MÜHENDİSLİKTE TERSİNE PROBLEM UYGULAMALARI İÇİN GENETİK PROGRAMLAMA YAKLAŞIMI

Yenal ARSLAN

Zonguldak Karaelmas Üniversitesi
Fen Bilimleri Enstitüsü
Makine Eğitimi Anabilim Dalında
Bilim Uzmanlığı Tezi
Olarak Hazırlanmıştır

KARABÜK
Haziran 2005

KABUL:

Yenal ARSLAN tarafından hazırlanan "MÜHENDİSLİKTE TERSİNE PROBLEM UYGULAMALARI İÇİN GENETİK PROGRAMLAMA YAKLAŞIMI" başlıklı bu çalışma jürimiz tarafından değerlendirilerek, Makine Eğitimi Anabilim Dalında Bilim Uzmanlığı Tezi olarak oybirliğiyle kabul edilmiştir. 20/06/2005

Başkan: Prof. Dr. Hüseyin EKİZ (SAÜ) .....

Üye : Yrd. Doç. Dr. A. Turan ÖZCERİT (SAÜ) .....

Üye : Yrd. Doç. Dr. Cevdet GÖLOĞLU (ZKÜ) .....

ONAY:

Yukarıdaki imzaların, adı geçen öğretim üyelerine ait olduğunu onaylarım. 12/7/2005

Prof. Dr. Ihsan TOROĞLU
Fen Bilimleri Enstitüsü Müdürü

ÖZET

Bilim Uzmanlığı Tezi

MÜHENDİSLİKTE TERSİNE PROBLEM UYGULAMALARI İÇİN GENETİK PROGRAMLAMA YAKLAŞIMI

Yenal ARSLAN

Zonguldak Karaelmas Üniversitesi
Fen Bilimleri Enstitüsü
Makina Eğitimi Anabilim Dalı

Tez Danışmanı: Yrd. Doç. Dr. Cevdet GÖLOĞLU
Haziran 2005, 59 sayfa

Mühendislik problemlerinden doğası gereği yararlanıcılar için oldukça fazla çözüm elde edilme imkanı vardır. Bulunan çözümler arasında birden fazla kabul edilebilen veya optimum olarak değerlendirilen çözümler bulunmaktadır. Mühendislikte ileri çözümün yanında tersine çözümün istendiği durumlarla da karşılaşılmaktadır. Tersine problem kabaca cevabın bilindiği fakat sorunun bilinmediği problemler olarak tanımlanabilir.

Bu çeşit problemler için Evrimsel Algoritmaların bir alt dalı olan Genetik Programlama (GP) uygun çözümler sağlamaktadır. GP yaklaşımı, daha önceden tanımlanmış algoritmalar yerine evirilerek ilerlemekte ve sonucu ortaya çıkarmaktadır. Yapılan bu çalışmada, GP yaklaşımı kullanılarak, bir yazılım geliştirilmiş ve mühendislikte tersine problem uygulamaları için çözümler üretilmiştir. Elde edilen sonuçlar, GP yaklaşımının kullanıldığı yazılımın başarılı sonuçlar ürettiğini göstermiştir.

ÖZET (Devam Ediyor)

Geliştirlen yazılım mühendislik alanında cevabın bilindiği fakat sorunun bilinmediği problemlere kolaylıkla uygulama şansına sahiptir.

Anahtar Sözcükler: Genetik Programlama, Tersine problem, İşleme.
Bilim Kodu: 628.08.01

ABSTRACT

M.Sc. Thesis

GENETIC PROGRAMMING APPROACH FOR INVERSE PROBLEM IMPLEMANTATIONS IN ENGINEERING

Yenal ARSLAN

Zonguldak Karaelmas University
Graduate School of Natural and Applied Sciences
Department of Mechanical Education

Thesis Advisor: Asst. Prof. Cevdet GÖLOĞLU
June 2005, 59 pages

The vast number of possible solutions can be obtained for the beneficiaries from engineering problems due to their natures. Among the solutions discovered more than ones can be accepted or can be evaluated as optimum. In engineering domain, despite, generally, the need of forward problem solution cases, it can be faced with the cases that inverse solutions are required. Inverse problem can be stated as the problems that the solution is known but the answer.

Genetic Programming (GP) which is a branch of Evolutionary Algorithms can provide solutions for this kind of problems. The GP approach uses a method to evolve computer programs for drawing the conclusion instead of using pre-defined algorithms. In this study conducted, a computer program with GP approach has been developed and the solutions

ABSTRACT (Continued)

have been generated for inverse problems in engineering. The results obtained show the computer program employing GP approach generates successful solutions.

The software developed gives a chance to be easily implemented for the problems that the answers are known but the problems are not known.

Keywords: Genetic programming, Inverse problem, Machining.
Science Code: 628.08.01

TEŞEKKÜR

Çalışmalarımda bana destek veren danışmanım sayın Yrd. Doç. Dr Cevdet GÖLOĞLU ile sayın Arş. Gör. Oğuzhan DOĞAN'a ve TEDAŞ Bilgi İşlem Merkezinden sayın Müd. Yrd. Koray ÇETİN'e sonsuz saygı ve teşekkürlerimi sunarım.

Bu günlerde gelmemde maddi manevi desteklerini benden esirgemeyen aileme çok teşekkür ederim.

İÇİNDEKİLER

İÇİNDEKİLER (devam ediyor)

Sayfa

İÇİNDEKİLER (devam ediyor)

Sayfa

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

SİMGELER

EVi	:	i. bireyin seçilme olasılığı
Fi	:	i. bireyin uyumluluğu
H	:	Seçilme olasılığı en yüksek olan bireyin rulet tekerleğindeki taradığı alan
L	:	Seçilme olasılığı en düşük olan bireyin rulet tekerleğindeki taradığı alan
N	:	Popülasyonun boyutu
P_c	:	Çaprazlama olma olasılığı
P_m	:	Mutasyon olma olasılığı
P_r	:	Yeniden üretim olma olasılığı
T	:	Sıcaklık
$\overline{F}$	:	Popülasyondaki ortalama uyumluluk
σ	:	Uyumluluk varyansı

KISALTMALAR

artuyum	:	Artan uyumluluk
ayaruyum	:	Ayarlanmış uyumluluk
bulunan	:	Test verisinin hesaplanan uyumluluğu
GB	:	Giga bayt
GHz	:	Giga hertz
noruyum	:	Normal uyumluluk
tveri	:	Test verisi

BÖLÜM 1

GİRİŞ

Çözüm uzayı oldukça büyük ve karmaşık olan ve geleneksel optimizasyon yöntemleri ile çözülemeyen problemleri çözmek için çeşitli yöntemler geliştirilmiştir. Bunlardan biri de Evrimsel Algoritmalardır. Evrimsel Algoritmalarda doğada görülen biyolojik değişimin yöntemleri kullanılır. Evrimsel Algoritmalar geleneksel çözüm arama yöntemlerinden farklıdırlar. Geleneksel çözüm arama yöntemleri, probleme bir çözüm adayı önerir ve onu degiştirerek daha iyi çözümler elde etmeye çalışır. Aksine Evrimsel Algoritmalar, bir çözüm adayları popülasyonu oluşturur ve bu popülasyon zamanla evrimleşir. Bir adayın, çözüme ne kadar yakın olduğu, bir uyumluluk fonksiyonu ile belirlenir. Hepsinde de, algoritma her adayın ne kadar güçlü olduğunu hesaplar ve buna göre bir sonraki neslin ebevynleri olacak ya da yok olacak bireyleri belirler. Daha sonra, makul bir yeni nesil oluşturmak için ebeveynlere genetik arama işlemcilerini (yeniden üretim, çaprazlama ve mutasyon) uygular. Bu döngü her defasında daha güçlü bireyler oluşturarak tekrarlanır.

Evrimsel Algoritmalar içerisinde problem çözümüne yönelik çeşitli yaklaşımlar mevcuttur. Bu yaklaşımların birbirinden temel farkları problemin ve problem hakkındaki bilginin gösterim şeklidir (Ege, 2004). Buna göre Evrimsel Algoritmalar: Genetik Algoritmalar, Evrimsel Programlama, Evrim Stratejileri ve Genetik Programlama (Boun, 2001) diye dört guruba ayrılabilir.

Yapılan çalışmada Evrimsel Algoritmaların bir alt kolu olan Genetik Programlama kullanılarak çözüm uzayı oldukça büyük olan problem gruplarından tersine problem çözümü yapılmıştır. Geliştirilen Genetik Programlama modelleyicisi ile seçilen örnek problemler üzerinde çözümler gerçekleştirilmiştir.

BÖLÜM 2

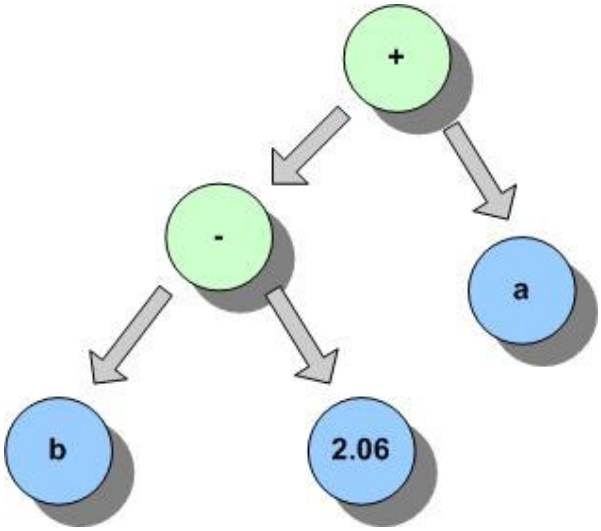
GENETİK PROGRAMLAMA

2.1 GENETİK PROGRAMLAMANIN BİLEŞENLERİ

Genetik Programlama, ele alınan problemin yapı taşlarından oluşturulan muhtemel ilkel çözüm tarzlarının belli bir uyum kriterine göre evrilerek mükemmelleşmesini amaçlayan bir Evrimsel Algoritma tekniğidir (Koza, 1992; Koza, 1994).

2.1.1 Terminal Ve Fonksiyonlar

Teknik, ele alınan problemi çözmeye yönelik bir nesil ilkel yapının rastgele oluşturulması ile başlar. Bu yapılara kromozom (birey) da denir. Bireyleri oluşturabilmek için iki farklı sınıftan gelen nesneler kullanılır. Bu nesneler terminaller ve fonksiyonlardır. Terminaller problemin kurulmasına yarayan ve problemin tanımlanmasında doğrudan etkin olan değiskenler veya sabitlerdir. Örneğin; bir karıncanın besin bulma probleminde, karıncanın yaşamını devam ettirmesi için bulması gereken besinin, problemin tanımlı olduğu evrendeki konumu veya karıncanın bu evrendeki konumu besin bulma probleminde birer terminaldirler. Her problemim kendine has terminal yapısı vardır ve terminallerin seçimi konusunda kesin kurallar yoktur. Problemi Genetik Programlamaya uyarlayan kişiler bunları belirlerler (Ege, 2004). Fonksiyonlar ise terminalleri belli bir işlem mantığına göre birleştirmeye yarayan nesnelerdir. Fonksiyonlar ve terminallerin hiyerarşik bir anlayış içinde anlamlı bir yapı oluşturacak şekilde birleştirilmesi sonucunda kromozomlar oluşur. Genetik Programlamada kullanılan kromozomların yapısı şekil olarak bir kaynaktan çıkan ve daha sonra dallanıp budaklanan bir ağaca (tree) benzer. Terminaller ağaçsal yapının dallanıp budaklanarak sonlandığı noktalara yerleştirilir. Fonksiyonlar ise bu uç noktalardan ağacın kaynağına kadar olan diğer bütün kısımlarda yer alırlar. Şekil 2.1'de matematiksel Terminal ve Fonksiyonlardan oluşturulmuş bir kromozomun ağaçsal yapısı gösterilmektedir



Şekil 2.1 Bir bireyin ağaçsal yapısı (+ ve – Fonksiyonları; a, b, 2.06 Terminalleri).

2.1.2 Uyumluluk (Fitness) Fonksiyonu

Nesildeki bireyler ele alınan problemin tipine ve tanımına göre matematiksel ya da işlevsel sonuçlar verirler. Bu sonuçlar, tanımlanmış bir prosedüre göre uyum diye adlandırılan, bireyin problemi çözebilme yeteneğini gösteren bir tür sayısal niceliğe çevrilirler ve bu niceliğe göre sıralanırlar. Sıralananan bireylerin bir kısmı, yine çoğu kez uyumlarına dayanan bir prosedüre (Genetik İşlemcilere) göre seçilirler. Bu temel genetik işlemciler; Yeniden Üreme, Çaprazlama ve Mutasyon olarak özetlenebilir.

2.1.3 Genetik İşlemciler

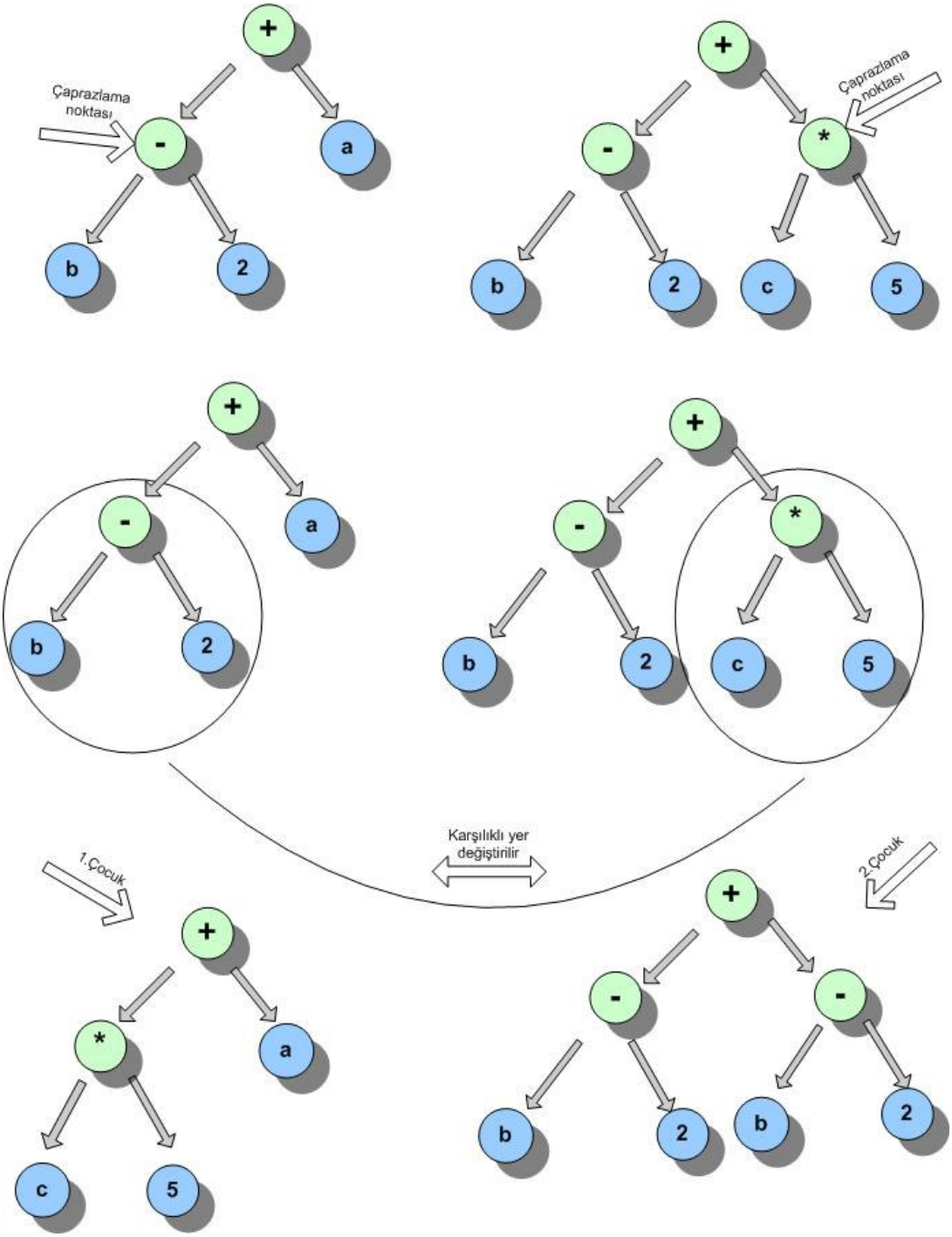
2.1.3.1 Yeniden Üreme (Reproduction)

Seçilen bireylerden bazıları gelecek nesile doğrudan aktarılır (yeniden üreme). Bir kısmı da gelecek nesile bazı farklılaşma işlemlerinden geçirilerek yeni bir birey olarak girerler.

