1994).

Tüm bu gösterimler içinde ön-ek gösterimi özel bir yer tutar. Hem LISP (Tanimoto, 1987) (LISt Processing) programlama dili ailesinde tercih edilen gösterim olması hem de bu gösterimin ağaç gösterimi ile birebir ilişkili olması bu gösterimi ilginç kılmaktadır. Yukarıda 4.1 ve 4.2 eşitliklerinde değindiğimiz ifadeyi ön-ek gösterimini temel alan LISP ailesi dillerde

$$(+ \ 7 \ (\times \ 3 \ 5)) \quad (4.3)$$

olarak yazabiliriz. Bu ifadede $7, (\times \ 3 \ 5)$ ifadesinin değeri ile toplatılır. Görüldüğü üzere işlem önceliği meselesi ön-ek gösteriminde de doğallıkla çözülür. Operatörleri takiben gerekli sayıdaki operandın yer alması da bu gösterimin adının ön-ek gösterimi olmasının sebebi olmuştur. İlginçtir ki ön-ek gösterimi sayesinde ifadeler ağaç grafikleri ile kolaylıkla gösterilebilir. Söz konusu ifadenin toplama işlemini 7 ve $(\times \ 3 \ 5)$ operandlarına uyguladığını söylemiştik. 7 sayısı kendisine değerlendirirken $(\times \ 3 \ 5)$ ifadesi ise 15 sayısına değerlendirilir. Son olarak da toplam operatörü 7 ve 15 operandlarına uygulanır. Bu ifadenin ağaç gösterimi

Şekil 4.1. $(+ \ 7 \ (\times \ 3 \ 5))$ ifadesinin ağaç gösterimi

olur. İfade alttan üste doğru değerlendirir

Şekil 4.2. $(+ \ 7 \ (\times \ 3 \ 5))$ ifadesinin değerlendirme yoluyla indirgenmesinin ağaç gösterimi

ağacı elde edilir. Son işlem de 22 sayısına değeriyecektir. Söz konusu gösterim karmaşık ifadelerin gösterimde de rahatlıkla kullanılabilir. Örneğin $f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}$ ile tanımlanan standart normal dağılım fonksiyonunun ağaç gösterimi aşağıdaki gibi olur.

Şekil 4.3. Standart normal dağılım fonksiyonunun ağaç gösterimi

Söz konusu ağaç ile LISP gösterimi olan,

`(* (/ 1 (sqrt (* 2 π))) (pow e (* (/ -1 2)(pow z 2))))`

ifadesi arasında bilgi kaybetmeden dönüşüm mümkündür. Burada sqrt karekök, pow kuvvet, * çarpma, / bölme operatör ve fonksiyonlarına karşılık gelmektedir (LISP dilinde operatörler de birer fonksiyondur).

Genetik programlamada, başlangıçta oluşturulan aday çözümlerin seçilmesi, çaprazlanması ve mutasyona uğratılması ile yeni toplulukların oluşturulduğuna değinmiştik. Yukarıda değindiğimiz son-ek ve ön-ek gösterimlerinin, ifadelerin ayrıştırılması ve çaprazlanması bakımından orta-ek gösterime göre büyük oranda üstün olduğunu söyleyebiliriz. Bu bağlamda özellikle ön-ek gösteriminin ve bu gösterimi benimseyen LISP ifadelerinin genetik programlamada uygulanması daha çok tercih edilir. Aşağıdaki iki ağaç gösteriminin bir problemi çözmek üzere oluşturulmuş iki aday çözüme ait olduğunu varsayalım:

Şekil 4.4. İki aday çözüme ait ağaç gösterimi

Şekil 4.4 ile gösterilen iki ağacın sağ dalları kesilir ve kesilen parça diğer ağacın sağ dalı olarak bağlanırsa aşağıda Şekil 4.5 ile verilen ağaçlar elde edilebilir:

Şekil 4.5. Çaprazlanma sonrası oluşan ifadelerin ağaç gösterimleri

Böylece Şekil 4.4 ile ağaç gösterimleri verilen iki ağacın LISP gösterimleri

`(/ (* -1 b) (+ 3 (* a b)))`

ve

`(/ (- 2 (+ b 3)) (* 2 a))`

iken, çaprazlama işleminin uygulanması sonrasında LISP gösterimleri

`(/ (* -1 b) (* 2 a))`

ve

`(/ (- 2 (+ b 3)) (+ 3 (* a b)))`

olan *döl* (*offspring*) çözümlerin Şekil 4.5 ile gösterilen ağaçları elde edilir. Birinci döl çözümün orta-ek gösteriminin $\frac{-b}{2a}$ olduğu açıktır ve asıl amacın bir parabolün tepe noktasının formülünün bulunması olan bir uygulamada bu döl çözümün başarısı (*fitness*) diğer tüm aday çözümlerden fazla olacaktır.

Mutasyon operatörü ile bir aday çözümün herhangi bir elemanı aynı türden rastgele seçilmiş başka bir elemanla değiştirilir. Böylece arama uzayındaki farklı ifadelerin veya programların oluşturulması ve denenmesi sağlanmış olur. Mutasyon ile bir toplam operatörünün örneğin bir çarpım operatörüyle değiştirilmesi, matematiksel ifadede oldukça büyük bir etkiye sebep olabilir. Bununla birlikte ifadedeki bir sabitin 1 olan değerinin 2 ile değiştirilmesi daha küçük ancak anlamlı bir etkiye de sebep olabilir. Çaprazlama ve mutasyon operatörleri aday çözümlere ardı sıra uygulandığında hem etkin ifade parçaları gelecek nesillere taşınmış olur hem de yeni operatör ve sabit eklemeleriyle arama uzayının daha etkin bölgelerine sıçrama gerçekleşir.

Aşağıda Şekil 4.6 ile verilen ağaç gösterimlerinden soldakinin bir toplam operatörü çarpım operatörü ile değiştirilerek mutasyon işlemi uygulanmıştır:

Şekil 4.6. Tek bir operatörün mutasyonu öncesi ve sonrası oluşan ağaç gösterimleri

Başlangıçta Şekil 4.6'ya ilişkin LISP gösterimi,

`(* (+ 5 (+ 13 7)) (- 10 (* 5 2)))`

olan ifadenin değeri 0 iken, mutasyona uğratılmış ve

`(* (+ 5 (* 13 7)) (- 10 (* 5 2)))`

ile gösterilen ifadenin değeri de 0 değerini alır. Örnekte açıkça görüldüğü gibi mutasyon operatörü çok büyük ve küçük değişikliklere sebep olabileceği gibi operatörlerinin özellikleri de göz önüne alındığında hiçbir değişikliğe sebep olmayabilir.