2.1.3.2 Çaprazlama (Crossover)

Çaprazlama bu farklılaşmayı sağlayan işlemlerden biridir. Seçilmiş iki birey onları oluşturan ağaçsal yapıların herhangi bir noktasından ikiye parçalanır. Ayrılan parçalar

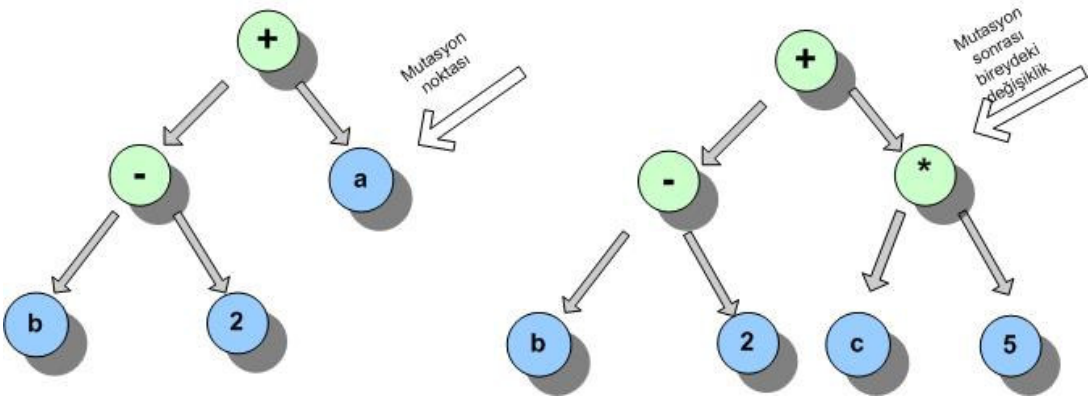
çapraz olarak yer değiştirirler. Şekil 2.2'de çaprazlama öncesinde varolan iki birey ve bu çaprazlama sonrası oluşan iki yeni birey örnek olarak verilmiştir.



Şekil 2.2 Çaprazlama için seçilmiş iki birey ve bireylerin seçilmiş çaprazlama noktaları ile çaprazlama sonrası oluşan yeni bireyler (çocuklar).

2.1.3.3 Mutasyon (Mutation)

Mutasyon seçilen bir bireyin ağaç yapısının herhangi bir noktadan kırılması ve kırılan noktanın altında kalan bölgeye yeni bir ağaçsal yapı eklenmesi ile yeni bir birey elde edilmesi işlemidir. Şekil 2.3'de mutasyon için seçilen bir birey ve bireyden mutasyon sonucu elde edilen yeni birey gösterilmektedir.



Şekil 2.3 Mutasyon için seçilmiş birey ve bireyin seçilmiş mutasyon noktası ve mutasyon sonrası oluşan yeni birey.

Çaprazlama yeniden üreme ve mutasyon işlemlerin belli bir oranda uygulanması sonucu eski nesilden yeni bir nesil elde edilir. Bu yeni nesil de yukarıda anlatılan işlevlerden geçirilerek başka nesillerin oluşmasını sağlayacaktır. Ne zaman herhangi bir nesilde elde ettiğimiz bireylerden bir tanesi istediğimiz sonuca ulaşırsa Genetik Programlama yöntemi o zaman sonlandırılır. Yöntem bu en iyi sonuç veren bireyi problemin çözümü olarak sunar.

2.1.4 Genetik Programlamanın Aşamaları

Genetik Programlamanın aşamaları şu şekilde sıralanabilir;

Birey havuzununun (popülasyonunun) rastgelelikten de yararlanarak oluşturulması.

Bir durma koşulu sağlanıncaya kadar aşağıdaki eylem dizisini yinelemesi:

a) Havuzda oluşan bireylerin herbirisinin kodladığı problem çözümünün kalitesinin değerlendirilmesi.

b) Bu değerlendirmenin ışığında evrimsel işlem yapıcıları (işlemciler) uygulayarak havuzu değişikliğe uğratması.

Seçim (selection) : Değişikliğe uğramış bir havuzdan bir sonraki kuşağa hangi bireylerin aktarılacağının seçimidir.

Yeniden Üretim : Seçilen bireylerden hangilerinin diğer hiçbir genetik operatör tarafından değişikliğe uğramadan (çaprazlama, mutasyon) yeni nesile aktarılmasının seçimine denir.

Çaprazlama : Havuzdaki bireylerden bazılarının eşleştirilmesi ve bu ata olarak adlandırılan bireylerin gen bilgilerini içeren, çocuk olarak adlandırılan, yeni kromozomların üretilmesidir. Böylece her çocuğun her bir geni atalarından birisi ile aynı olacaktır.

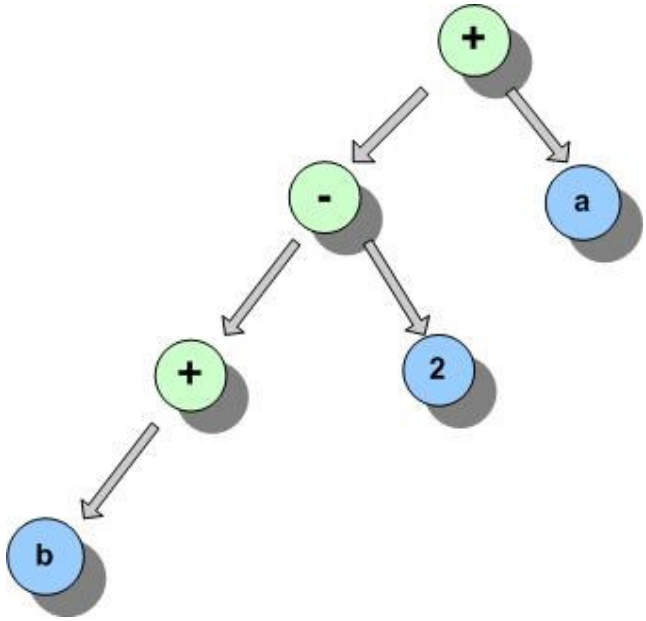
Mutasyon : Doğadakine pek benzer olarak, havuz bireylerinin arasından rasgele seçilenlerinin bazı gen değerlerinin rasgele değiştirilmesidir.

2.1.4.1 Birey Havuzunun Oluşturulması

Genetik programlarda başlangıç popülasyonları problem çözümüne ulaşmada çok önemli bir etkendir. Bu nedenle probleme uygun "başlangıç popülasyonu oluşturma yöntemi" seçilmelidir. Bu yöntem deneme yanılma yöntemi ile aşağıda listelenmiş bazı başlangıç popülasyonu oluşturma yöntemleri arasından seçilebilir.

Büyüme Yöntemi (Grow Method)

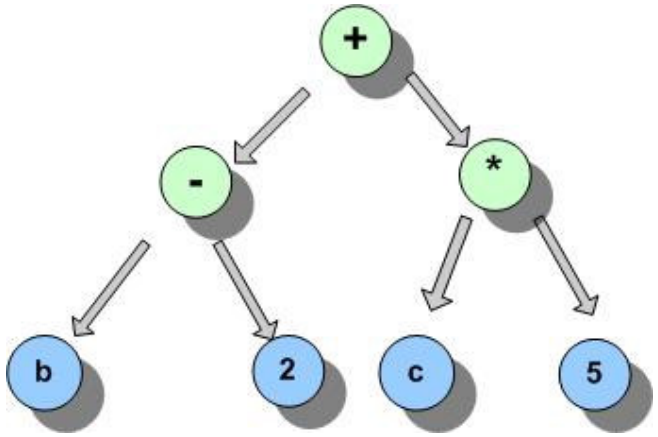
Bu yöntem ile oluşan bireyler m derinligine sahip ağaçlardır. Algoritması; ağacın kökünden (root) başlayarak, yapraklar (node) rastgele seçilen fonksiyon yada terminallerle doldurulur. Burada bir kökün yapraklarının aynı derinlikte olması gerekmez farklı farklı derinliklerde olabilirler. Şekil 2.4'de Büyüme yöntemi ile oluşturulmuş ağaç yapısı gösterilmektedir.



Şekil 2.4 Büyüme yöntemi ile oluşturulmuş ağaç yapısı.

Doyurma Yöntemi (Full Method)

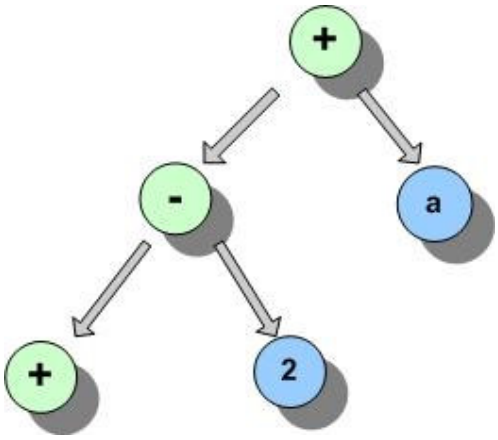
Bu yöntem Büyüme yöntemine çok benzer fakat burada kökün bütün yapraklarının aynı uzunlukta olması ve ağacın maksimum derinliğine (m) eşit olması gerekir. Şekil 2.5'de Doyurma yöntemi ile oluşturulmuş ağaç yapısı gösterilmektedir.



Şekil 2.5 Doyurma yöntemi ile oluşturulmuş ağaç yapısı.

Yarı-Yarı Dağıtma Yöntemi (Ramped Half-And-Half Method)

Bu yöntem Koza'nın (1992) ortaya çıkarttığı yukarıda bahsedilen iki popülasyon oluşturma tekniğini de kapsayan bir yöntemdir. Buna göre kökün bir yaprağı büyüme yöntemi ile oluşturulurken bir yaprağı doyurma yöntemi ile oluşturulur. Şekil 2.6'da Yarı-yarı Dağıtma yöntemiyle oluşturulmuş ağaç yapısı gösterilmektedir.



Şekil 2.6 Yarı-yarı Dağıtma yöntemiyle oluşturulmuş ağaç yapısı.

2.1.4.2 Uyumluluk (Fitness) Fonksiyonunun Belirlenmesi

Uyumluluk fonksiyonu popülasyondaki bireylerin, problem gözönüne alındığında ortama olan uyumlarını ölçmekte kullanılan bir fonksiyondur. Uyumluluk fonksiyonu da tıpkı terminal ya da fonksiyonlar gibi probleme göre ve Genetik Programlama modelleyicisine göre değişiklik gösterecektir ve uyumluluk fonksiyonunun seçiminde kesin kurallar yoktur.

Uyumluluk fonksiyonu, popülasyondaki her bir bireyi parametre olarak alan ve bu bireylerin ortama uyumluluklarını karşılaştırabilmemizi sağlayacak şekilde uyumluluk değerlerini oluşturup geri döndüren ve gerçekleştirimi problemden probleme ve Genetik Programlama modelleyicisinden diğer Genetik Programlama modelleyicisine değişen bir fonksiyondur.

2.1.4.3 Uyumluluk Fonksiyonunun Çeşitleri

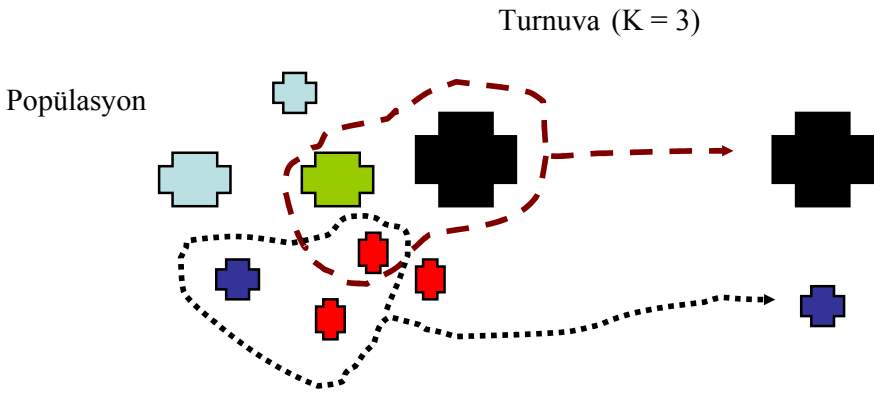
Hata Tabanlı (Error-based): Uyumluluk fonksiyonunun hata bazlı olması toplam hataya göre ters orantılı olmasıdır.
Maliyet Tabanlı (Cost-based): Uyumluluk fonksiyonunun kullanılan kaynakla ters orantılı olması (zaman, yer, para, kullanılan materyal, ağacın yapraklarının sayısı) dır.
Fayda Tabanlı (Benefit-based): Uyumluluk fonksiyonunun alınan faydayla doğru orantılı olmasıdır.
Cimrilik Tabanlı (Parsimony-base): Uyumluluk fonksiyonun bireylerin sadeliği ile doğru orantılı olmasıdır.
Enropy Tabanlı (Entropy base): Uyumluluk fonksiyonunun bir bireyler topluğunun istatistiksel doğruluğuna dogru ya da ters orantılı olmasıdır.

2.1.4.4 Seçim (Selection)

İşlemlerin uygulanacağı bireylerin seçiminde kullanılan yöntemi ifade eder. Problem tipine göre seçim yöntemlerinden biri seçilir. Herbirinin kendine has avantaj ve dezavantajları vardır (Hancock, 1994). Aşağıda seçim yöntemlerinden birkaçı sıralanmıştır (Maza and Tidor, 1991).

Turnuva Seçimi (Tournament Selection)

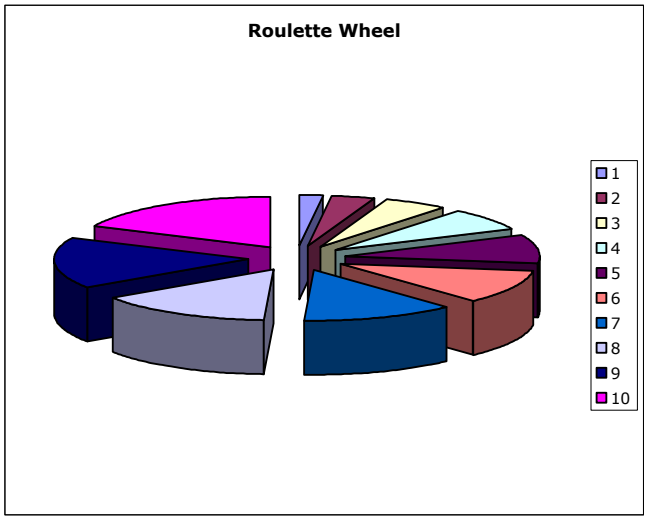
Öncelikle popülasyon içerisinde rastgele K adet (3, 5, 7) birey seçilir. Turnuva seçiminde K'nın seçimi genellikle popülasyonun büyüklüğüne göre değişir örneğin 5000 büyüklüğünde bir popülasyonunuz varsa o zaman K'yı daha yüksek alınmalıdır. K=7 yada daha fazla olmalıdır. Bu bireylerin içerisinde uyumluluğu en iyi olan birey seçilir. Bu işlem seçilen bireylerin sayısı popülasyon sayısına eşit olana kadar devam eder. Şekil 2.7'de turnuva seçimi yöntemi açıklanmaktadır.



Şekil 2.7 Turnuva seçiminin gösterimi (Maza and Tidor, 1991).

Uyum-Oranlı Seçimi (Fitness-Proportionate Selection)

Roulet tekerleği (Roulette wheel) seçimi olarakta bilinir. Uyumluluğu en yüksek olanın seçilme şansı daha yüksektir. Fakat uyum oranlı seçimin dezavantajı, her seçimde en iyiyi seçme olasılığı yüksek olduğundan bölgesel maksimumlara takılma olasılığının çok yüksek olmasıdır. Dolayısı ile genetik program muhtemel çözümü eğer en iyi birey ile en kötü bireyin çaprazlamasında bulacaksa uyum oranlı seçimde bu çok zor olacaktır. Uyum oranlı seçimin çalışmasını açıklayacak olursak; örneğin rulet tekerleği döndüğünde skalada en fazla yer kaplayan bireyin seçilme olasılığının daha yüksek olduğunu görürüz (Şekil 2.8).