Gösterimler ve operatörleri de bir arada ele alarak bir genetik programlama probleminin aşamalarını aşağıdaki şekilde gösterebiliriz:

1. N elemanlı bir topluluğu rastgele aday çözümlerle oluştur. Bu topluluk, geçerli topluluk adını alsın.
2. Tüm aday çözümler için uygunluk değerini hesapla.

3. Topluluktan uygunluk değerlerine göre iki iyi ebeveyn aday çözüm seç.
4. Belirli bir olasılıkla çaprazlama operatörünü uygulayarak iki yeni döl çözüm elde et. Çaprazlama gerçekleştiyse döl çözümlerle, gerçekleşmediyse seçilen ebeveynlerle 5. adıma geç.
5. Belirli bir olasılıkla mutasyon operatörünü 4. adımdaki çözümlere uygula.
6. Bir önceki adımdaki çözümleri yeni topluluğa ekle. Yeni topluluk büyüklüğü N' den küçükse adım 3'e dön.
7. Eski topluluğu sil, yeni topluluk geçerli topluluk olsun. Durma kriteri gerçekleştiyse işlemleri bitir. Aksi durumda adım 2'ye dön (Koza, 1992).

Yukarıda aşamaları verilen algoritmanın topluluk büyüklüğü, durma kriteri gibi unsurları problemden probleme değişiklik gösterir. Seçilen büyük N değerleri işlem sayısını ve dolayısıyla işlem sürelerini artırırken, arama uzayının daha fazla noktasının aranması gibi olumlu bir etkiye de sebep olur. Durma kriteri ise nesil sayısının belirli bir sabite ulaşması veya topluluğun en iyi üyesinin uygunluk değerinin belirli bir değere ulaşması olarak belirlenebilir.

Genetik programlamanın işleyişini basit bir örnek üzerinde göstermeye çalışalım. Elimizde ikili (binary) değerler alan X_1 , X_2 ve Y değişkenlerinin olduğunu varsayalım. Ayrıca bu değişkenler arasındaki ilişkinin de X_1 ve X_2 'den Y 'ye doğru olduğu konusunda sağlam dayanaklarımız olduğunu ancak ilişkinin fonksiyonel kalıbı hakkında fikrimiz olmadığını düşünelim.

Tablo 4.1.
XOR operatörüne ait doğruluk tablosu

X1	X2	Y
1	1	0
0	0	0
1	0	1
0	1	1

Söz konusu doğruluk tablosu XOR adındaki mantıksal operatöre aittir ve iki şekilde kural çıkartılabilir. 1 ve 0 değerlerinin mantıksal olduğu düşünülürse çıkartılan kural “ X_1 ve X_2 aynı değeri alırsa Y 'nin değeri 0 olur. X_1 ve X_2 farklı değerler alırsa Y 'nin değeri 1 olur.” cümlesiyle ifade edilebilir. Öte yandan değişkenlerin değerlerinin mantıksal değil de tam sayı olduğu düşünülürse çıkartılan kural

$$Y = |X_1 - X_2| \quad (4.4)$$

olur. Burada $|\cdot|$, mutlak değer fonksiyonunu gösterir. Sözle ifade edilen birinci kural aynı zamanda


```
if (X1 == X2){  
    Y = 0;  
}else{  
    Y = 1;  
}
```

gibi bir bilgisayar programına karşılık gelebilir. XOR mantıksal operatörünün işleyişini öğrenen bir genetik programlamada hangi operatörlerin kullanılacak olması da burada önem kazanır. Algoritmanın *if-then-else* ifadelerini kullanmasına izin verilirse sözle ifade edilen kural ortaya çıkartılabilir. Algoritmanın işleyişinde yalnızca +, -, *, / ve mutlak değer operatör ve fonksiyonlarının kullanılmasına izin verilirse ulaşılan en iyi kural $Y = |X_1 - X_2|$ olacaktır.

Aşağıdaki tabloda rastgele oluşturulmuş 100 elemanlı bir topluluğun ilk neslindeki en iyi ilk 10 aday çözümü gösterilmiştir:

```
1: -1.0: [4, 'Abs', 2, '/']  
2: -1.0: [4, 'Abs', 2, '/']  
3: -1.111111111111112: [2, 4, '/', 3, '*', 4, '/']  
4: -1.111111111111112: [3, 1, '/', 1, '-', 'Abs']  
5: -1.111111111111112: ['x2', 2, '-', 'x2', '+', 3, '*', 4, '/']  
6: -1.164444444444448: ['x2', 2, '-', 3, '+', 3, '*', 4, '/']  
7: -1.25: [1, 4, '/', 1, '*', 3, '/']  
8: -1.25: [4, 2, '*', 2, '/', 'Abs']  
9: -1.5: [1, 3, '-', 'x1', '/']  
10: -1.555555555555556: ['x1', 3, '-', 2, '/', 'Abs']
```

Birinci satırı ele alalım. [4, 'Abs', 2, '/'] ile ifade edilen topluluğun en iyi aday çözümüdür ve orta-ek gösterimi $\frac{2}{|4|}$ ’dür. Herhangi bir değişkene bağlı olmayan ve sabit olarak kabul edilen bu değer tüm Y değerlerini 1/2 olarak tahmin eder. Y değerleri (0, 0, 1, 1) ve $\hat{Y}$ değerleri (1/2, 1/2, 1/2, 1/2) iken hataların kareler toplamı 1 olur. Bu değer en küçük olması istendiği için uygunluk değeri de -1 olarak elde edilir. Söz konusu topluluğun ilk 5 elemanı 500 nesil sonrasında aşağıdaki gibi gerçekleşmiştir.

```
1: -0.0: ['x2', 'Abs', 'x1', '-', 'Abs']  
2: -0.0: ['x2', 'Abs', 'x1', '-', 'Abs']  
3: -0.0: ['x2', 'Abs', 'x1', '-', 'Abs']  
4: -0.0: ['x2', 'Abs', 'x1', '-', 'Abs']  
5: -0.0: ['x2', 'Abs', 'x1', '-', 'Abs']
```

Yeni topluluğun ilk elemanı orta-ek notasyonu ile $|X_1 - X_2|$ şeklinde gösterilebilir ve hata karelerin toplamı 0 şeklinde elde edilmiştir. Bu sonuç yeterlidir ve ilişki doğru olarak keşfedilmiştir ancak ikinci mutlak değer gereksizdir çünkü X_2 negatif değerler almaz. Genetik arama farklı başlangıç toplulukları ile tekrar denenirse,

```
1: -0.0: ['x2', 'x1', '-', 'Abs']
```

çözümü elde edilebilir ki, söz konusu aday çözümün ön-ek gösterimi $|X_1 - X_2|$ olur.