Şekil 2.8 Uyum oranlı seçim (Maza and Tidor, 1991).

$$EV_i = F_i / \overline{F} \qquad (2.1)$$

Burada;

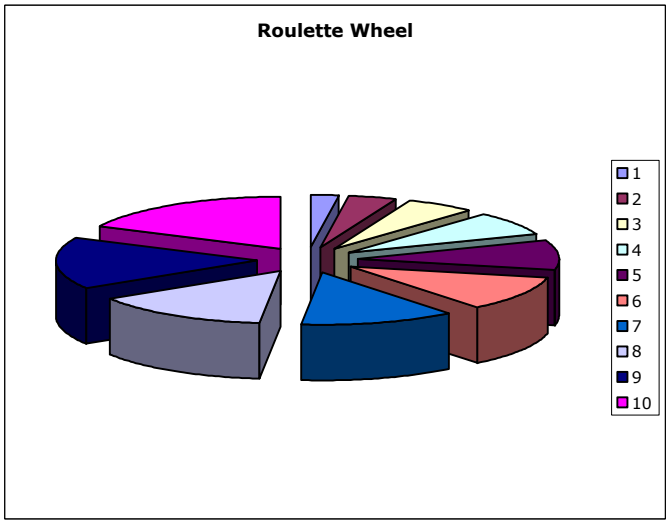
EVi = i. seçilme olasılığı,

Fi = i. bireyin uyumluluğu,

$\overline{F}$ = Popülasyondaki ortalama uyumlulukdur.

Sigma Skalalı Seçimi (Sigma Scaling Selection)

Bu tür seçimde bir bireyin seçilme olasılığı ne çok düşük olabilir ne de çok yüksek olabilir. Burada varyans bize bütün bireylerin seçilme olasılığının makul bir sınır içerisinde olmasını sağlar (Şekil 2.9).



Şekil 2.9 Sigma scaling seçimi (Maza and Tidor, 1991).

$$EV_i = 1 + \frac{F_i - \overline{F}}{2\sigma} \qquad (2.2)$$

Burada;

EVi = i. bireyin seçilme olasılığı,

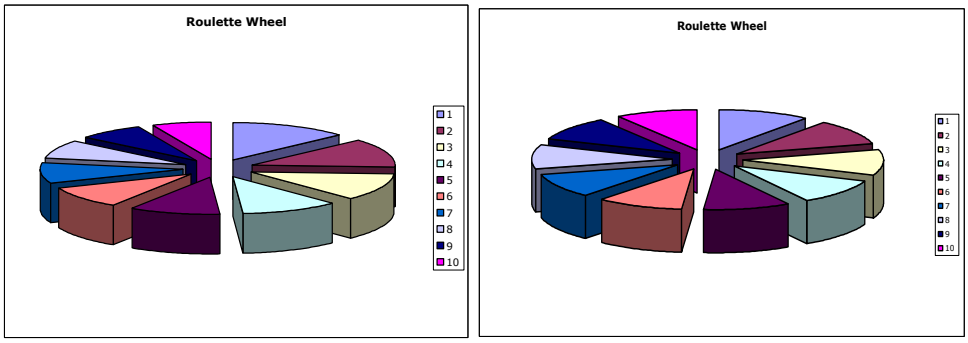
Fi = i. bireyin uyumluluğu,

$\overline{F}$ = Popülasyondaki ortalama uyumluluk,

σ = Uyumluluk varyansıdır.

Sıralama Seçimi (Ranking Selection)

Sıralama (rank) seçiminde, uyumluluk değerleri ayarlandıktan sonra popülasyondaki bireyler ranklarına göre sıralanırlar (Şekil 2.10). Bu ranklar 1 den (ki bu bireyin en iyi olduğuna işaret eder) popülasyonun büyüklüğüne (ki bu da en kötü bireydir) kadar değerler alırlar. Seçimin sonunda yeni popülasyonda bir önceki popülasyonun en iyi bireyleri ile en kötü bireyleri eşit oranda seçilmiş olur. Sıralama seçimi bize bireylerin uyumluluklarına göre gerçek dağılımını verir (Baker 1985; Whitley 1989). Dezavantajı ise popülasyonda diğerlerinden farklı olarak uyumluluğu çok yüksek olan bireyi kaçırma olasılığıdır. Bir de eğer bir popülasyondaki bireylerin uyumlulukları birbirlerinden çok az farklarla ayrılıyorlar ise sıralama seçiminin içlerinden biraz daha iyi olanı bulabilmesi için birçok jenerasyonun geçmesi gerekmektedir. Sonuç olarak sıralama seçimi en iyiyi çok fazla baskın yapmayarak bölgesel maksimumlara takılmamızı engeller (Koza 1992).



Şekil 2.10 Sıralama seçimi (Maza and Tidor, 1991).

$$EV_i = \frac{H - \frac{(i-1)(H-L)}{N-1}}{\frac{L+H}{2}} \tag{2.4}$$

Burada;

H = Seçilme olasılığı en yüksek olan bireyin rulet tekerleğindeki taradığı alan,

L = Seçilme olasılığı en düşük olan bireyin rulet tekerleğindeki taradığı alan,

EVi = i. bireyin seçilme olasılığı,

N = Popülasyonun boyutudur.

Bu seçim yöntemlerine ek olarak Genetik Programlamada bazen elitizm de kullanılır Elitizm popülasyondaki en iyi K bireyin doğrudan yeni jenerasyona (çarprazlama ve mutasyon olmadan) aktarılmasıdır. Genetik Programlamada genellikle K=1 olarak alınır.

2.1.4.4 Çaprazlama

Havuzdaki bireylerden bazılarının eşleştirilmesi ve bu ata olarak adlandırılan bireylerin gen bilgilerini içeren, çocuk olarak adlandırılan, yeni kromozomların üretilmesidir. Böylece her çocuğun her bir geni atalarından birisi ile aynı olacaktır. Çaprazlamaya aynı zamanda sexual reproduction da denir. Çaprazlamada havuzdaki bireylerden rastgele iki birey seçilir. Bu bireylerden birincisine, birinci ata ikincisine, ikinci ata denir. Bu bireylerden yine rastgele bir çaprazlama noktası işaretlenir. Bu noktaya çaprazlama noktası (crossover point) denir. Bu çaprazlama noktası kökün dışında heryerde olabilir. Sonrasında birinci atanın çaprazlama noktasının altında kalan alt ağaç ile ikinci atanın çaprazlama noktasının alt ağacı karşılıklı yer değiştirilir. Oluşan yeni ağaçlara birinci çocuk ve ikinci çocuk denir. Genellikle çaprazlama oranı 0.5-0.9 arasında alınır.

2.1.4.5 Mutasyon

Mutasyonda havuzdaki bireylerden rastgele bir tanesi seçilir. Bu seçilen bireyden rastgele bir nokta belirlenir. Bu belirlenen noktaya mutasyon noktası (mutation point) denir. Seçilen noktanın alt ağacındaki bütün terminaller silinerek yerlerine aynı terminal setinden rastgele seçilen yeni terminaller girilir. Aynı şekilde alt ağaçta bulunan bütün fonksiyonlar silinerek aynı fonksiyon setinden bulunan rastgele seçilen yeni fonksiyonlar girilir. Mutasyon bizi Genetik Programlamada zaman zaman rastladığımız local minimum yada local maksimumlardan kurtulmamızı sağlar. Genellikle mutasyon oranı 0.01-0.05 arasında alınır.

2.1.5 Genetik Programın Parametrelerinin Belirlenmesi

Genetik parametrelerin belirlenmesi problemin türüne göre değişiklik arzeder. Hangi probleme hangi değişkenlerin daha uygun olduğunu ya daha önce yapılmış doğruluğu ispatlanmış deney sonuçlarından ya da deneme yanılma yöntemini kullanarak bulunur

Popülasyonun boyutu (Population Size): Genetik programda her bir nesilde oluşturulacak bireylerin sayısını ifade eder.

Çaprazlama olasılığı (Crossover Probability): Bir sonraki nesilde yaşamını sürdürecek bireylerin belirlenmesi işleminde çaprazlama operatörünün kullanılma olasılığını ifade eder. Genellikle 0.5 ila 0.9 arasında değişen oranlarda seçilir.

Mutasyon olasılığı (Mutation Probability): Bir sonraki nesilde yaşamını sürdürecek bireylerin belirlenmesi işleminde mutasyon operatörünün kullanılması olasılığını ifade eder. Genellikle 0.01 ile 0.1 arasında alınır.

Birey seçim yöntemi (Individual Selection Method): Bir sonraki nesilde yaşamını sürdürecek bireylerin belirlenmesi işleminde çaprazlama ve mutasyon gibi işlemlerin uygulanacağı bireylerin seçiminde kullanılan yöntemi ifade eder. Bu yöntemler Turnuva, Uyum-oranlı, Sigma skalalı ve Sıralama seçimleri olarak alt başlık halinde 2.1.4.3'de açıklanmıştır.

Sonlandırma kriteri (Termination Criterion): Genetik programın sonlandırılması gerekip gerekmediğini belirleyen kriterdir. Örneğin, Genetik Programlama modelleyicisi belli bir süre sonunda genetik programın sonlandırılması gerektiğini düşünüyorsa programı bu süre kriterine göre tasarlayabilir. Maksimum nesil sayısı da (aşağıda açıklanacak) bir sonlandırma kriteri olabilir. Programın herhangi bir bireyin aranılan uygunluk aralığına ulaştığını tespit etmesi de bir sonlandırma kriteri olabilir.

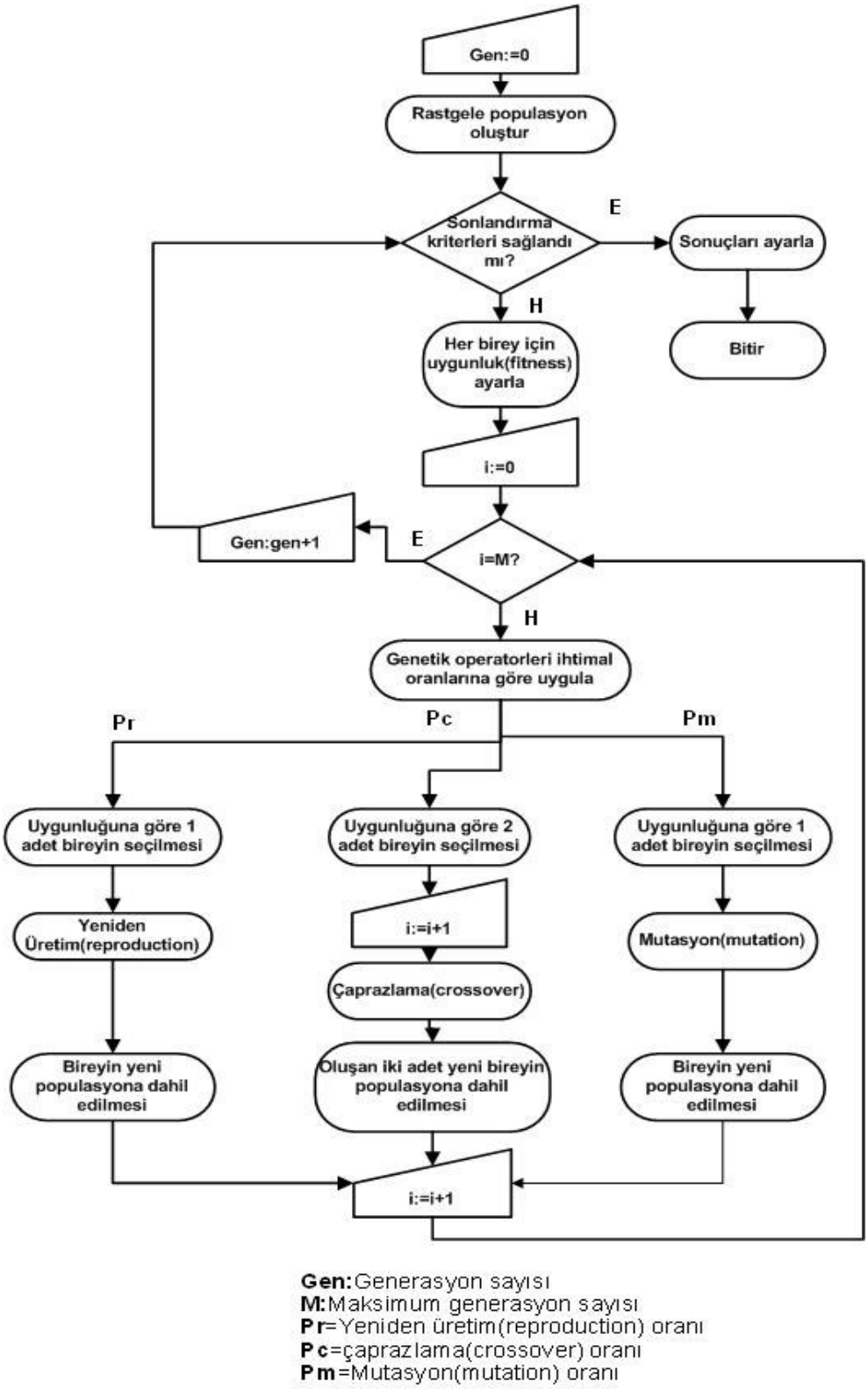
Maksimum Nesil Sayısı (Maximum Number of Generations) : Genetik Programda üretilecek maksimum nesil sayısını ifade etmektedir. Program maksimum nesil sayısına ulaştığında sonlandırılır.

Çaprazlama sonrasında oluşacak program ağaçlarının maksimum derinliği (Maximum Depth of Tree After Crossover) : Çaprazlama sonrasında çok büyük programların oluşması engellenmek istenirse, bu parametre ile oluşacak programların boyutları sınırlandırılabilir.

Mutantların (Mutasyon sonrasının) maksimum derinliği (Maximum Mutant Depth) : Mutasyon sonrasında çok büyük programların oluşması engellenmek istenirse, bu parametre ile oluşacak programların boyutları sınırlandırılabilir.

Başlangıç popülasyonu oluşturma yöntemi (Population Initialization Method) : Genetik programlarda başlangıç popülasyonları problem çözümüne ulaşmada çok önemli bir etkendir. Bu nedenle probleme uygun "başlangıç popülasyonu oluşturma yöntemi" seçilmelidir.

Şekil 2.11'de Genetik Programlamanın akış şeması görülmektedir;



Şekil 2.11 Genetik Programlamanın akış şeması.

2.2 TERSİNE PROBLEM

Tersine problem kabaca cevabın bilindiği fakat sorunun bilinmediği problemler olarak tanımlanabilir. Diğer bir deyişle sonuçlar veya neticeler bilinir ama nedenler bilinmez. Bu konuyu bilinen bir örnekle açıklama yapacak olursak; üç akarsuyun birleşerek bir nehir oluşturduğunu düşünelim. Diğer bir bilgi ise üç ayrı fabrikanın kirli atıklarının akarsulara bırakıldığı olsun. Bu mantıkla, kolayca fabrikaların nehirde ne kadar kirliliğe sebebiyet verdiklerini hesaplamak mümkündür. Bu ileri problem çözümü veya geleneksel çözüm olarak adlandırılabilir. Fakat, diğer (tersine) problem şeklinde ise; bilinen nehirdeki kirlilik miktarıdır. Bu noktada problemin çözümü hangi fabrikanın akarsuya kirli atık bıraktığını bulmaktır. Bu problem tersine problem olarak adlandırılır.

Bu çeşit problemler için Genetik Programlama uygun çözümler sağlamaktadır. Genetik Programlama ile oluşturulan yazılım, daha önceden tanımlanmış algoritmalar yerine evrilerek ilerler ve sonucu ortaya çıkarır. Problemin çözümünde her zaman yüzde yüz başarı sağlanamasa bile yüzde doksan ve üzeri olarak elde edilen sonuçlar tatminkar olarak kabul edilmektedir. Genetik Programlama tersine problemlerin çözümünde fonksiyonların tahmin edilmesinde iyi bir yöntem olarak karşımıza çıkmaktadır (Platel et al., 2005).

Trivailo et al. (2004a ve 2004b) yaptıkları çalışmayı yapısal uçak parametrelerinin belirlenmesinin tersine problemi üzerinde gerçekleştirmişlerdir. Yapay Sinir Ağları ve Genetik Algoritmalar ile sistem elemanları parametrelerinin (dayanıklılık ve kütle) tahminini sağlamışlardır.

Tasarım optimizasyonu için yakınsama modeli oluşturmada Genetik Programlama yöntembilimi kullanılmıştır (Toropov and Alvarez, 1998). Yapılan çalışmada iki farklı alanda problem uygulamaları gerçekleştirilmiştir. Çelik yapısında bulunan hasarın yakınsanması yapılmış ve çimentodan derin dökümü yapılmış kolonların kesme gerilmelerinin tahmini gerçekleştirilmiştir. Çalışma sonundaki çıktılar sayesinde yüksek kaliteli yakınsamaların gerçekleştirildiği görülmüştür.