Genetik programlamada keşfedilen kuralın doğru ancak çok fazla gereksiz ifade içerdiği görülebilir. Bu bağlamda arama sırasında ifadelerin derinliklerinin veya uzunluklarının sınırlandırılması gerekebilir. İfadelere sınır getirme, problemin çözümü de bilinmediği için, gerçek kuralın keşfedilmesini engelleyebilir. Bu yüzden öncelikle sınır getirmeden bir arama yapmak ve gerekirse sonradan sadeleştirme yoluna gidilebilir.

Aşağıdaki genetik programlama çözümü XOR problemi için *if-then-else* kurallarının da kullanıldığı durumdaki çözümlerden biridir:

```
['x2', 'x1', 1, 'x2', '!=', 1, 1, 0, '==', 0, 'x1', 'x2', '==']
```

Bu çözümün gereksiz kurallar içerdiği görülebilir. Son-ek gösterimi bilgisayar programıyla ifade edilirse,

```
if(x2 == x1){
    y = 0;
}else{
    if(0 == 1){
        y = 1;
    }else{
        if(x2 != 1){
            y = x1;
        }else{
            y = x2;
        }
    }
}
```

kodu elde edilir. Sıfır asla bire eşit olamayacağından söz konusu kod şu şekilde sadeleştirilebilir:

```
if(x2 == x1){
    y = 0;
}else{
    if(x2 != 1){
        y = x1;
    }else{
        y = x2;
    }
}
```

Bu durumda oluşan kod söz konusu ilişkiyi daha sade betimler. Oluşan kurala göre girdi değişkenleri aynı ise çıktı değeri hep sıfır olur. Farklı oldukları durumda da çıktı değişkeni her zaman bir değerini alır. Burada genetik programlamayla aynı kural beklenmedik bir yorumla ortaya atılmış olur. Aynı problem için denemeler tekrarlanırsa aynı doğru sonucu veren, şekil olarak farklı ancak sonuç olarak aynı başka kurallar bulunabilir.

5. Genetik Programlama ve Ekonometri İlişkisi

Genetik programlamanın en yaygın uygulama alanı Koza (1992) tarafından önerilen sembolik regresyon adlı yöntemdir. Bu yöntem genetik programlamanın evrimsel sürecini uygulayarak mevcut veriye uyan matematiksel ifadeleri keşfetmeyi amaçlar (Giustolisi ve Savic, 2006: 208). Koza'ya (1992) göre sembolik regresyon ile değişkenler arasında ilişki kuran sembolik bir fonksiyonun tanımlanması için aranan matematiksel ifade bağımsız değişkenleri girdi (input) bağımlı değişkeni çıktı (output) olarak kabul eden bir bilgisayar programı olarak görülmelidir.

Genetik programlama aracılığı ile sembolik regresyon, non-parametrik ve doğrusal olmayan bir uygun model yapısı ve parametrelerini arama tekniğidir. Sembolik regresyonda geleneksel regresyon tekniklerinden farklı olarak modelin yapısının önsel olarak bilindiğine ilişkin bir varsayım bulunmamaktadır (Kotanchek, Vladislavleva ve Smits, 2009). Bu nedenle basitçe sadece model katsayılarının belirlenmesinden çok modelin yapısının ve katsayılarının birlikte belirlenmesi söz konusudur. Başka bir deyişle sembolik regresyon veriden modelin gösterimine ilişkin bir tümevarımdır. Bu özellikleri nedeni ile de doğrusal regresyon, polinomsal yaklaşımlar ve hatta yapay sinir ağları (artificial neural network) gibi tekniklerden keskin bir biçimde ayrışır (Affenzeller v.d., 2009:46-47).

Geleneksel regresyon modellerinde mevcut gözlem değerlerine uygunluk gösterecek fonksiyonel kalıbın önsel olarak bilindiği varsayılır. Uyumun yeterince iyi olmadığı durumlarda araştırmacının veriye uygun modeli elde edene dek farklı fonksiyonel kalıpları denemesi söz konusudur. Bu süreç sadece zahmetli olmakla kalmayıp aynı zamanda araştırmacının kişisel yetenek ve deneyimlerine fazlaca bağlı olmaktadır. Bu nedenle uzman araştırmacılar bile model seçimi esnasında yanılabilirler yaşayabilir. Örneğin bazı alanlarda söz konusu modelin doğrusal ya da eğrisel olacağına ilişkin önemli bir gelenek olsa dahi verinin daha iyi uyum sağlayacağı daha karmaşık bir kalıp bulmak mümkün olabilir. Sembolik regresyon fikrinin ardında yatan düşünce de tam olarak buradan ortaya çıkmaktadır. Özetle sembolik regresyon, mevcut veriye uygunluk sağlayan matematiksel modeli fonksiyonel kalıba ilişkin herhangi bir varsayım yapmadan bulmayı sağlayan yöntemdir (Poli, v.d., 2008: 114).

Belirli bir bağımsız değişkenler topluluğunun bir bağımlı değişkeni açıkladığının düşünüldüğü ekonometrik modeli ele alalım.

$$y = f(X; \beta) + \varepsilon \quad (5.1)$$

Burada y , $n \times 1$ mertebeli bağımlı değişken vektörünü; X , sütunlarında $n \times 1$ mertebeli $x_1, x_2, \dots, x_p$ vektörlerinin yer aldığı ve $n \times p$ mertebeli bağımsız değişken matrisini; β , $p \times 1$ mertebeli bilinmeyen parametrelerin bir vektörünü ve ε ise $n \times 1$ mertebeli bir hata vektörünü gösterebilir. Fonksiyonel kalıp doğrusal ve $p = 2$ olarak seçildiğinde söz konusu model

$$y = \beta_0 + \beta_1 x_1 + \varepsilon \quad (5.2)$$

olarak yazılabileceği gibi, fonksiyonel kalıp bir Sigmoid fonksiyonu ve $p = 3$ olarak seçildiğinde model

$$y = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}} + \varepsilon \quad (5.3)$$

halini alır. Fonksiyonel kalıp hakkında bir bilginin olmadığı durumda problem model seçimi başlığında incelenebilirken, modelin hangi değişkenleri içermesi gerektiği ise değişken seçme problemi olarak adlandırılır. Teorinin hem model hem de değişkenler hakkında yetersiz kaldığı durumda ise hem modelin hem de değişkenlerin belirlenmesi birlikte ele alınması gereken bir probleme dönüşmektedir.