Xu and Liu (2002) yaptıkları çalışmada, valfsiz çalışan bir mikro pompalı yanma odasındaki basınç kayıplarını optimize etmek için Genetik Programlama kullanmışlardır. Problemin çözümü tersine bir mantıkla gerçekleştirilmiştir.

Nastran and Balic (2002) çalışmalarında metal tel davranışını Genetik Programlama kullanarak tahmin etmeye çalışmışlardır. Şekillendirildikten sonra tel geometrisinde olan etkinliğini bir kağıt klasöründe bulunan tel şeklinin üzerinde çalışmışlardır. Telin kavis boyutlarının yük altında ölçülen parametrelere göre elde edilen deney verileri dikkate alınarak tahmini bir davranış formülü elde etmişlerdir.

Milfelner et al. (2005) radyus uçlu kesici takım kullanılarak yapılan bir imalat sırasında takımda oluşan kesme kuvvetlerini ölçerek, optimum kesme parametrelerinin oluşan kesme kuvvetlerine olan etkisini Genetik Programlama kullanarak bulmaya çalışmıştır.

Çözüm uzayı oldukça büyük ve karmaşık olan, geleneksel optimizasyon yöntemleri ile çözülemeyen problemleri çözme için kullanılan Evrimsel Hesaplama (Evolutionary Computation) ve Yapay Sinir Ağları (Artificial Neural Network) optimizasyon yaklaşımları oldukça geniş bir popülariteye sahip olmuştur. Her tekniğin avantajları ve dezavantajları vardır (Kapishnikov, 2002).

Yapay sinir ağları kısa boyutlu seri tahminlerinde oldukça başarılı olarak kullanılabilir. İlgili teknik, etkilenilen ve doğrudan bağımlı olan değişkenler arasındaki bağın doğrusal olmadığı (nonlinear) ve çok gürültülü olduğu durumlarda uygulama etkilidir (Giles et al., 2001). Buna örnek borsadaki verilerdir. İlgili tekniğin temel eksikliği ise, bize tekniğin altındaki işlemin anlaşılması için herhangi bir şeyin verilmemesi adeta kapalı bir kutu olmasıdır. Bunun yanında bize sonuç olarak bir problemle ilgili kurallar sağlayamamaktadır. Kullanıcılar bu yüzden yapay sinir ağlarının çalışmasını tercihte biraz sınırlı kalmaktadırlar.

Diğer yanda Genetik Programlama uygulamasında yukarıda bahsedilen sorunlar ortaya çıkmamaktadır. Daha da arzu edilen, kullanıcıya çözüm olarak karmaşık kurallar yerine basit kurallar (çözüm formülleri, fonksiyonlar) sunabilmektedir. Bunun yanında bu teknik oldukça fazla hesaplama zamanına ihtiyaç göstermektedir (Kapishnikov, 2002).

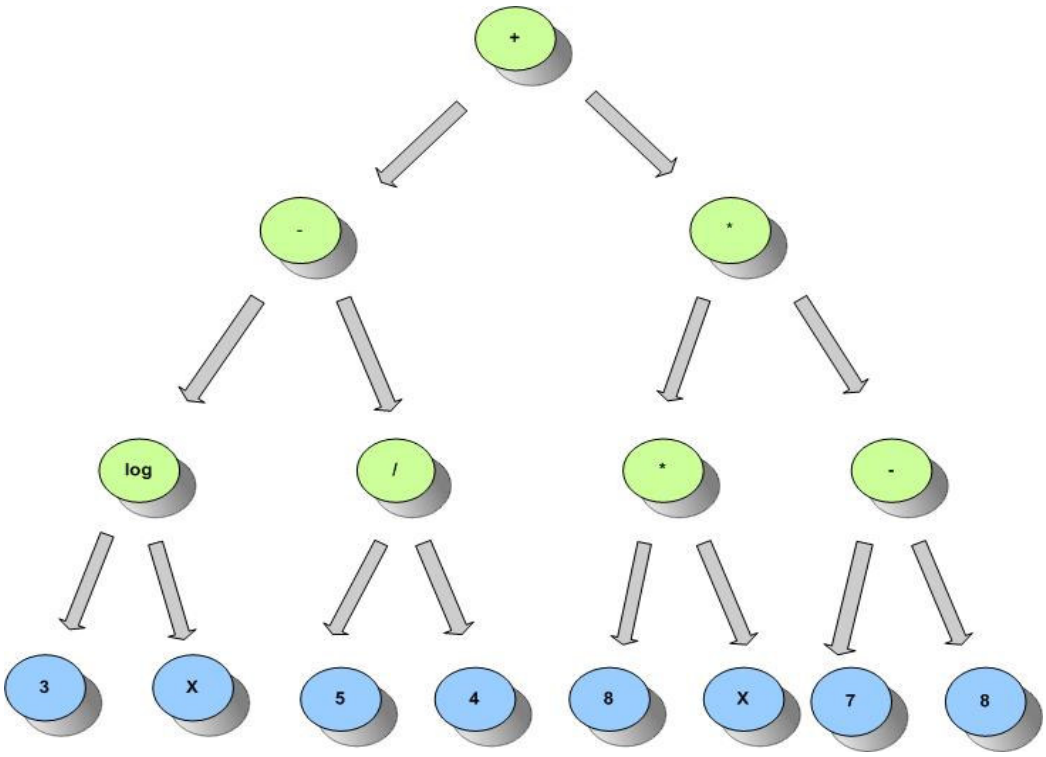
BÖLÜM 3

GENETİK PROGRAMLAMA MODELLEYİCİSİ

Yapılan çalışmada öncelikle popülasyon oluşturma yöntemi olarak ikili ağaç (binary tree) yapısı kullanılmıştır. Okuma yöntemi olarak inorder tree okuma algoritması kullanılmıştır. İkili ağaçta inorder dolaşma 3 aşamada olur;

1. İlk önce, sol-alt ağaç tekrarlanarak (recursively) dolaşılır.
2. Sonra, ana hücreye (root node) ulaşılır.
3. Sonunda da, sağ-alt ağaç tekrarlanarak (recursively) dolaşılır.

Kullanılan ağacın başlangıç büyüklüğü 7 derinliğindedir. Bir ağacın derinliği ağacın içerisinde var olan fonksiyonlar sayılarak bulunur. Şekil 3.1'de derinliği 7 olan ağaç yapısı gösterilmektedir.



Şekil 3.1 Derinliği 7 olan ağaç yapısının gösterimi.

3.1 BAŞLANGIÇ POPÜLASYONUNUN OLUŞTURULMASI

Oluşturulan ağacın büyümesi yukarıdaki bölümde bahsi geçen Doyurma yöntemi ile sağlanmıştır. Yani kullanılan ağacın bütün yaprakları aynı derinliktedir. Rastgele oluşturulan bireylerin içerisinde programın başlangıcında seçtiğimiz toplama (+), çıkarma (-), çarpma (*), bölme (/), üst alma (^), logaritma (log), sinüs (sin), kosinüs (cos) fonksiyonları ile +10 ile -10 arasında 10000 hassasiyetinde rastgele ondalıklı sayılar ve test verisinde belirtilen değişkenler olabilmektedir.

3.2 UYUMLULUK FONKSİYONUNUN BELİRLENMESİ

Uyumluluk fonksiyonu olarak programda Hata Tabanlı (Error-based) uyumluluk fonksiyonu kullanılmıştır. Bunun hesaplanması şu sekilde olmaktadır. Öncelikle olası çözüm olarak düşündüğümüz formüle dışarıdan dosyadan okuttuğumuz test verileri uygulanır herbir çözümün testte bulunan gerçek değere olan uzaklığının mutlak değeri alınır. Bütün test verisine uygulandıktan sonra bulunan mutlak değerli yaklaşık hataların aritmetik ortalaması alınır. Bu bulduğumuz değere "ayarlanmış uyumluluk" (adjust fitness) denir.

$$\text{ayaruyum} = 1/(1 + \sum_{1}^{n} \text{tveri(n)} - \text{bulunan(n)})/n \qquad (3.1)$$

Burada;

n: Test verisinde belirtilen test adedi,

ayaruyum: Ayarlanmış uyumluluk,

tveri: Test verisi,

bulunan: n. test verisinin hesaplanan uyumluluğudur.

Popülasyondaki bütün bireylerin ayarlanmış uyumlulukları bulunduktan sonra onların normal uyumlulukları bulunur. Bireylerin popülasyon içerisindeki temsil yeteneğine normal uyumluluk denir.

$$\text{noruyum} = \text{ayaruyum} / \sum_{1}^{p} \text{ayaruyum(p)} \qquad (3.2)$$

Burada;

p: Popülasyonun büyüklüğü,
noruyum: Normal uyumluluktur.

3.2.1 Test Verileri Ve Doğrulama Verilerinin Girilmesi

Genetik Programlama modelleyicisi çalışmaya başlamadan önce programı eğitmek için bir veriler topluluğuna (test verisi) ihtiyaç duyulur. Test verisinin sayısı ne kadar fazla olursa o kadar Genetik Programlama modelleyicisinin hata yapma olasılığı azalır. Yapılan çalışmada girilen test verisi adedi sonsuzdur. Herbir deney verisi içinde girilebilecek deney değişkeni sayısı en çok on adet olarak geliştirilen Genetik Programlama modelleyicisinde sınırlandırılmıştır. İstenirse bu sayı artırılabilir. Genetik Programlama modelleyicisi sonlandığında bulunan çözümün tutarlılığını test etmek için bir doğrulama (verification) veriside kullanılabilir. Doğrulama verisi olmazsa olmaz değildir. Fakat doğrulama verisi kullanıldığında sonuçtan daha da emin olunur. Doğrulama verisinin değişken adedi ve formatı test verisi ile aynı olmalıdır. Burada Genetik Programlama modelleyicisinin çalışma mantığı eğer bulunan çözüm ile doğrulama dataları çok kötü bir sonuç üretti iseler çözümün tutarsız olduğu probleme bir cevap olamadığı anlaşılır ve çözüm geçersizdir. Eğer doğrulama verileri ile çözüm daha iyi sonuçlar üretti ise o halde çözüm başarılıdır denir ve uyumluluğu ile birlikte çıktısı verilir.

3.2.2 Bireylerin Uyumluluklarının Hesaplanmasında Dikkat Edilecek Hususlar

Bireyler için uyumluluk hesaplanmasında bazı, arzu edilmeyen sonuçlara yönlenmeyi önlemek için yakınsama olarak ifade edilebilecek çözümlere başvurulur.

Sıfıra bölme hataları: Matematiksel işlem sonucunun artı yada eksi sonsuz olduğu ya da yakınsadığı durumlardır.

Sonsuz çözüm: Bilgisayar programındaki değişkene sığdırılamayan yada çözüm uzayında olamayacağı varsayılan büyük değerlerdir.

Parantez öncelikleri: Bireylerin uyumluluklarının hesaplanmasında dikkat edilmesi gerekir. Birey çok büyüdüğünde parantezleri takip etmek içinden çıkılmaz bir hal alabilir.

Yukarıda sıralanan bu hususlara dikkat edilmelidir. Programda oluşabilecek bazı hataların böylece önüne geçilebilir. Bir bireyin uygunluğu hesaplanırken fonksiyonlarından biri sonsuz değer üretti ise fonksiyon sonucunu sıfır yapabilir ya da bilgisayar programında kullandığınız değişken tipinin alabilecegi maksimum değere eşitlenebilir. Yapılan çalışmada bir fonksiyon sonsuz değer üretiyorsa o bireyin uygunluğu sıfıra eşitlenmektedir. Böylelikle çözüme uzak olduğu programa bildirmiş olunmaktadır.

3.3 SEÇİM YÖNTEMİ MEKANİZMALARI

Yapılan çalışmada seçim yöntemi olarak programda iki çesit seçim yöntemi seçmek mümkündür. Bunlar; uyum oranlı seçim ve turnuva seçimidir.

3.3.1 Uyum Oranlı Seçim Yöntemi

Bu seçim yöntemi yukarıda anlatıldığı üzere uygunluğu yüksek olanın seçilme ihtimalinin fazla olduğu seçme yöntemidir. Uyumlulukları oranında popülasyondaki bütün bireyler sanal bir çarkıfelek üzerinde yer alırlar, tekerlek popülasyon sayısınca döner ve her seferinde hangi bireyin önünde durursa o birey seçilir. Burada popülasyondaki bireyler öncelikle uyumluluklarına (ayarlanmış uyumluluk) göre büyükten küçüğe sıralanırlar. Bir bireyin seçilme ihtimali o bireyin normal uyumluluğu ve ondan önceki seçilmemiş bireylerin normal uyumluluklarının toplamıdır. Buna artan uyumluluk (cumulative fitness) denir.

$$\text{artuyum} = \text{noruyum}(n) + \sum_{1}^{n-1} \text{noruyum}(n) \qquad (3.3)$$

Burada;
n: n. Birey,
artuyum: Artan uyumluluk,
noruyum: Normal uyumluluktur.

3.3.2 Turnuva Seçimi

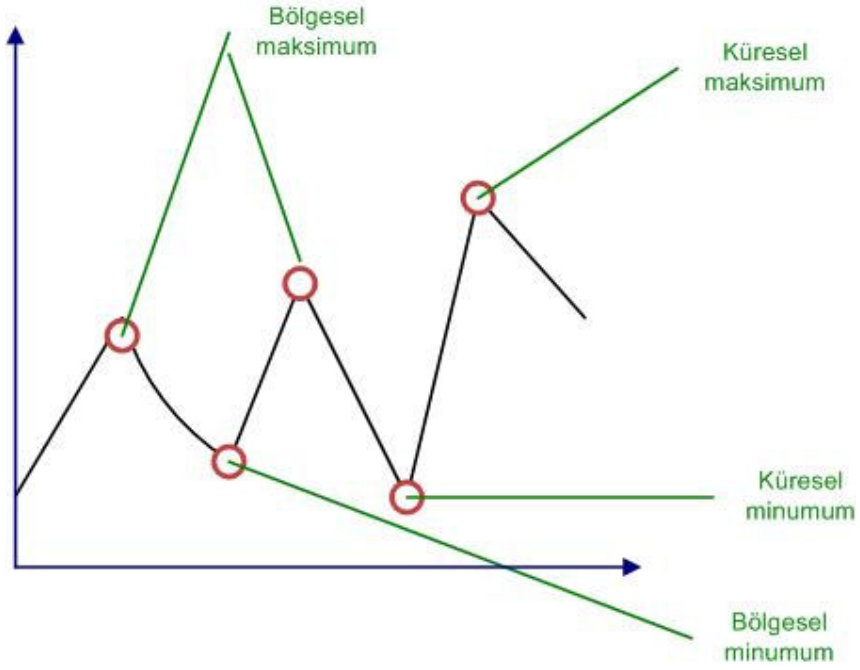
Bu seçim yönteminde popülasyonun içerisinden rastgele 7 adet birey seçilir. Bu bireyler kendi aralarında tek tek norm fitnesslerine göre kıyas edilir. Normal uyumluluğu en büyük olan birey yeni popülasyona atılır.

3.4 ÇAPRAZLAMA

Yapılan çalışmada çaprazlama için öncelikle popülasyonun içerisinden rastgele iki adet birey seçilir. Bu seçilen iki adet bireyin çaprazlanıp çaprazlanmayacağı Genetik Programlama modelleyicisinin başlangıcında belirtilen çaprazlama oranı ile ifade edilir. Rastgele 0-1 arasında bilgisayar ondalıklı bir değer döndürür, eğer bu bulunan değer çaprazlama oranından düşükse bu iki bireyin çaprazlanması gerekiyor demektir. Çaprazlanacak olan bireylerden rastgele bir çaprazlama noktası belirlenir. Bu çaprazlama noktası kök olmamalıdır. Burada çaprazlama noktası bir fonksiyon da olabilir bir terminal de olabilir ve seçilen çaprazlama noktalarından bireyler kopartılarak kopartılan parçalar karşılıklı değiştirilir. Burada dikkat edilmesi gereken nokta çaprazlama noktası eğer bir fonksiyonsa bütün alt terminalleri ve fonksiyonları ile beraber koparıldığıdır. Çaprazlama sonrasında Genetik Programlama modelleyicisi yeni oluşan bireylerin başlangıçta belirtilen maksimum ağaç derinliğini aşıp aşmadıklarını kontrol eder. Eğer yeni oluşan bireyler yada bireylerden biri maksimum ağaç derinliğini aşıyorsa o bireyler yada birey yeni popülasyona alınmaz ve yeniden çaprazlama yapılır.