Fonksiyonel kalıbın bilindiği ancak modelin hangi değişkenleri içermesi gerektiği konusunda bir fikir sahibi olunmadığı durumlarda problem kombinasyonel zorluklar içerir. Bağımsız değişken havuzunda p adet değişken yer aldığında bu değişkenlerin hangi $k \leq p$ tanesinin modelde yer alması gerektiği en kaba haliyle $\binom{p}{k}$ adet alt kümenin değerlendirilmesini gerektirebilir. Değişken seçimi problemi için en temel modele sırayla değişkenlerin eklenmesi (*forward selection*) veya en kapsamlı modelden sırayla değişkenlerin atılması (*backward elimination*) ve stepwise regresyon gibi yöntemler ile uygun model aranabilir (Ruengvirayudh ve Brooks, 2016). Uygun modelin ne olacağına bazı bilgi kriterleri (Akaike (1973, 1974) Akaike Bilgi Kriteri (Akaike Information Criterion - AIC), Akaike (1978) Bayes Bilgi Kriteri (Bayes Information Criterion-BIC), Schwarz (1978) Schwarz Bilgi Kriteri (Schwarz Information Criteria-SCI), v.b.) yardımıyla karar verilebilir. Değişkenlerin karşılıklı ilişkilerinin modeldeki etkileri sebebiyle hangi değişkenin modelden çıkarıldığı veya eklendiği sonucu değiştirebildiği için kapsamlı arama teknikleri de geliştirilmiştir. Regresyon modelindeki uygun bağımsız değişkenlerin veya zaman serisi modellerinde bağımlı değişkenin gerekli gecikmeli değerlerinin teşhisinde genetik algoritmaların sıkça kullanıldığını görüyoruz (Leardi, Boggia ve Terrile, 1992; Tolvi, 2004; Andersen ve Bro, 2010; Balcombe, 2005).

kullanılması fikri ilk kez Yule (1926) tarafından ortaya atılan otoregresif modeller (AutoRegressive - AR) olarak karşımıza çıkmaktadır. Örneğin bir hisse senedine ait getiri oranlarının, geçmiş bazı değerlerinin fonksiyonu olduğunu düşünebiliriz. Böyle bir ilişkinin keşfedilmesi söz konusu ise hisse senedinin gelecek değerlerinin tahmininde otoregresif süreçleri kullanabiliriz. Bu türden tahminlerin tek bir fonksiyonla tahmin edilemediği durumlar da çokça görülür (*Eşik otoregresif (threshold autoregressive - TAR)* modelleri (Tong, 1978), *kendinden uyarımlı eşik otoregresif (self-exciting threshold autoregressive – SETAR)* modeller (Tong, 1990) bu durumlara örnek gösterilebilir.)

Örneğin, bir zaman serisinin t anındaki değeri için

$$y_t = \begin{cases} f(y_{t \in S_1}) + \varepsilon_t, & k_1 \\ g(y_{t \in S_2}) + \varepsilon_t, & k_2 \\ h(y_{t \in S_3}) + \varepsilon_t, & k_3 \end{cases} \quad (5.4)$$

veri üretme süreci geçerli olsun. Burada S_1 , S_2 ve S_3 gecikme indislerinin kümeleri, $f(\cdot)$, $g(\cdot)$ ve $h(\cdot)$ bilinmeyen fonksiyonel kalıplar; k_1 , k_2 ve k_3 ise bazı ayrı koşulları gösterebilir. Bu genel gösterim sırasıyla k_1 , k_2 veya k_3 koşullarından biri gerçekleştiğinde bağımlı değişkenin t anındaki değerinin hangi fonksiyonla tahmin edileceği için genelleştirilmiştir. Burada fonksiyonel kalıpların, fonksiyonlara ait gecikme değerlerinin ve koşulların bilinmediğinin varsayıldığı önemlidir. Bununla birlikte

$$y_t = \begin{cases} \alpha_0 + \alpha_1 y_{t-1} + \varepsilon_t, & y_t < c \\ \varepsilon_t, & y_t = c \\ \beta_0 + \beta_3 y_{t-3} + \beta_7 y_{t-7} + \varepsilon_t, & y_t > c \end{cases} \quad (5.5)$$

modelinin $S_1 = \{1\}$, $S_2 = \emptyset$ ve $S_3 = \{3, 7\}$; $f(x) = \alpha_0 + \alpha_1 x$, $g(x) = 0$ ve $h(a, b) = \beta_0 + \beta_3 a + \beta_7 b$ ile $y_t < c$, $y_t = c$ ve $y_t > c$ kuralları için asıl model tarafından yuvalandığı söylenebilir. Doğrusal olmayan bir zaman serisi modeli gibi düşünülebilecek bu gösterimin hem fonksiyonel kalıplarının hem de diğer bilinmeyenlerinin tahmin edilmesi de bir genetik programlama uygulaması olarak karşımıza çıkar. Bu hali ile bu örnek Tong (1978) ve Tong ve Lim (1980) tarafından önerilen ve Tong (1990) tarafından ayrıntıları tartışılan eşik otoregresif modellerinin özel bir hali kendinden uyarımlı eşik otoregresif modellerin tahminin nasıl bir genetik programlama problemi olarak kabul edilebileceğini konusunda fikir vermektedir.

Geleneksel zaman serisi tahmin yöntemleri modelin doğrusal olması durumunda etkin sonuçlar vermekte ancak doğrusal olmayan durumlarda kesin sonuçlar konusunda başarısız olmaktadır. *Yapay sinir ağları ve destek vektör makineleri (support vector machines)*

gibi yöntemlerin kullanılması ile doğrusal olmayan zaman serilerinin gerçeğe yakın tahmin edilmesindeki başarı arttırılmaktadır. Ancak gerçek hayatta ilgilenilen zaman serisinin doğrusal ya da doğrusal olmayan bir kalıba uygun olduğunu kestirmek her zaman kolay değildir (Lee ve Tong, 2011: 72). Bu nedenle hibrit yöntemlerin tahmin başarısını arttırmak için geçerli bir yol olabileceği düşüncesinden yola çıkarak Lee ve Tong (2011) *otoregresif hareketli ortalamalar* (*AutoRegressive Integrated Moving Average – ARIMA*) modelleri ile genetik programlamayı bir araya getiren hibrit bir zaman serisi tahmin modeli önermişlerdir.

Son zamanlarda oldukça popüler olan ve yapay sinir ağının özel bir türü olan *ileri beslemeli yapay sinir ağları* (*artificial feed-forward neural networks*) ile genetik programlama arasındaki ilişki de ilginçtir. Genetik programlamaya benzer olarak yapay sinir ağlarında girdi ve çıktı değişkenleri arasındaki ilişki aranmaya çalışılır. Farklı olarak yapay sinir ağlarındaki çıktı değişkeninin, gizli katman sayısına ve her bir katmandaki nöron sayısına bağlı olarak girdi değişkenlerinin bir bileşke fonksiyonu olduğu düşünülür. Diğer bir deyişle, çıktı değişkeni bir önceki katmanın doğrusal olmayan bir bileşimiyken, önceki katmanlar da bir önceki katmanın doğrusal olmayan bileşimi şeklinde yazılır. Ancak genetik programlamadan farklı olarak aktivasyon fonksiyonları olarak bilinen bu fonksiyonların türleri (sigmoid, hiperbolik, tanjant, vb.) önceden belirlidir. Burada değinildiği haliyle yapay sinir ağları modelleri genetik programlama modellerinden, fonksiyonel kalıpların önceden bilinmesi özelliğiyle ayrılır.

Korelasyon, regresyon ve yapay sinir ağları temelli tahminçiler tamamen otomatik süreçlere bırakıldığında doğru olmayan sonuçlara ulaşılabilir. Arasında ilişki görülen iki değişkenin ilişkisi, ikisinin de ilişkili olduğu üçüncü bir değişken yüzünden anlamlı bulunmuş olabilir. Aynı tehlike genetik programlama ile keşfedilen ve değişkenler arasındaki ilişkiyi modelleyen ifadeler için de geçerlidir. Bu bağlamda genetik programlama ile keşfedilen bir ifade veya programın bir karar alıcı tarafından incelenmesi gerekebilir.

Genetik programlama, değişkenler arasındaki ilişkinin keşfedilmesinden öte değişkenler arası ilişkinin anlamlılığının testi için araçlar geliştirme konusunda da kullanılmıştır. Moore, Olson, Chen ve Sipper (2019) genetik programlama aracılığı ile *t* testine alternatif bir test istatistiğinin otomatik olarak keşfine ilişkin bir çalışma yapmış ve ortalamaların anlamlılığın testi için *t* testi kadar başarılı sonuç veren bir yaklaşım elde etmişlerdir. İyi bir test istatistiğinde bulunması gereken özellikleri düşük birinci tip hata, yüksek güç, ölçek değişmezliği ve mümkün olduğunca basit olması olarak tanımlayan araştırmacılar, Pareto Optimizasyonu çerçevesinde genetik programlama sistemlerinin değerlendirilmesini gerçekleştirmişlerdir. *Humies ödülleri* kapsamında 2019 yılının finalistleri arasında

olan bu çalışmanın sonuçları istatistik disiplini alanında çözümlenmemiş test istatistiği problemlerinde genetik programlamanın bir çözüm sunabileceği yönünde yenilikçi bir bakış açısı sunmaktadır.

Genetik programlamanın yaygın kullanım alanlarından biri de teknik analizdir. Finansal piyasalarda “*al-tut*” (*buy and hold*) stratejisi oldukça bilinen bir uygulamadır. Basitçe firma ve faaliyet gösterdiği sektör umut verici görünüyorsa yatırımcılar o firmanın hisse senetlerini alma ve uzun dönem ellerinde tutma eğilimdedirler. Bilinen bir diğer yaklaşım ise daha çok kısa dönemli dalgalanmalara odaklanan dinamik bir yaklaşım olarak “*pazar zamanlaması*” (*market timing*) yaklaşımıdır. Bu yaklaşımda teknik analiz yolu ile kısa zaman dönemi içinde uygun alım satım sinyalleri yaratan kuralların bulunması amaçlanır. Genetik programlamanın bu al-sat sinyallerini otomatik olarak veriden yakalama konusundaki başarısı teknik analiz konusunda kullanımını da yaygınlaştırmıştır (Potvin, Soriano ve Vallée, 2004). Örneğin; toplam getiriyi maksimize eden “ c , k ve l sabit eşik değerleri olmak üzere, eğer x_{t-1} belirli bir aralıktaysa ayrıca $x_{t-3} > c$ ise o halde tut, aksi durumda $x_{t-2} < k$ ve $x_{t-4} > l$ ise sat” gibi hem bilinmeyenlerinin hem de gecikmeli değerlerinin önceden bilinmediği bir kuralı otomatik olarak keşfetmek genetik programlama için finans alanında sıradan bir uygulama sayılır. Potvin, Soriano ve Vallée (2004) menkul kıymetler piyasasında genetik programlama yolu ile kural arama çalışmasında genetik programlama temelli alım satım kararlarının piyasanın düşüşte ya da istikrarlı olduğu dönemlerde kazançlı ancak piyasanın yükselişte olduğu dönemlerde başarısız olduğu sonucuna varmışlardır. Menkul kıymetler piyasalarında genetik programlama ile alım satım kuralları aramaya ilişkin diğer çalışmalara Mallick, Lee ve Ong (2008), Esfahanipour ve Mousavi (2011) örnek gösterilebilir. Ayrıca Manahov ve Zhang (2019) çalışmasında yüksek frekanslı işlem verilerini kullanarak finansal piyasaları tahminlemede genetik programlamanın kullanımını incelemiştir.

6. Sonuç

Bilgisayar biliminin odaklandığı temel noktalardan biri de bilgisayarların bir problemi çözmek için yapılması gerekenin nasıl yapılacağının söylenmeden yapmasını sağlayabilme çabasıdır. Yapay zekâ ve makine öğrenmesi gibi kavramların da ardında yatan temel motivasyon buna benzemektedir. Bu aşamada genetik programlama biyoloji bilimdeki evrim prensiplerini temel alarak otomatik bilgisayar programları keşfetmeyi amaçlayan bir teknoloji sunmaktadır. Genetik programlama her ne kadar bilgisayar bilimi başlığı altında anılan bir kavram olsa da mühendislikten ekonomiye çok geniş kullanım alanları bulmakta ve uygulama alanlarında insanla rekabet edebilir, yenilikçi sonuçlar veren çözümler sunmaktadır.

Bu çalışmada genetik programlama düşüncesinin tarihsel süreç içerisinde gelişimi makine zekâsı, yapay zekâ ve makine öğrenmesi kavramları eşliğinde tartışılmıştır. Kapsamlı bir literatür taraması ve kavramın tanımlanmasından sonra genetik programlamanın mühendislik, elektronik, görüntü işleme, işaret işleme, işletme, ekonomi, finansal zaman serileri, tıp, biyoloji, eğlence ve bilgisayar oyunları gibi alanlardaki uygulamaları tanıtılmıştır.