3.5 MUTASYON

Popülasyondaki rastgele seçilen bir bireyin mutasyona uğrayıp uğramayacağı Genetik Programlamanın girişinde belirtilen mutasyon oranıyla belirlenir. 0-1 arasında rastgele bulunan ondalıklı değer eğer mutasyon oranından küçükse o zaman o bireyde mutasyon yapılır değilse yapılmaz. Mutasyon yapılmaya karar verilen bireyde rastgele bir mutasyon noktası belirlenir. Bu mutasyon noktası kök olmamalıdır. Burada mutasyon noktası bir fonksiyon da olabililir bir terminal de olabilir. Eğer seçilen nokta terminalse oraya yeni bir terminal eklenir (var olan terminal setinden bir değişkende olabilir bir sabit sayıda olabilir). Eğer seçilen nokta bir fonksiyonsa o zaman o fonksiyon kendi ve bütün alt yaprakları ile birlikte mutasyona uğrar. Fonksiyonlar kendi içerisinde (var olan fonksiyon setinden) terminaller kendi içerisinde değişime uğrarlar. Mutasyon bölgesel minumum veya bölgesel maksimumlara takılmamak için Genetik Programlamada hayati öneme sahiptir (Şekil 3.2).



Şekil 3.2 Bölgesel minumum ve bölgesel maksimumların gösterilmesi.

Mutasyon sonrası artık Genetik Programlamada bir jenerasyon sona ermiştir. Bunun sonucunda eğer program sonlandırma kriterlerine ulaşmıssa sonlanır değilse devam eder. Yapılan çalışmada sonlandırma kriterleri 2 tanedir.

Programın maksimum jenerasyon sayısına ulaşması,

Programın istenilen uyumluluk seviyesine ulaşması.

Genetik Programlamada istenilen uyumluluk seviyesi genelde %10 hata civarı olarak kabul edilmektedir. Yapılan programda istenilen hata seviyesi %0 dır. Ama bunun yanında %10 hata seviyesine kadar başarılı kabul etmekte mümkündür. Yani program herhangi bir bireyin ayarlanan uyumluluğu bir oluncaya kadar ya da maksimum jenerasyon sayısına ulaşıncaya kadar devam etmektedir.

BÖLÜM 4

TERSİNE PROBLEM UYGULAMALARI

Problemler genellikle aşağıdaki şekilde gösterilen biçimde çözüme kavuşurlar. Problem girdilerin belli bir oranda birbirlerinden etkileşerek çözümü üretmeleri ile çözüme ulaşır. Tersine problem uygulamaları değişkenlerin birbirleri ile etkileşimini formülize eder (Şekil 4.1).

Şekil 4.1 Tersine problemlerin genel yapısı.

Aşağıda problem uygulamaları için örnek denemeler sunulmuştur.

Örnek 1 deki denemelerde kullanılan bilgisayarın teknik özellikleri: Windows XP (sp2) işletim sistemi, İntel Pentium IV 1.40 GHz işlemci ve 760 Mb hafızadır.

Örnek 2 deki denemelerde kullanılan bilgisayarın teknik özellikleri: Windows 2003 server işletim sistemi, 4 adet Intel xeon 2.20 Ghz işlemci ve 4 GB hafızadır.

Denemelerde kullanılan modelleyici: Visual Basic.Net kullanılarak tasarlanmıştır. Elde edilen çözüm içinde bazı notasyonlar (program modellenirlen eklenen) bulunmaktadır. Bunlar ilgili programı okumada kolaylıklar sağlamaktadır. Formüllerde yer alan "!" işareti

kendinden önceki sabitin değerinin negatif olduğunu belirtir. Parantezler içerisinde bulunan "," fonksiyonun parametrelerini ayırmak için kullanılmıştır. Bir matematiksel değere sahip değildir.

4.1 ÖRNEK 1

Örnek 1 de çok fazla karmaşık olmayan ve cevabı bilinen standart bir 3. dereceden üstel fonksiyon üzerinde çalışılmıştır. Burada amaç tasarlanan Genetik Programlama modelleyicisinin performansını test etmektir. Uygulanan problemin test verisi aşağıdaki gibidir (Çizelge 4.1).

Çizelge 4.1 Örnek 1'in test verileri.

Değişken	Sonuç
1	3
2	14
3	39
7	399
9	819

Yukarıda görülen test verisi tanıdık bir fonksiyondur. Bu fonksiyon

$$f(a) = a + a^2 + a^3 \tag{8}$$

Bu problem aşağıda üç ayrı deneme ile çözüme kavuşturulacaktır.

Kullanılan terminal seti, +10 ile -10 arasında ve 10000 hassasiyetinde ondalıklı olarak rastgele değer alır. Toplama (+), çıkarma (-), çarpma (*), bölme (/) fonksiyonları kullanılmıştır. Bölümden (/) sonra oluşabilecek sıfıra bölme hatalarından sakınmak için modelleyici sıfıra bölme hatası ürettiğinde o bireyin uygunluğu sıfır alınarak bir dahaki jenerasyonda popülasyondaki yerini almaması sağlanmıştır. Aynı şekilde diğer operatörlerden (+, - ,*) kaynaklanan sonsuza yakınsama probleminden ötürü sonsuza yakınsayan fonksiyon sonuçlarının olduğu bireylerin de uyumlulukları sıfır alınarak bir dahaki jenerasyonda popülasyondaki yerlerini almamaları sağlanmıştır.

Aşağıdaki formüllerin gösterimine LISP gösterimi denir (Hicklin, 1986; Fujiki, 1986; Fujiki and Dickinson, 1987). LISP gösterimindeki notasyonda önce fonksiyon yazılır sonra bu fonksiyonun işlem yapacağı parametreler sırasıyla birinci ve ikinci parametre olarak sıralanır. Örneğin (+ 4 5) ifadesi 4 ile 5 in toplanacağı anlamı taşımaktadır. Yani (4 + 5) demektir. Bir örnek daha verecek olursak (-(* 3 9) (+ 5 1)) ifadesi ((3 * 9)–(5 + 1)) ifadesi ile eşdeğerdir. LISP gösteriminde en çok dikkat edilmesi gereken nokta parantezler ve parantez öncelikleridir. Genetik Programlama modelleyicileri tasarlanırken LISP'in kendi programlama dili kullanılabildiği gibi diğer programlama dilleri de kullanılmaktadır (Pascal, Frotran, C, Java, Visual Basic, Delphi). Esasen Genetik Programlama modelleyicisi tasarlamada birkaç tane yöntem vardır (Robinson, 2001).

LISP tabanlı programlama: Bu tür programlamada LISP dilinin Genetik Programlamada birçok kolaylıklar sağladığını görürüz. Buna karşın diğerlerine göre yavaştır.

Lineer Genetik Programlama: Burada asıl zorluk fonksiyonları ve ağaçları programlama dilinde yazabilmek ve hesaplamalarda öncelikleri hesap etmededir. Buna karşın hızlıdır.

Makina kodlu Genetik Programlama: Bu tür programlamada makina kodları kullanılır yazımı çok zordur. Fakat çok hızlıdır.

Stack (yığın) tabanlı Genetik Programlama: Bu tür programlamada yığın yapıları kullanılır. Yığın yapıları programlamada (özellikle parantez önceliğinde) kolaylıklar sağlar.

4.1.1 Deneme 1

4.1.1.1 Modelleyici Parametreleri

Deneme 1 için popülasyon büyüklüğü 250, çaprazlama oranı 0.9, mutasyon oranı 0.05, çaprazlama sonrası maksimum derinlik 100 ve maksimum jenerasyon 100 alınmıştır. Seçim yöntemi olarak turnuva seçimi belirlenmiştir.

4.1.1.2 Bulunan Çözüm

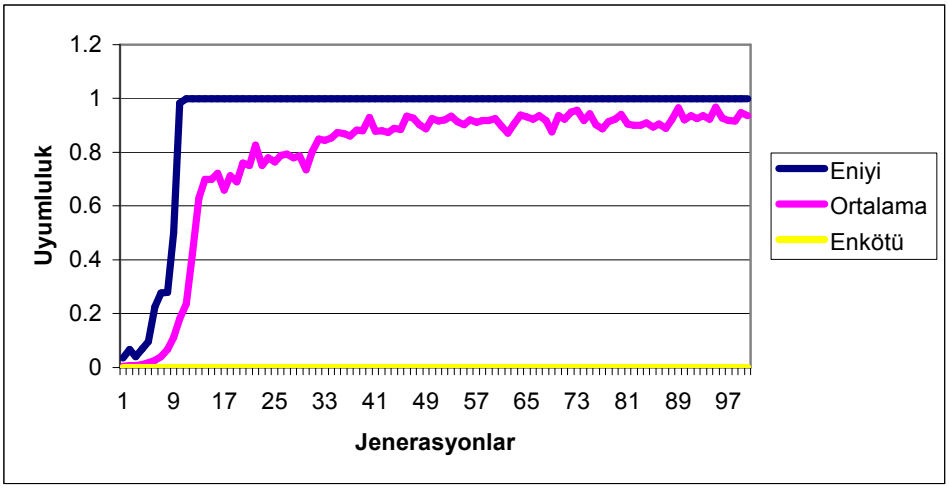
Bu denemedeki parametrelerle program 10 kez bilgisayarda uygulanmıştır. Her bir deneme yaklaşık 1-2 dakika civarında sürmüştür. Modelleyici bu denemelerin hepsinde de kabul edilebilir hata sınırı olan %10 hata sınırının altında programlar üretmiştir. Bunlardan en iyi olanının teknik detayları aşağıda anlatılmaktadır.

Modelleyici 100 adet jenerasyonun (modelleyicinin çalışması bir dakika elli saniye sürmüştür) sonucunda 1 uyumluluk değerine sahip aşağıda gösterilen formülü en iyi olarak bulmuştur.

F(a) =

(-(+(*(+(*a,a,),a,),a,),a,)(/(-2.5758,2.5758,),(-a,a,),)) (4.1)

Aşağıda herbir jenerasyon sonrası en iyi, ortalama ve en kötü bireylerin uyumlulukları gösterilmiştir (Şekil 4.2).



Şekil 4.2 Örnek 1-Deneme 1'in uyumluluk grafiği.

Denklem (4.1)'de elde edilen model 11. jenerasyonda elde edilmiştir.

4.1.2 Deneme 2

4.1.2.1 Modelleyici Parametreleri

Deneme 2 için popülasyon büyüklüğü 500, çaprazlama oranı 0.9, mutasyon oranı 0.05, çaprazlama sonrası maksimum derinlik 100 ve maksimum jenerasyon 100 alınmıştır. Seçim yöntemi olarak uyum oranlı seçim yöntemi belirlenmiştir.

4.1.2.2 Bulunan Çözüm

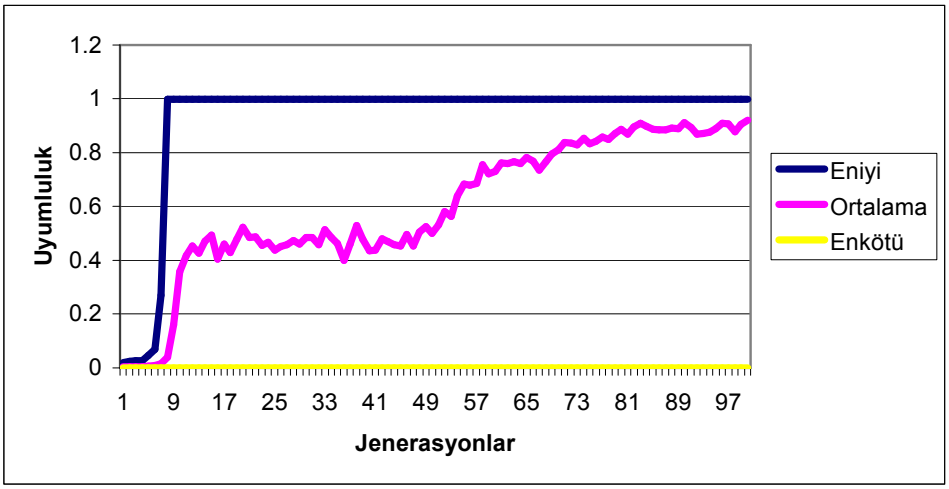
Bu denemedeki parametrelerle program 10 kez bilgisayarda uygulanmıştır. Her bir deneme yaklaşık 1-2 dakika civarında sürmüştür. Modelleyici bu denemelerin hepsinde de kabul edilebilir hata sınırı olan %10 hata sınırının altında programlar üretmiştir. Bunlardan en iyi olanının teknik detayları aşağıda anlatılmaktadır.

Modelleyici 100 adet jenerasyonun (modelleyicinin çalışması bir dakika elli saniye sürmüştür) sonucunda 1 uyumluluk değerine sahip aşağıda gösterilen formülü en iyi olarak bulmuştur.

F(a) =

(+(*(*a,a,)(+(/a,a,),a,)),a,) (4.2)

Aşağıda her bir jenerasyon sonrası en iyi, ortalama ve en kötü bireylerin uyumlulukları gösterilmiştir (Şekil 4.3).



Şekil 4.3 Örnek 1-Deneme 2'nin uyumluluk grafiği.

Denklem (4.2)'de elde edilen model 8. jenerasyonda elde edilmiştir.

4.1.3 Deneme 3

4.1.3.1 Modelleyici Parametreleri

Deneme 3 için popülasyon büyüklüğü 500, çaprazlama oranı 0.9, mutasyon oranı 0.05, çaprazlama sonrası maksimum derinlik 100 ve maksimum jenerasyon 100 alınmıştır. Seçim yöntemi olarak turnuva seçimi belirlenmiştir.

4.1.3.2 Bulunan Çözüm

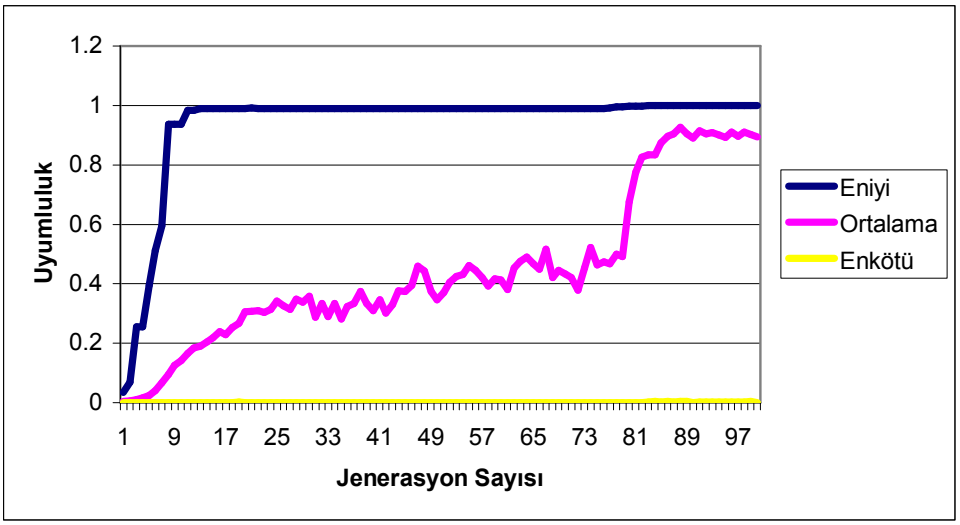
Bu denemedeki parametrelerle program 10 kez bilgisayarda uygulanmıştır. Her bir deneme yaklaşık 1-2 dakika civarında sürmüştür. Modelleyici bu denemelerin hepsinde de kabul edilebilir hata sınırı olan %10 hata sanırının altında programlar üretmiştir. Bunlardan en iyi olanının teknik detayları aşağıda anlatılmaktadır.

Modelleyici 100 adet jenerasyonun (modelleyicinin çalışması onbir dakika elli saniye sürmüştür) sonucunda 0.999 uyumluluk değerine sahip aşağıda gösterilen formülü en iyi olarak bulmuştur.

F(a) =

(+((*(*a,a,),a,),1.1201,)(+(+0.7731!,(+(*a,a,)(+a,2.0988,)),),(/4.5284!,(+(+2.4409,(+(+2.4409,(+(+2.0988,2.0988,),(+(+(+2.4409,(+(+(+2.0988,(+(*a,4.5284!,)(+a,a,)),),2.0988,)(+a,2.4409,)),),(/(+(*(+a,a,),4.5284!,)(+5.0439,2.0988,)),(+(+a,(+2.0988,(+a,a,),)),(+(+2.0988,2.4409,),(+(+(+1.1201,(+(+(+2.0988,(+(*a,4.5284!,)(+a,a,)),),2.0988,)(+2.0988,2.0988,)),),(/(+(*a,4.5284!,)(+a,2.4409,)),(+(+a,(+(+5.0439,(+a,(+a,(*a,4.5284!,),),),)),(+a,a,),)),2.0988,)),),1.1201,),),)),),1.1201,),)),(+2.4409,a,),)),a,)),)) (4.3)

Aşağıda her bir jenerasyon sonrası en iyi, ortalama ve en kötü bireylerin uyumlulukları gösterilmiştir (Şekil 4.4).