Son birkaç on yılda genetik programlamanın, uygulama alanlarında insanla rekabet edebilir nitelikte pek çok yenilikçi ve patentlenebilir çözümler ürettiği izlenmektedir. Çalışmanın ilgili bölümünde söz konusu yenilikçi çalışmaların yayımlandığı, tartışıldığı ve genetik programlama alanındaki uygulamalı ve teorik gelişmelerin takip edilebileceği konferanslar, oluşumlar ve ödüller tanıtılmıştır. Bu oluşumlar arasında belki de en popülerleri ilk kez 2004 yılında Genetic and Evolutionary Computation Conference (GECCO) isimli konferansta düzenlenen ve *Humies* adı verilen ödüllü bir yarışmanın ardından günümüze dek her yıl verilen “*Human Competitive Awards*” ödülleridir.

Genetik programlamanın yaygın kullanım alanlarından biri de sembolik regresyondur. Sembolik regresyon parametrik olmayan bir uygun model ve model katsayıları arama yöntemidir. Genetik programlamanın ekonometri alanındaki uygulamalarının kesişiminde yer alan sembolik regresyon, genetik programlamanın evrimsel süreci yardımı ile veriye uygun matematiksel gösterimi ve model parametreleri bir arada keşfetmeyi sağlayan doğrusal olmayan bir yöntemdir. Çalışmada genetik programlamanın çalışma prensipleri, temel gösterimler ve genetik programlamanın genetik algoritmalar ile benzerlikleri ve farklılıklarının tanıtıldığı teknik açıklamalar sonrasında sembolik regresyon tanıtılmıştır.

Sembolik regresyon geleneksel regresyon yöntemlerinden farklı olarak modelin fonksiyonel biçimi hakkında önsel olarak herhangi bir bilginin varlığını gerektirmez. Bunun yanı sıra modelde yer alan değişkenleri ve modelin fonksiyonel kalıbını aynı anda tahmin etmesi özelliği ile de geleneksel regresyondan ayrılır. Aynı zamanda modelde hangi değişkenlerin yer alması gerektiği konusunda teorik olarak bilginin yetersiz olduğu durumlarda da genetik programlama önemli avantajlar sağlamaktadır. Bu bağlamda genetik programlamanın ekonometride model seçimi ve değişken seçimi konularında faydalı bir araç olarak kullanılabileceğine ilişkin örnekler sunulmuştur. Ekonometri - genetik programlama ilişkisi hakkındaki kavrayış genetik programlamanın doğrusal olmayan finansal zaman serilerinin modellenmesi ve tahmininde bir alternatif olarak kullanılabileceğine ilişkin tasarlanan örnekler ile zenginleştirilmiştir. Model parametrelerinin tahminin yanı sıra değişkenlerin anlamlılığının testi için de bir araç olarak kullanılması konusundaki çalışmalara da yer verilmiştir. Aynı zamanda genetik programlamanın örüntü tanımlama konusundaki

yetenekleri nedeni ile finansal verilerde kural arama konusunda nasıl başarılı sonuçlar verebileceği ifade edilmiştir.

Sonuç olarak, genetik programlamanın değişkenler arasındaki ilişkiler ve modelin yapısı hakkında bilginizin yetersiz olduğu ya da bilinen ilişkilerin yanlış olduğu konusunda şüphe duyulduğu durumlarda kullanılabilecek alternatif bir araç olabileceği ifade edilebilir. Model seçimi, değişken seçimi, veride örüntü arama, yeni anlamlılık testlerinin keşfi konularında yenilikçi çözümler üretme konusunda başarılı bir yöntemdir. Problemlerin çözümü için insan ile rekabet edebilir özellikle çözümler üretse de bazı dezavantajları da bulunmaktadır. Çözüm olarak yeterli sayının üzerinde model gösterimi sunmakta ve neredeyse hemen her zaman aşırı uyumlu (over-fitted) çözümler üretmektedir. Bu dezavantajları göz önünde bulundurarak araştırmacılar eldeki probleme yorumlanabilir çözümler seçme konusunda dikkatli olmaları gerektiğini söylemek yanlış olmaz. Öte yandan genetik programlamanın, XOR mantık operatörü örneğinde olduğu gibi, değişkenler arasındaki ilişkileri başka fonksiyonel kalıplarla üstelik mükemmel bir şekilde açıklayan yeni formülleri keşfedebilmesi, teorisinin sunduğu açıklamalar yanında meselenin başka açılardan yorumlanması yönünde de yeni kapılar açmaktadır.

Bu çalışmanın birtakım kısıtlılıkları da mevcuttur. Genetik programlamanın yalnızca ekonometri alanında değil istatistik, yöneylem araştırmaları konularında da pek çok problem için başarılı çözümler ürettiği uygulamalar mevcuttur. Bu çalışmada ilişkiler ekonometri uygulamaları konusunda sınırlı tutulmuştur. Aynı zamanda sembolik regresyonun genetik programlama yerine diğer evrimsel algoritmalar kullanılarak uygulandığı hibrit yöntemler de literatürde yer almaktadır. Bu uygulamalara da temel düzeyde değinilerek çalışmanın kapsamı dışında tutulmuştur. İlerleyen çalışmalarda genetik programlamanın istatistik ve yöneylem araştırmaları alanındaki çözümlerini tartışan araştırmalar yapılabilir.

Kaynakça/References

- Adalı, E. (2017). Yapay Zekâ, *İstanbul Teknik Üniversitesi Yayını: İnsanlaşan Makineler ve Yapay Zekâ*, 75, 8–13.
- Affenzeller, M., Winkler, S., Wagner, S. & Beham A. (2009). *Genetic Algorithms and Genetic Programming Modern Concepts and Practical Applications*, Taylor & Francis Group, LLC.
- Akaike, H. (1973). Information Theory and an Extension of the Maximum Likelihood Principle. In B.N. Petro and F. Csaki ed. 2nd International Symposium on Information Theory, pp. 267–281.
- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Trans. Autom. Control*, 19(6), 716–723.
- Akaike, H. (1978). A Bayesian analysis of the Minimum AIC Procedure. *Annals of the Institute of Statistical Mathematics*, 20, 9–14.
- Andersen, C. M., & Bro, R. (2010). Variable selection in regression—a tutorial. *Journal of Chemometrics*, 24, 728–737.

- Arganis, M., Val, R., Prats, J., Rodríguez, K., Domínguez, R. & Dolz, J. (2009). Genetic Programming and Standardization in Water Temperature Modelling. *Hindawi Publishing Corporation Advances in Civil Engineering*, Volume 2009, Article ID 353960, 10 pages.
- Arnaldo, I., Krawiec, K., & O'Reilly, U. (2014). Multiple regression genetic programming. *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ACM, pp. 879–886.
- Aryafar, A., Khosravi, V., Zarepourfard, H. & Rooki, R. (2019). Evolving genetic programming and other AI-based models for estimating groundwater quality parameters of the Khezri plain, Eastern Iran. *Environmental Earth Sciences*, 78, 69.
- Azaria, Y., & Sipper, M. (2005a). GP-gammon: Genetically programming backgammon players. *Genetic Programming and Evolvable Machines*, 6(3), 283–300.
- Balcombe, K. G. (2005). Model selection using information criteria and genetic algorithms. *Computational Economics*, 25(3), 207–228.
- Banzhaf, W., Nordin, P., Keller, R. E. & 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.
- Bauer, R. J., Jr. (1994). *Genetic Algorithms and Investment Strategies*. John Wiley.
- Bhanu, B. & Lee, S. (1994). *Genetic Learning for Adaptive Image Segmentation*. Boston: Kluwer Academic Publishers.
- Chadalawada J, Havlicek V., & Babovic, V. (2017). A genetic programming approach to system identification of rainfall-runoff models. *Water Resources Management*, 31, 3975–3992.
- Chen, S. H., Duffy, J., & Yeh, C. H. (2002). Equilibrium selection via adaptation: Using genetic programming to model learning in a coordination game. *The Electronic Journal of Evolutionary Modeling and Economic Dynamics*, 15 January 2002.
- Corno, F., Sanchez, E., & Squillero, G. (2005). Evolving assembly programs: how games help microprocessor validation. *IEEE Transactions on Evolutionary Computation*, 9(6), 695–706.
- Cramer, N. L. (1985). A Representation for the Adaptive Generation of Simple Sequential Programs, *Proceedings of An International Conference On Genetic Algorithms and Their Applications*, pp. 183–187.
- Davidson, J. W., Savic, D. A., & Walters, G.A. (2003). Symbolic and numerical regression: experiments and applications. *Information Sciences*, 150, 95–117.
- Dijkstra, E. W. (1961). Algol 60 translation: An algol 60 translator for the x1 and making a translator for algol 60. *Stichting Mathematisch Centrum. Rekenafdeling. Stichting Mathematisch Centrum*.
- Dorsey, R., & 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.
- Drecourt, J. P. (1999). Application of neural networks and genetic programming to rainfall run off modeling. *Water Resources Management*, 13, 219–231.
- Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. (2004). Least angle regression. *The Annals of statistics*, 32(2), 407–499.
- Esfahanipour, A., & Mousavi, S. (2011). A genetic programming model to generate risk-adjusted technical trading rules in stock markets. *Expert Systems with Applications*, 38(7), 8438–8445.
- Fischer, M. M., & Leung, Y. (1998). A genetic-algorithms based evolutionary computational neural network formodelling spatial interaction data. *The Annals of Regional Science*, 32(3), 437–458.
- Forsyth, B. R. (1981). A Darwinian approach to pattern recognition. *Kybernetes*, 10, 159–166.
- Freitas, A. A. A. (2008). Review of evolutionary Algorithms for Data Mining, In: Maimon, O., Rokach, L. (Editors.), *Soft Computing for Knowledge Discovery and Data Mining*. Springer US, 79–111.
- Gen, M. & Cheng, R. (1997). *Genetic Algorithms and Engineering Design*. New York: John Wiley and Sons.

- Giustolisi, O., & Savic, D. A. (2006). A symbolic data-driven technique based on evolutionary polynomial regression. *Journal of Hydroinformatics*, 8(3), 207–222.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison-Wesley.
- Hauptman, A., & Sipper, M. (2005). GP-endchess: Using genetic programming to evolve chess endgame players. In M. Keijzer, et al., editors, *Proceedings of the 8th European Conference on Genetic Programming*, volume 3447 of Lecture Notes in Computer Science, pages 120–131, Lausanne, Switzerland, 30 March - 1 April 2005. Springer.
- Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI.
- İnce, G. (2017). İnsanlığın Yapay Zekâyı İmtihani, *İstanbul Teknik Üniversitesi Yayını: İnsanlaşan Makineler ve Yapay Zekâ*, 75, 14–17.
- Jin, N. (2005). Equilibrium selection by co-evolution for bargaining problems under incomplete information about time preferences. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 2661–2668, Edinburgh, 2–5 September 2005.
- Jin, N., & Tsang, E. P. K. (2006). Co-adaptive strategies for sequential bargaining problems with discount factors and outside options. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 7913–7920, Vancouver, 6–21 July 2006.
- 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. (2005). Extended daily exchange rates forecasts using wavelet temporal resolutions. *New Mathematics and Natural Computation*, 1(1), 79–107.
- Kotanchek M. E., Vladislavleva, E. Y., & Smits, G.F. (2009) Symbolic regression via GP as a discovery engine: Insights on outliers and prototypes. In: Riolo RL, O'Reilly UM, McConaghy T (eds) *Genetic Programming Theory and Practice VII, Genetic and Evolutionary Computation*, Springer, Ann Arbor, pp.55–72.
- Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press.
- Koza, J. R. (1997). Genetic Programming. *Encyclopedia of Computer Science and Technology to be edited by Allen Kent and James G. Williams*, 1–26.
- Koza, J. R., Bennett, III F. H., & Stiffelman, O. (1999) Genetic programming as a Darwinian invention machine. In R. Poli, et al., editors, *Genetic Programming, Proceedings of EuroGP '99*, volume 1598 of LNCS, pages 93–108, Goteborg, Sweden, 26–27 May 1999. Springer-Verlag.
- Koza, J. R., Keane, M. A., Yu, J., Bennet III, F. H., & Mydlowec, W. (2000). Automatic Creation of Human-Competitive Programs and Controllers by Means of Genetic Programming. *Genetic Programming and Evolvable Machines*, 1, 121–164.
- Koza, R., Al-Sakran, S.H., & Jones, L. W. (2005). Automated re-invention of six patented optical lens systems using genetic programming. In H.-G. Beyer, et al., editors, *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 2, pages 1953–1960, Washington DC, USA, 25–29 June 2005. ACM Press.
- Koza, J. R. (2010). Human-competitive results produced by genetic programming. *Genet Program Evolvable Mach*, 11, 251–284.
- Langdon, W. B. (1998). *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!*, Springer.
- Langdon, W. B., & Poli, R. (2002). *Foundations of genetic programming*. Springer-Verlag.