Şekil 4.4 Örnek 1-Deneme 3'ün uyumluluk grafiği.

Denklem (4.3)'de elde edilen model 92. jenerasyonda elde edilmiştir.

4.2 ÖRNEK 2

Uygulanan problemin test verisi aşağıdaki gibidir (Çizelge 4.2). İlgili çizelgede yer alan yatay sıralar on adet deneyi ifade etmektedir. Değişken1 (a) ilgili malzeme için kullanılan kesici devri, Değişken2 (b) ilerleme değeri, Değişken3 (c) kesici için talaş derinliği, Değiken4 (d) kesicinin yanal adımıdır. En son sütundaki sonuç (f) ise ilgili değişkenler sonunda elde edilen yüzey pürüzlülük değeridir (Sakarya, 2005). Buradaki problem, uygun bir formülün bu deney değerlerine göre elde edilmesidir. Problem tersine bir problemdir.

Çizelge 4.2 Örnek 2'nin test verileri.

Değişken1 (a)	Değişken2 (b)	Değişken3 (c)	Değişken4 (d)	Sonuç (f)
450	120	0.2	2	1.2875
600	180	0.2	4	1.6775
550	120	0.4	5	1.0025
550	240	0.2	5	2.285
550	300	0.2	3	2.215
600	120	0.6	3	1.5575
500	180	0.4	3	2.895
600	240	0.4	2	1.0225
500	120	0.8	4	1.9675
450	180	0.6	5	2.375

Uygulanan problemin doğrulama verisi aşağıdaki gibidir (Çizelge 4.3).

Çizelge 4.3 Örnek 2'nin doğrulama verileri.

Değişken1 (a)	Değişken2 (b)	Değişken3 (c)	Değişken4 (d)	Sonuç (f)
550	240	0.6	4	3.145
550	180	0.8	2	1.7775
450	300	0.4	4	3.75
500	300	0.6	2	3.1
600	300	0.8	5	3.8675
450	240	0.8	3	4.11

Kullanılan fonksiyon seti toplama (+), çıkarma (-), çarpma (*), bölme (/) ve üst alma (^) dır. Bunun dışındaki diğer parametre ve hassasiyetler örnek 1 ile aynıdır. Aşağıdaki parametrelerle denemeler 10 kez denenmiştir ve bu 10 denemenin sonucunda en iyi olan program örnek olarak gösterilmiştir.

4.2.1 Deneme 1

4.2.1.1 Modeleyici parametreleri

Deneme 1 için popülasyon büyüklüğü 5000, çaprazlama oranı 0.9, mutasyon oranı 0.05, çaprazlama sonrası maksimum derinlik 200 ve maksimum jenerasyon 100 alınmıştır. Seçim yöntemi olarak turnuva seçimi belirlenmiştir.

4.2.1.2 Bulunan Çözüm

Bu denemedeki parametrelerle program 10 kez bilgisayarda uygulanmıştır. Çizelge 4.4 bu deneylerin sonuçları gösterilmiştir. Her bir deneme yaklaşık 10 saat civarında sürmüştür. Bu denemelerin 3 ünde modelleyici kabul edilebilir hata sınırı olan %10 hata sanırının altında programlar üretmiştir. Bunlardan en iyi olanının teknik detayları aşağıda anlatılmaktadır (Çizelge 4.4).

Çizelge 4.4 Örnek 2-Deneme 1'nin sonuç listesi.

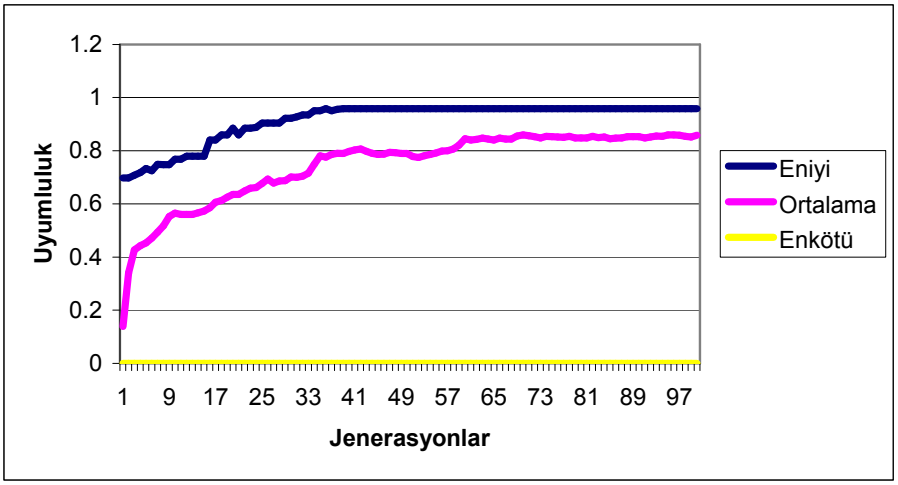
Deney No	Bulunan Uyumluluk
1	0.885
2	0.888
3	0.871
4	0.958
5	0.922
6	0.891
7	0.960
8	0.868
9	0.891
10	0.890

Modelleyici 100 adet jenerasyonun (modelleyicinin çalışması on saat yirmi dakika otuzbeş saniye sürmüştür) sonucunda 0.960 uyumluluk değerine sahip aşağıda gösterilen formülü en iyi olarak bulmuştur. Çizelge 4.3'te verilen dogrulama verileri ile yapılan doğrulama testinde doğrulama verisinin sonucun tutarlılığını onayladığı görülmüştür.

F(a,b,c,d)=

(^(*(+(/5.9755!,d,),(+(/(/(/b,4.1846,),a,),(/4.1846,(^(/7.4861,d,)(^(-(a,b,)
(*(/(/b,b,),4.8373!,),a,)),c,)),),),(+(/(/(/b,(^(-(-a,b,) (*(/(/b,b,),7.4861,),a,)),c,),),a,)
,(/4.1846,(^(/4.8373!,2.1221,)(^(-(-a,b,) (*(/d,(+c,a,)),a,)),c,)),),),d,),),)
(+(^(/b,7.4861,),a,),(+(+(^(/7.4861,d,)(^(-(-a,b,) (*(/4.1846,(+c,a,)),a,)),c,))
,(/b,d,),),(+(-(-a,b,)(*(/d,(+(/4.1846,(^(/7.4861,d,)(^(-(-a,b,)(*(/(/b,b,)
(+c,a,)),a,)),c,)),),a,)),a,)),(/6.0266!,4.1846,),)),)))(/(/b,4.1846,)(+b,a,))) (4.4)

Aşağıda her bir jenerasyon sonrası en iyi, ortalama ve en kötü bireylerin uyumlulukları gösterilmiştir (Şekil 4.5).



Şekil 4.5 Örnek 2-Deneme 1'in uyumluluk grafiği.

Denklem (4.4)'de elde edilen model 36. jenerasyonda elde edilmiştir.

4.2.2 Deneme 2

4.2.2.1 Modelleyici Parametreleri

Deneme 2 için popülasyon büyüklüğü 5000, çaprazlama oranı 0.9, mutasyon oranı 0.05, çaprazlama sonrası maksimum derinlik 200 ve maksimum jenerasyon 100 alınmıştır. Seçim yöntemi olarak uyum oranlı seçim yöntemi belirlenmiştir.

4.2.2.2 Bulunan Çözüm

Bu denemedeki parametrelerle program 10 kez bilgisayarda uygulanmıştır. Çizelge 4.5 de bu deneylerin sonuçları gösterilmiştir. Her bir deneme yaklaşık 10 saat civarında sürmüştür. Bu denemelerin hiçbirinde modelleyici kabul edilebilir hata sınırımız olan %10 hata sanırının altında program üretememiştir. Bunlardan en iyi olanının teknik detayları aşağıda anlatılmaktadır (Çizelge 4.5).

Çizelge 4.5 Örnek 2-Deneme 2'nin sonuç listesi.

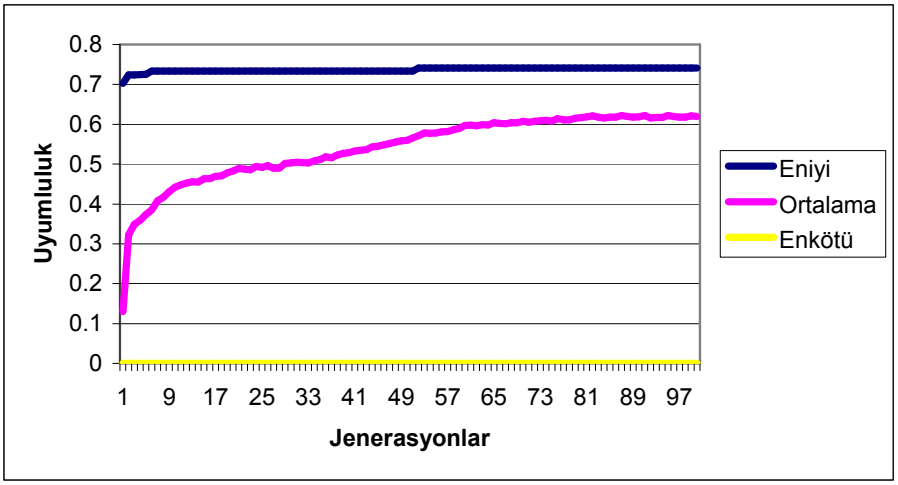
Deney No	Bulunan Uyumluluk
1	0.741
2	0.720
3	0.705
4	0.691
5	0.694
6	0.723
7	0.708
8	0.679
9	0.706
10	0.733

Modelleyici 100 adet jenerasyonun (modelleyicinin çalışması on saat bir dakika elli saniye sürmüştür) sonucunda 0.741 uyumluluk değerine sahip aşağıda gösterilen formülü en iyi olarak bulmuştur. Çizelge 4.3'te verilen dogrulama verileri ile yapılan doğrulama testinde doğrulama verisinin sonucun tutarlılığını onayladığı görüldü.

F(a,b,c,d) =

```
(^(+b,(^(+(-(/c,a,),d,),(+a,(+(/(^2.2272!,(/d,(^(^(-a,b,)(-(^(-
(+a,c,),(^(*(^(+4.1275,(^c,c,),)(*b,b,)),c,)(-8.3945,a,)),)(-c,a,)),a,)),(-
b,0.1175!,)),)),(+9.0807!,(/(-a,(+(-b,5.9786!,)(+6.2349,(+(*a,(^4.1548,d,)),(-(-
c,b,),(*d,(^c,b,),),),),)),)(/(^a,b,),c,)),),),d,),),),(+(/a,a,),c,),),)(*(/b,(^(+(+(/d,(^(^(-
3.9286!,b,)(-(^(-c,c,)(-(+(^(/d,(-b,a,),),(-b,d,)),(-(^9.6037,a,),d,),),c,)),a,)),(-b,d,)),),(-
c,(^0.0288!,(/d,b,),),),),b,),(+(-(*(-(*a,(^(*0.9960,0.0688!,)(-
a,a,)),)(/a,b,)),b,),b,),a,),),),d,))                                    (4.5)
```

Aşağıda her bir jenerasyon sonrası en iyi, ortalama ve en kötü bireylerin uyumlulukları gösterilmiştir.



Şekil 4.6 Örnek 2-Deneme 2'nin uyumluluk grafiği.

Denklem (4.5)'de elde edilen model 52. jenerasyonda elde edilmiştir.

4.2.3 Deneme 3

4.2.3.1 Modelleyici Parametreleri

Deneme 3 için popülasyon büyüklüğü 5000, çaprazlama oranı 0.8, mutasyon oranı 0.2, çaprazlama sonrası maksimum derinlik 200 ve maksimum jenerasyon 100 alınmıştır. Seçim yöntemi olarak turnuva seçim yöntemi belirlenmiştir.

4.2.3.2 Bulunan Çözüm

Bu denemedeki parametrelerle program 10 kez bilgisayarda uygulanmıştır. Çizelge 4.6 de bu deneylerin sonuçları gösterilmiştir.. Her bir deneme yaklaşık 10 saat civarında sürmüştür. Bu denemelerin hepsinde modelleyici kabul edilebilir hata sınırımız olan %10 hata sanırının altında program üretmiştir. Bunlardan en iyi olanının teknik detayları aşağıda anlatılmaktadır (Çizelge 4.6).

Çizelge 4.6 Örnek 2-Deneme 3'nin sonuç listesi.

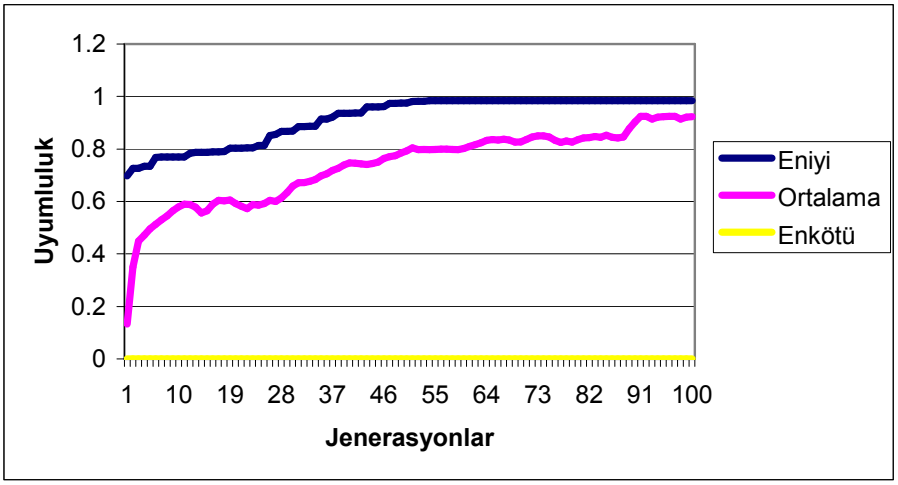
Deney No	Bulunan Uyumluluk
1	0.984
2	0.960
3	0.910
4	0.942
5	0.948
6	0.955
7	0.951
8	0.964
9	0.901
10	0.974

Modelleyici 100 adet jenerasyonun (on iki saat otuz dakika sürdü) sonucunda 0.984 uyumluluk değerine sahip aşağıda gösterilen formülü en iyi olarak bulmuştur. Çizelge 4.3'te verilen dogrulama verileri ile yapılan doğrulama testinde doğrulama verisinin sonucun tutarlılığını onayladığı görüldü.

F(a,b,c,d) =

(^(+(*(^(/(^(*a,5.1595,)(-(/(+c,b,)(-a,(/(-b,a,)(+4.3419!,(+(*(^(/(^(^(/(^(*a,(-a,(/(-b,c,)(+c,a,)),),)(-(/(+c,b,)(-a,(/(-b,a,)(+4.3419!,(+(*(^(/(^(*d,b,)(-(/(+9.6697,b,)(-a,(/(-b,c,)(+7.6396,a,)),)),c,)),d,)(/4.3419!,5.1595,)),c,),(-d,0.4982!,)),)),)),)),c,)),d,)(/4.3419!,9.6697,))(-(/(+c,b,)(-a,(/(-b,c,)(+7.6396,a,)),)),c,)),d,)(/4.3419!,9.6697,)),c,),(-d,0.4982!,)),)),)),c,)),d,)(/4.3419!,9.6697,)),c,),(-d,0.4982!,))(/(+0.4982!,b,)(-a,(/5.1595,(^(/(^(*a,5.1595,)(-(/(+c,b,)(-a,(/(-b,(-(-a,(/(-b,(/(+(+(*(^(/(^a,(-(/(+c,b,)(-a,(/(-b,a,)(+4.3419!,(+(*(^(/1.4353,d,)(/4.3419!,9.6697,)),c,),(-d,0.4982!,)),)),)),)),c,)),d,)(/(+c,b,),9.6697,)),c,),(-d,0.4982!,)),(-a,(/(-a,(/(+(*c,4.3419!,),(-d,0.4982!,))(-a,c,)),),c,),)),a,),),(-b,(/(/(+c,b,)(-a,c,)),(-(+7.6396,a,),b,),),),),),b,),),(-b,(/(/(+c,b,)(-a,9.6697,)),(-a,b,),),),),)),c,)),d,)(/(^d,c,)(/(+(/(+4.3419!,(-a,4.3419!,)),a,),(-a,b,)),a,))),),))) (4.6)

Aşağıda her bir jenerasyon sonrası en iyi, ortalama ve en kötü bireylerin uyumlulukları gösterilmiştir.