- Leardi, R., Boggia, R., & Terrile, M. (1992). Genetic algorithms as a strategy for feature selection. *Journal of chemometrics*, 6(5), 267–281.
- Lee, Y. S., & Tong, L. I. (2011). Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming. *Knowledge-Based Systems*, 24(1), 66–72.
- Lo, A. W. (2004). The Adaptive Markets Hypothesis: Market Efficiency from an Evolutionary Perspective. *Journal of Portfolio Management*, Forthcoming. Available at SSRN: <https://ssrn.com/abstract=602222>
- Lohn, J. D., Hornby, G. S., & Linden, D. S. (2004). Evolutionary antenna design for a NASA spacecraft. In U.-M. O'Reilly, et al., editors, *Genetic Programming Theory and Practice II*, chapter 18, pages 301–315. Springer, Ann Arbor, 13-15 May 2004.
- Mallick, D., Lee, V. C. S., & Ong, Y. S. (2008). An empirical study of Genetic Programming generated trading rules in computerized stock trading service system. *2008 International Conference on Service Systems and Service Management*, Melbourne, VIC, Australia, IEEE.
- Man, K. F., Tang, K. S., Kwong, S., & Halang, W. A. (1997). *Genetic algorithms for control and signal processing*. Springer, London.
- Manahov, V., & Zhang, H. (2019). Forecasting Financial Markets Using High-Frequency Trading Data: Examination with Strongly Typed Genetic Programming. *International Journal of Electronic Commerce*, 23(1), 12–32.
- Marney, J. P., Miller, D., Fyfe, C., & Tarbert, H. F. E. (2001). Risk adjusted returns to technical trading rules: a genetic programming approach. In *7th International Conference of Society of Computational Economics*, Yale, 28-29 June 2001.
- Massey, P., Clark, J. A., & Stepney, S. (2005). Evolution of a human-competitive quantum fourier transform algorithm using genetic programming. In H.-G. Beyer, et al., editors, *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 2, pages 1657–1663, Washington DC, USA, 25-29 June 2005. ACM Press.
- Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer-Verlag.
- Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. Cambridge, MA: The MIT Press.
- Moore, J., Olson, R., Chen, Y., & Sipper, M. (2019). Automated discovery of test statistics using genetic programming. *16th Annual Humies Awards - 2019 - Prague*, Czech Republic. URL <http://www.human-competitive.org/awards>
- Mousavi, S., Esfahanipour A., & Zarandi, M. H. F. (2014). A novel approach to dynamic portfolio trading system using multitree genetic programming. *Knowledge-Based Systems*, 66, 68–81.
- Neely, J., Weller, P. A., & Dittmar, R. (1997). Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *The Journal of Financial and Quantitative Analysis*, 32(4), 405–426.
- Neely, C. J., & 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., & Weller, P. A. (2001). Technical analysis and central bank intervention. *Journal of International Money and Finance*, 20(7), 949–970.
- Neely, C. J. (2003). Risk-adjusted, ex ante, optimal technical trading rules in equity markets. *International Review of Economics and Finance*, 12(1), 69–87.
- Neely, C. J., Weller, P. A., & Ulrich, J. M. (2006). The adaptive markets hypothesis: evidence from the foreign exchange market. *Working Paper 2006-046B, Federal Reserve Bank of St. Louis, Research Division*, P.O. Box 442, St. Louis, MO 63166, USA.
- Orlov, M., & Sipper, M. (2009). Genetic programming in the wild: Evolving unrestricted bytecode. *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ACM.
- Pal, S. K., & Wang, P. P. (1996). *Genetic Algorithms and Pattern Recognition*. Boca Raton, FL: CRC Press.

- Perkis, T. (1994). Stack-based genetic programming. *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*, IEEE.
- Poli, R., Langdon, W. B., & 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> (With contributions by J. R. Koza).
- Potvin, J.Y., Soriano, P. & Vallée, M. (2004). Generating trading rules on the stock markets with genetic programming. *Computers & Operations Research*, 31(7), 1033–1047.
- Ruengvirayudh, P., & Brooks, G.P. (2016). Comparing Stepwise Regression Models to the Best-Subsets Models, or, the Art of Stepwise. *General Linear Model Journal*, 42(1), 1–14.
- Schwarz, G. (1978). Estimating the Dimensions of a Model, *The Annals of Statistics*, 6, 461–464.
- Shichel, Y., Ziserman, E. & Sipper, M. (2005). GP-robocode: Using genetic programming to evolve robocode players. In M. Keijzer, et al., editors, *Proceedings of the 8th European Conference on Genetic Programming*, volume 3447 of Lecture Notes in Computer Science, pages 143–154, Lausanne, Switzerland, 30 March - 1 April 2005. Springer.
- Smith, S. F. (1980). *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, University of Pittsburgh.
- Stender, J., Hillebrand, E., & Kingdon, J. (editors). (1994). *Genetic Algorithms in Optimization, Simulation, and Modeling*. Amsterdam: IOS Publishing.
- Tanimoto, S. L. (1987). *The elements of artificial intelligence: an introduction using LISP*. New York, NY, USA: Computer Science Press.
- Tolvi, J. (2004). Genetic algorithms for outlier detection and variable selection in linear regression models. *Soft Computing*, 8, 527–533.
- Tong, H. (1978). On A Threshold Model. Chen, C. (Ed.). *Pattern Recognition And Signal Processing*, p. 575–586, NATO ASI Series E: Applied Sc. (29), Sijthoff ve Noordhoff, Netherlands.
- Tong, H., & Lim, K.S. (1980). Threshold Autoregression, Limit Cycles and Cyclical Data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 42(3), 245–292.
- Tong, H. (1990). *Non-Linear Time Series: A Dynamical System Approach*, Oxford University Pres.
- Trujillo, L., & Olague, G. (2006). Using evolution to learn how to perform interest point detection. In X. Y. T. et al., editor, *ICPR 2006 18th International Conference on Pattern Recognition*, volume 1, pages 211–214. IEEE, 20–24 August 2006a.
- Turing, A. M. (1936). Oncomputable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, Series 2, 42 (1936-7), 230–265.
- Turing, A. M. (1948). Intelligent machinery. [Reprinted in ed. by D.C. Ince, *Mechanical Intelligence: Collected Works of A. M. Turing*. (North Holland, Amsterdam, 1992), pp. 107–127]. Also reprinted in ed. by B. Meltzer, D. Michie. 1969]. *Machine Intelligence 5*. (Edinburgh University Press, Edinburgh, 1948).
- Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236), 433–460. [Reprinted in D. C. Ince (ed), *Mechanical Intelligence: Collected Works of A. M. Turing*. (North Holland, Amsterdam, 1992), pp. 133–160].
- Wong, M. L., & Leung, K.S. (2002). *Data Mining Using Grammar Based Genetic Programming and Applications*. Kluwer Academic Publishers New York, Boston, Dordrecht, London, Moscow.
- Wu, A. S., & Banzhaf, W. (1998). Introduction to the special issue: Variable-length representation and noncoding segments for evolutionary algorithms. *Evolutionary Computation*, 6(4), iii–vi.
- Yule, G. U. (1926). Why do we sometimes get nonsense correlations between time series? A study in sampling and the nature of time series. *J. Roy. Statist. Soc.*, 89, 1–63.