Şekil 4.7 Örnek 2-Deneme 3'ün uyumluluk grafiği.

Denklem (4.6)'da verilen model 56. jenerasyonda elde edilmiştir.

BÖLÜM 5

SONUÇ

Yapılan çalışmada çözüm uzayı oldukça büyük ve karmaşık olan tersine problemler için Evrimsel Algoritmaların bir alt dalı olan Genetik Programlama ile ilgili bir yazılım gerçekleştirilerek oldukça başarılı sonuçlar elde edilmiştir (Arslan ve Göloğlu, 2005).

Popülasyon oluşturulurken kullanılan ağacın derinliğinin (3, 5, 7) problemin çözümüne etkisinin çok da fazla olmadığı görülmüştür. Popülasyon derinliği farklı olduğunda da program aynı sonuçları üretebilmiştir. Çünkü başlangıç popülasyonunda oluşan bireyler sonradan çaprazlama işlemcisinin etkisiyle büyümektedirler.

Popülasyonun büyüklüğünün problemin çözümüne olumlu yönde yaptığı katkı görülmüş, daha az sayıdaki jenerasyonda daha etkin sonuçlar çıkartmıştır. Fakat daha yavaş çalıştığı gözlemlenmiştir. Bu, popülasyon büyüdükçe onlarla genetik operatörlerin işlenmesi daha çok zaman aldığı nedeniyle açıklanabilir.

Genetik Programlama modelleyici tasarlanırken görülen en büyük zorluk programın yavaş çalışması ve bellekte fazlaca yer kaplaması olmuştur. Bunun nedeni bilgisayarın düz yazı (string) tipindeki işlemleri yaparken (ağaçlar tasarlanırken düz yazı tipinde kaydedildi) çok zaman kaybetmesidir. Ayrıca bir birey içindeki fonksiyonların çok fazla olması da yine zaman açısından bilgisayarı zorlayan konuların başında gelir.

Seçim yöntemleri olarak uyum oranlı seçim ve turnuva seçimi kullanılmıştır. Bu ikisi arasında standart basit problemlerde uyum oranlı seçim yönteminin daha önce sonuca gittiğini görülmüştür (deney 1 in deneme sonuçları). Fakat daha karmaşık olan (tersine problem) problemlerde turnuva seçiminin daha başarılı olduğu görülmüştür. Yapılan denemelerde turnuva seçimi %10 hata sınırının altında sonuçlar üretti. Bu sonuç iki noktayı göstermektedir. Birincisi, Genetik Programlamada rastgelelikten faydalanıldığı için program tekrar tekrar çalıştırmalıdır. Bu aynı parametrelerle ve farklı parametrelerle

olabilir. Hatta ne kadar çalıştırılırsa o kadar doğru sonuçlara ulaşma olasılığı fazla olmaktadır (yapılan çalışmalarda çözüme ulaşma zamanı çok fazla olduğu için 10 adet tekrarlama yapılmıştır). İkincisi, yukarıda bahsedilen turnuva seçimi karmaşık problemlerde (tersine problem) uyum oranlı seçime göre daha başarılıdır.

Test verisinde ne kadar fazla örneklem alındı ise, sonuçta Genetik Programlama modelleyicisinin ürettiği çözümün o derece tutarlı olduğu gözlemlenmiştir. Doğrulama verisinin genellikle çok fazla örneklem olduğunda bir etkisinin olmadığı yalnız sınırlı sayıda test örneklemi ile çalışılıyorsa hayati önemi olduğu görülmüştür.

Çaprazlama işlemcisinin 0.5 den aşağı tutulmaması gerektiği görülmüş aksi takdirde programın evrilmesi istenilen uyumluluk düzeyine erişmesi mümkün olmadığı saptanmıştır.

Mutasyon oranının 0.2 den daha büyük seçilmesinin çözüme olumsuz yönde katkı yaptığı görülmüştür. Mutasyon anına kadar gelmiş uyumluluğu iyi bireylerin kaybolmasına neden olmuştur.

Genel olarak Genetik Programlamanın çözüm uzayı oldukça büyük ve karmaşık olan ve geleneksel optimizasyon yöntemleri ile çözülemeyen problemleri çözmek için büyük kolaylıklar sağladığı görülmüştür. Genetik Programlamın sonucunda ortaya çıkan sonucun sade, kolay anlaşılır olduğu gözlemlenmiştir.

Bundan sonra yapılacak olan çalışmalarda Genetik Programlamanın nasıl daha hızlı sonuca gidebileceği üzerinde çalışmalar yapılabilir. Zira çözüm zamanının çok uzun olması ve güçlü makinalara ihtiyaç duyulması Genetik Programlama için bir dezavantaj olduğu açıktır.

Yapılan literatür taramasında Ülkemizde Genetik Programlama ile ilgili olarak çok fazla çalışma yapılmadığı görüldü. Yapılan bu çalışma bundan sonraki yapılacak olan çalışmalar için faydalı olacaktır.

KAYNAKLAR

Baker, J. E., (1985). Reducing bias and inefficiency in the selection algorithm. In Proceedings of an International Conference on Genetic Algorithms and Their Applications. Erlbaum.

Boun, (2001) Bogaziçi Üniversitesi, Robot Sitesi, http://robot.cmpe.boun.edu.tr/593

Ege, (2004) Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü http://bilmuh.ege.edu.tr/~bayindir/Evolap

Giles, C.L., Lawrence, S. and Tsoi, A.C., (2001), Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Interface, Machine Learning, vol.44, no.1/2, July/August, pp. 161-183.

Fujiki, Cory, and Dickinson, John., (1987). Using the genetic algorithm to generate LISP source code to solve the prisoner's dilemma. In Grefenstette, John J. (editor), Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms. Erlbaum.

Fujiki, Cory., (1986). An Evaluation of Holland's Genetic Algorithm Applied to a Program Generator. M.S. thesis, Department of Computer Science, University of Idaho, USA.

Hancock Peter J.B., (1994) An empirical comparison of selection in evolutionary algorithms, Department of Psychology University of Stirling, Scotland FK9 4LA

Hicklin, Joseph F., (1986). Application of the Genetic Algorithm to Automatic Program Generation. M.S. thesis, Department of Computer, Science, University of Idaho, USA.

Kapishnikov, A., (2002), Optimization of Technical Rules on the Basis of Intelligent Hybrid Systems, Scientific Proceedings of Riga Technical University, Information Technology and Management Science, Latvia.

Koza, J. (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection, The MIT Press, USA.

Koza, J. (1994) Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge, MA, USA.

Milfelner, M., Kopac, J., Cus, F. and Zuperl, U., (2005), Genetic equation for the cutting force in ball-end milling, Journal of Materials Processing Technology, vol. 164-165, pp. 1554-1560.

Maza M. De la and B. Tidor (1991) Increased Flexibility in Genetic Algorithms: The Use of Variable Boltzmann Selective Pressure to Control Propagation, Proc. of the ORSA CSTS Conference - Computer Science and Operations Research: New Developments in their Interfaces, pp 425-440.

Nastran, M. and Balic, J., (2002), Prediction of metal wire behavior using genetic programming, Journal of Materials Technology, vol. 122, pp. 368-373.

Platel, M.D., Chami, M., Clergue, M. and Collard, P., (2005). Teams of Genetic Predictors for Inverse Problem Solving, in Proceedings of Genetic Programming: 8th European Conference, EuroGP 2005, eds: Keijzer, M., Tettamanzi, A., Collet, P. et al., March 30 - April 1, Lausanne, Switzerland.

Robinson A. (2001) Genetic Programming: Theory Implementation, and the Evolution of Unconstrained Solutions, Division III Thesis Hampshire College **,** pp 20-22

Toropov, V.V. and Alvarez, L.F. (1998), Approximation model building for design optimization using genetic programming methodology, AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, USA.

Sakarya, N., (2005)**,** Cep işlemede takım yolu hareketlerinin takım ve yüzey parametrelerine etkilerinin incelenmesi, Bilim Uzmanlığı Tezi, Fen Bilimleri Enstitüsü, Zonguldak Karaelmas Üniversitesi, Karabük.

Trivailo, P. M., Dulikravich, G. S., Sgarioto, D and Gilbert, T., (2004b). Inverse Problem of Aircraft Structural Estimation: Application of Neural Networks, Inverse Problems, Design and Optimization Symposium, March 17-19, Rio de Janeiro, Brazil.

Trivailo, P.M., Gilbert, T., Glessich, E. And Sgarioto, D., (2004a). Inverse Problem Of Aircraft Structural Parameter Identification: Application of Genetic Algorithms Compared With Artificial Neural Networks, Inverse Problems, Design And Optimization Symposium, Rio De Janeiro, Brazil.

Whitley, D. (1989) The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In Schaffer, J. D.(editor), Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann.

Xu, Y.G. and Liu, G.R. (2002), A Novel Hybrid Genetic Algorithm And Its Application To Inverse Problems in Mems, Inverse 2002 Conference- Singapore.

ÖZGEÇMİŞ

Yenal ARSLAN 1990'de Bolu'da doğdu; ilk ve orta öğrenimini aynı şehirde tamamladı; Bolu Atatürk Lisesi'nden mezun olduktan sonra 1998 yılında Selçuk Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'ne girdi; 2002'de mezun oldu. Şuan TEDAŞ Genel Müdürlüğü Bilgi İşlem Merkezinde çalışmaktadır. Halen 2002 yılında girdiği ZKÜ Fen Bilimleri Enstitüsü Makina Eğitimi Anabilim Dalı'nda yüksek lisans programını sürdürmektedir.

ADRES BİLGİLERİ

Adres: TEDAŞ Genel Müdürlüğü
İnönü Bulvarı NO:22
Bahçelievler/ANKARA

Tel: (312) 212 6930
Faks: (312) 213-8873/74
E-posta: yenala@hotmail.com

EK AÇIKLAMALAR A
YAZILIMIN TANITILMASI

A. YAZILIMIN TANITILMASI

A.1 GenPRO YAZILIMININ TANITILMASI

GenpPRO yazılımı Visual Basic.NET programlama dilinde nesne tabanlı olarak yazılmıştır. Programın içerisinde herbir genetik operatör için classlar (selection.vb, crossover.vb, mutation.vb) oluşturulmuştur. Classlar programlar yazılırken okunabilirlik ve geliştirilebilirlik bakımından çok büyük kolaylıklar sağlamaktadır. Program single threading çalışmaktadır. Çünkü Genetik Programlamanın yapısı gereği bütün işlemler birbirinin devamı niteliğindedir. Böyle oluncada multi threading bir program yazmak olanaksızdır. GenPRO yazılımı 3124 satırdır. Genetik Programlamanın doğası gereği GenPRO yazılımı çalışırken güçlü makinalara ihtiyaç duymaktadır. GenPRO için gerekli konfigürasyon şu şekildedir;

Windows xp,Windows 2000 ailesi, windows 2003 server işletim sistemi
Microsoft .NET Framework ver 1.1

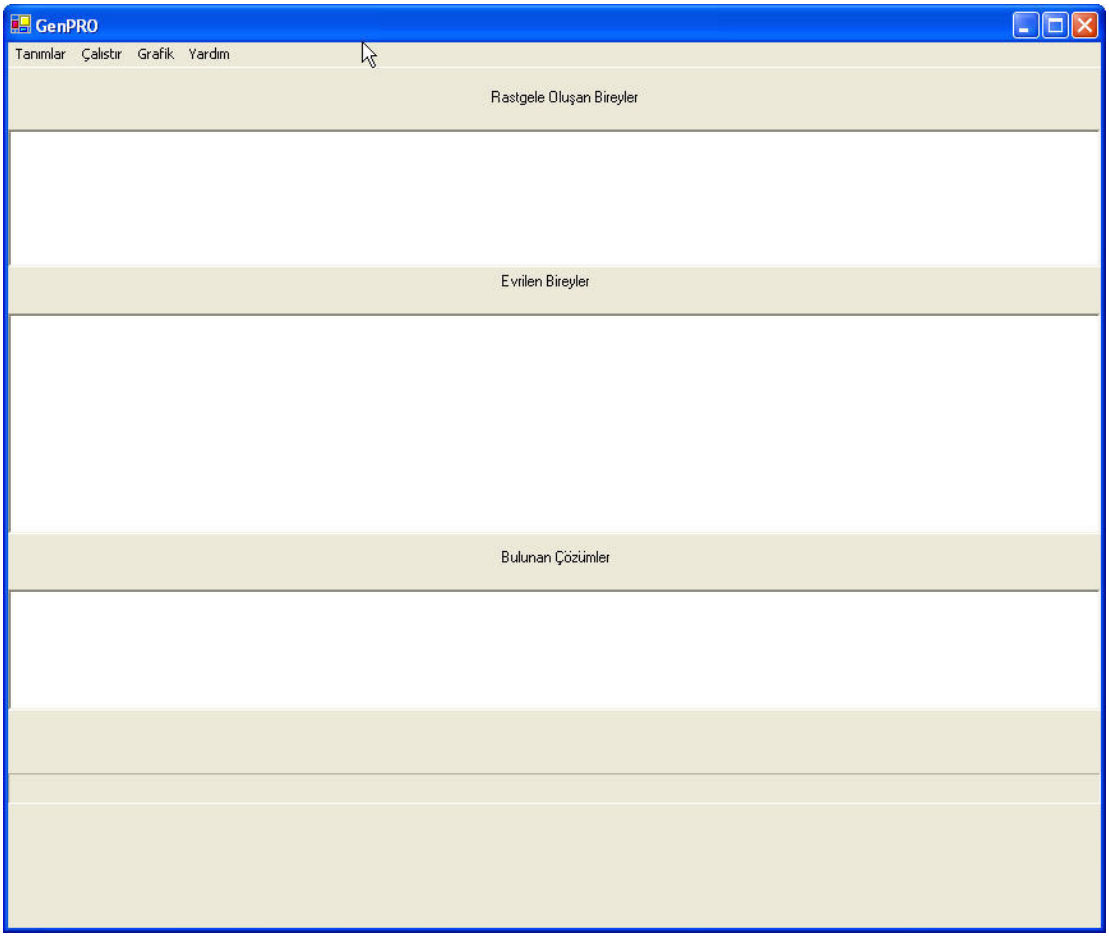
Ragional and language options – Standarts and formats - English (United States)

GenPRO yukarıda sayılan işletim sistemlerinin gerektirdiği minimum donanım özelliklerinde çalışabilir. Genetik Programlamanın doğası gereği doğru sonuçlara ulaşmak için genetik parametreler (Popülasyon sayısı, Maksimum jenerasyon) çok büyük alınabilir. Böyle durumlarda değişkenlerin içeriğinin çok büyük olması, üretilen modellerin içeriğinin genişlemesi, string tipinde işlemlerin bilgisayarda yavaş olması (GenPRO'da bireyler string olarak kaydedilmiştir) gibi nedenlerle, yazılım hem bilgisayardan aşırı derecede hafıza ister hemde sonlanması çok büyük zaman (bir kaç gün) alabilmektedir. Makinaların bunu karşılayacak konfigürasyonda olması gerekmektedir.

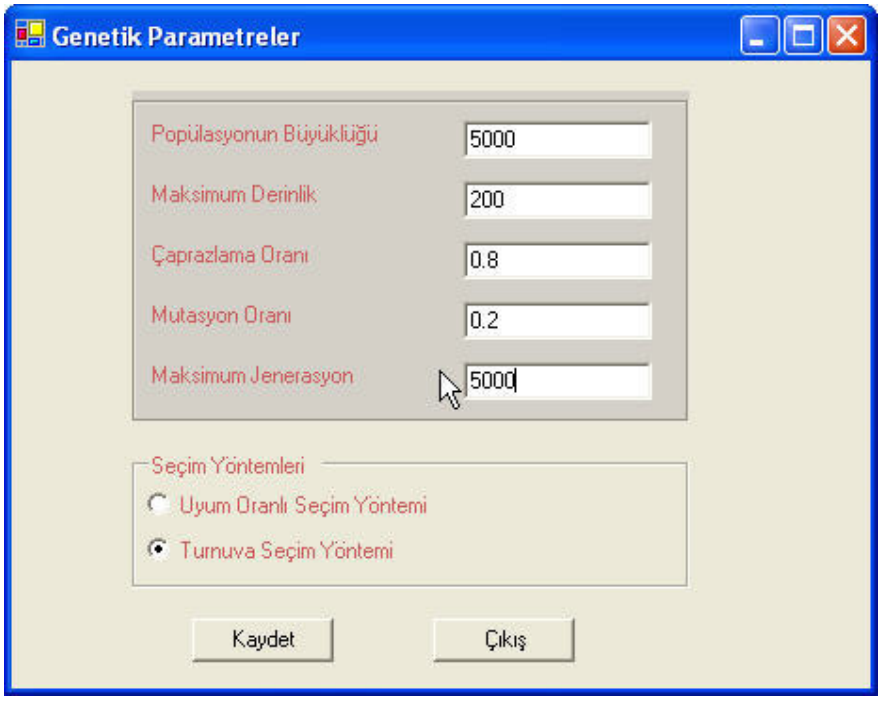
A.2 GenPRO'NUN ÇALIŞMASININ AÇIKLANMASI

A.2.1 GenPRO'nun Genetik Programlama İçin Hazırlanması

GenPRO kurulduktan ve çalışmaya başladıktan sonra GenPRO ana menüsü ile karşılaşılır (Şekil A.1). Yapılması gereken ilk şey "Tanımlar" tabından Şekil A..2'de gösterilen "Genetik Parametreler" sekmesine girip oradan genetik parametreleri girmektir. Genetik parametreleri girerken dikkat edilmesi gereken şey ondalıkı sayıları belirtirken araya nokta (.) konulmasıdır. Eğer nokta yerine virgül (,) konulursa GenPRO bunu algılamayacaktır. Bu problem Türkçemizde ondalıklı ifadelerin virgülle ayrılmasından İngilizcede nokta ile ayrılmasından kaynaklanmaktadır.



Şekil A.1 GenPRO ana menüsü.

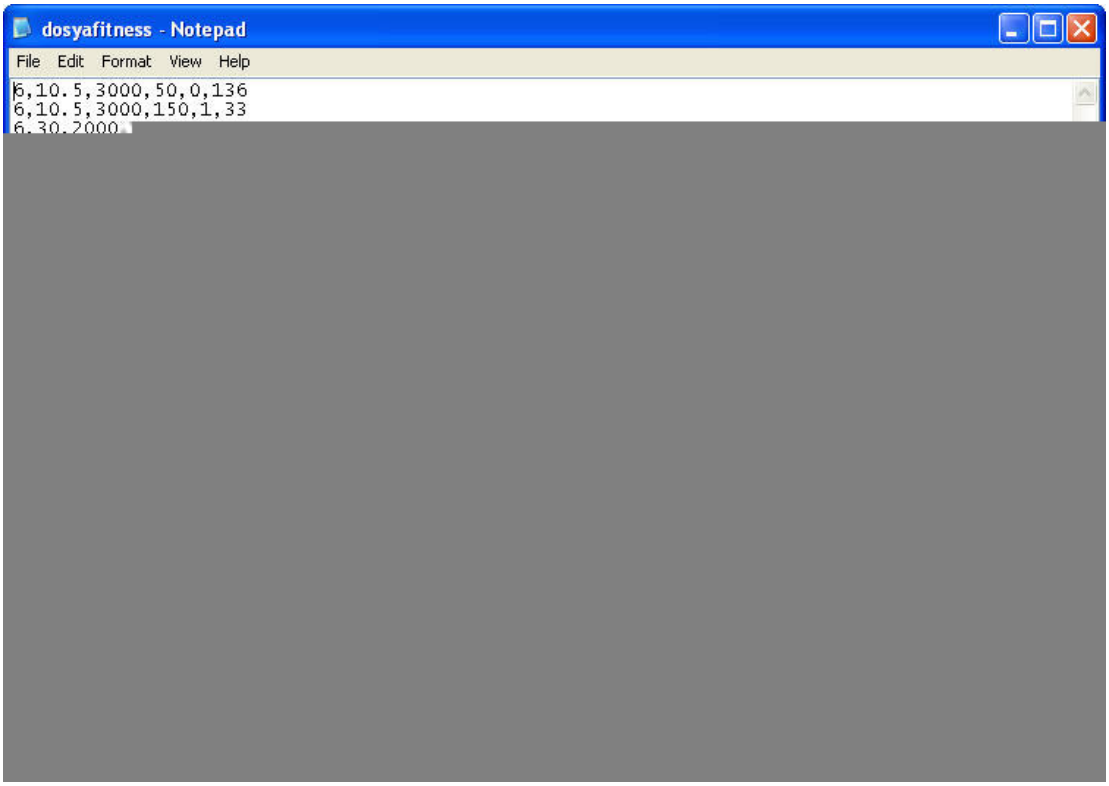


Şekil A.2 Genetik parametrelerin girilmesi.

Seçim yötemi olarak Şekil A.2'de görülen "Uyum Oranlı Seçim" veya "Turnuva Seçim Yöntemlerinden" birini seçmek mümkündür.

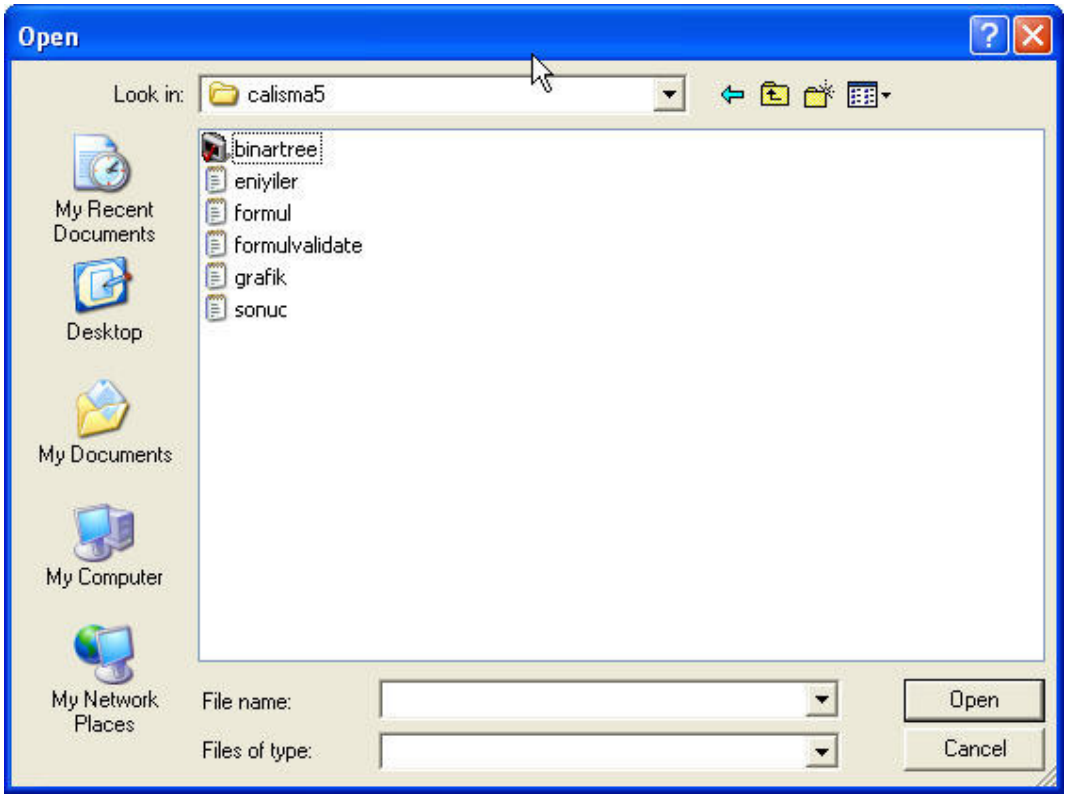
Sonraki aşamada, daha önce hazırlanan ve bir dosya halinde olan öğrenme verisi (*.txt) girilmelidir. Bunun için "Tanımlar" tabından "Öğrenme Verisi" sekmesini tıklayarak (Şekil A.3) okutulacak dosyanın kaç değişkenli (sonuç hariç) olduğunu belirtmek için açılan combobox'ta öğrenme verisinin değişken sayısı seçilir (Şekil A..4). "Dosya Oku" butonu tıklanır sonrasında çıkan dosya penceresinden (Şekil A.5) öğrenme verisi için önceden hazırlanan dosya seçilir. GenPRO on adete kadar değişken içeren öğrenme verilerilerini desteklemektedir.

Öğrenme verisi hazırlanırken dikkat edilecek husus değişkenler birbirinden virgül ile ayrılmalıdır. Sonuç en son değişkenin sağında belirtilmeli ve yine virgül ile ayrılmalıdır. Ondalıklı değişken yada sonuçlar nokta ile belirtilmelidir.



Şekil A.3 Öğrenme verisinin formatı.

Şekil A.4 Değişken sayısının seçilmesi.



Şekil A.5 Öğrenme verisinin seçilmesi.

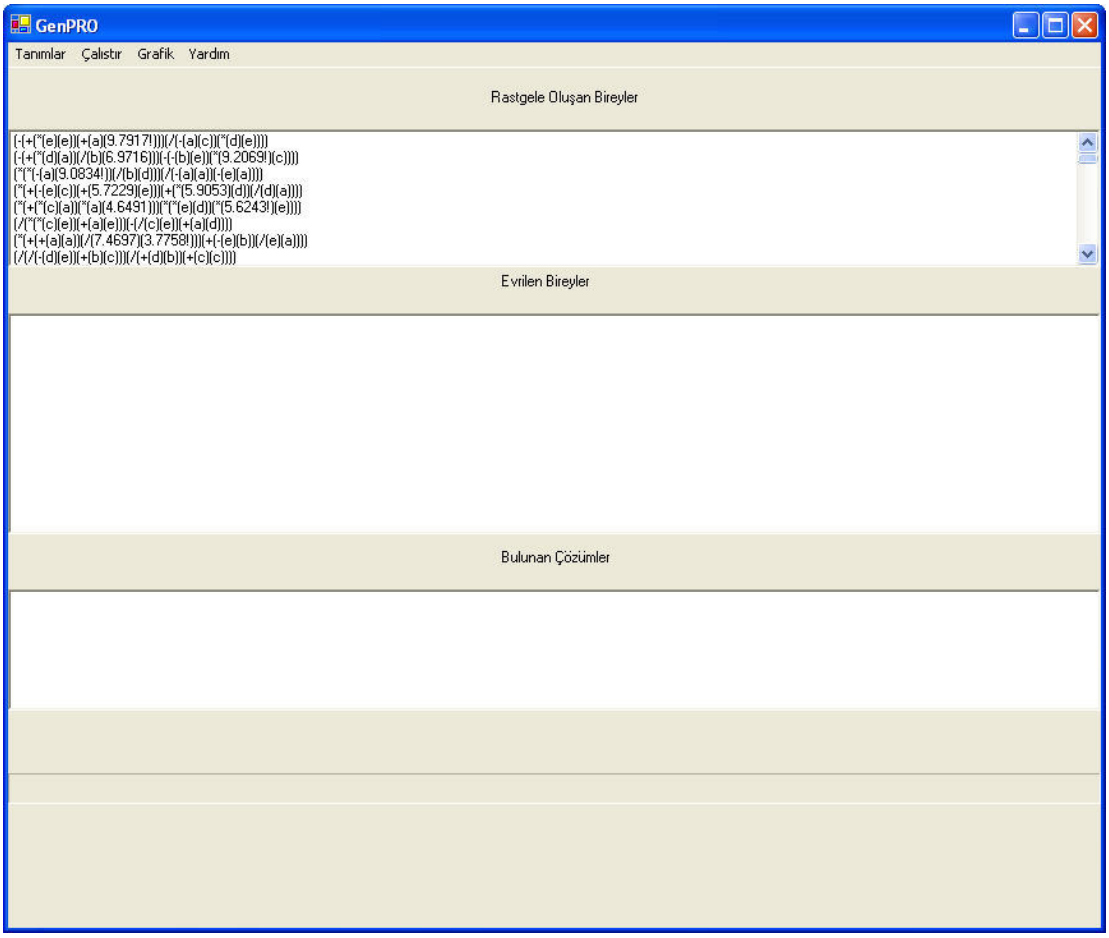
Sonra, doğrulama verisinin girilmesi gerekir. GenPRO doğrulama verisinin girilmesini zorunlu kılmaz bunu kullanıcıya bırakır. Genetik Programlamada kimi zaman doğrulama verisi kullanılmayabilir. Doğrulama verisinin formatı (değişken sayısı) öğrenme verisi ile aynı olmalıdır.

Son olarak; fonksiyon seçimini yapmak gerekir bunun için "Tanımlar" tabından "Fonksiyon seçimi" sekmesini tıklanır. Çıkan pencerede (Şekil A.6) kullanmak istenilen fonksiyonlar işaretlenerek "Kaydet" butonuna basılır. Bu adımla birlikte GenPRO'nun çalışması için ihtiyaç duyduğu nesneler girilmiş olur. Bundan sonra artık GenPRO Genetik Programlama için çalışmaya başlayabilir.

Şekil A.6 Fonksiyonların seçimi.

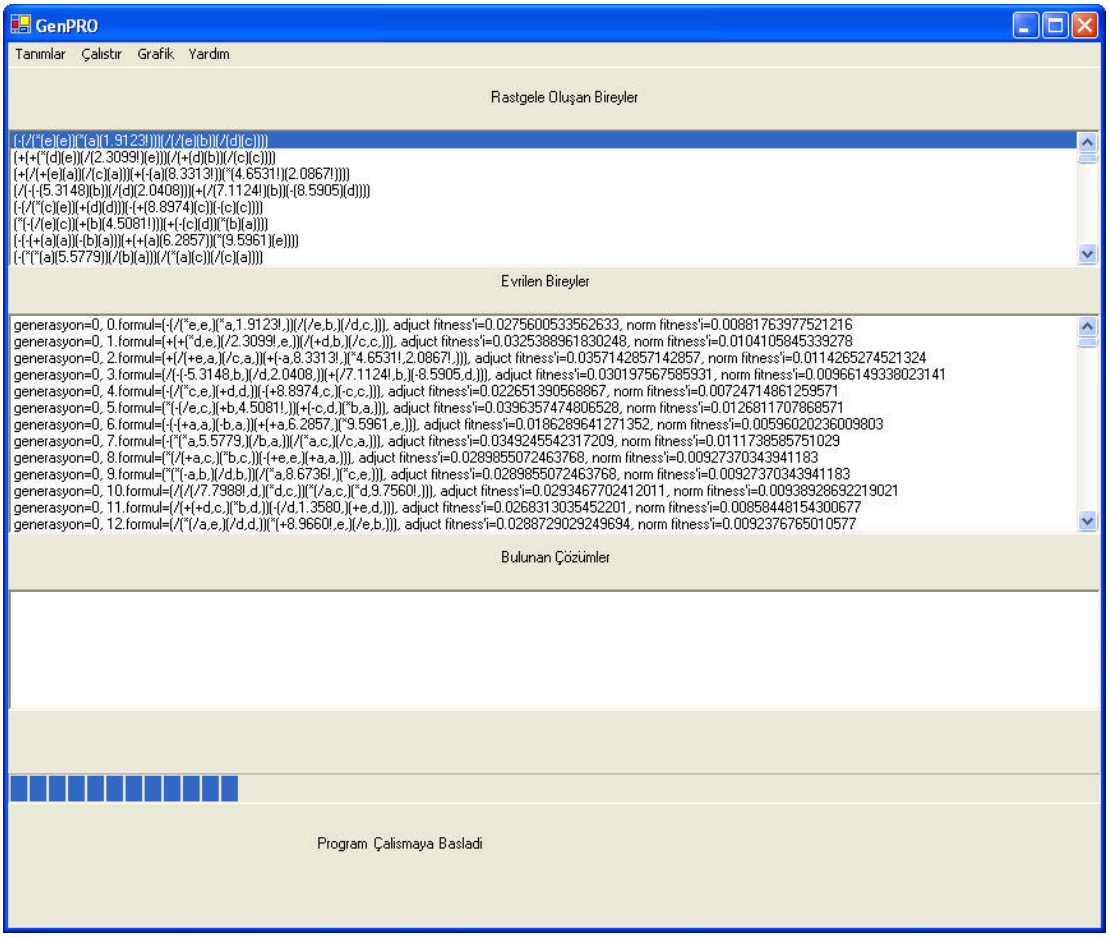
A.2.2 GenPRO'nun Genetik Programlama İçin Çalışmaya Başlaması

Bunun için ilk olarak "Çalıştır" tabından "Ağaç Oluştur" sekmesi tıklanır. Burada GenPRO başlangıçta belirtilen popülasyon büyüklüğü oranında bir zaman gecikmesi ile ikili ağaç yapısında rastgele bireyler oluşturur (Şekil A.7). Bu oluşan bireylerin sayısı popülasyon büyüklüğü kadardır.

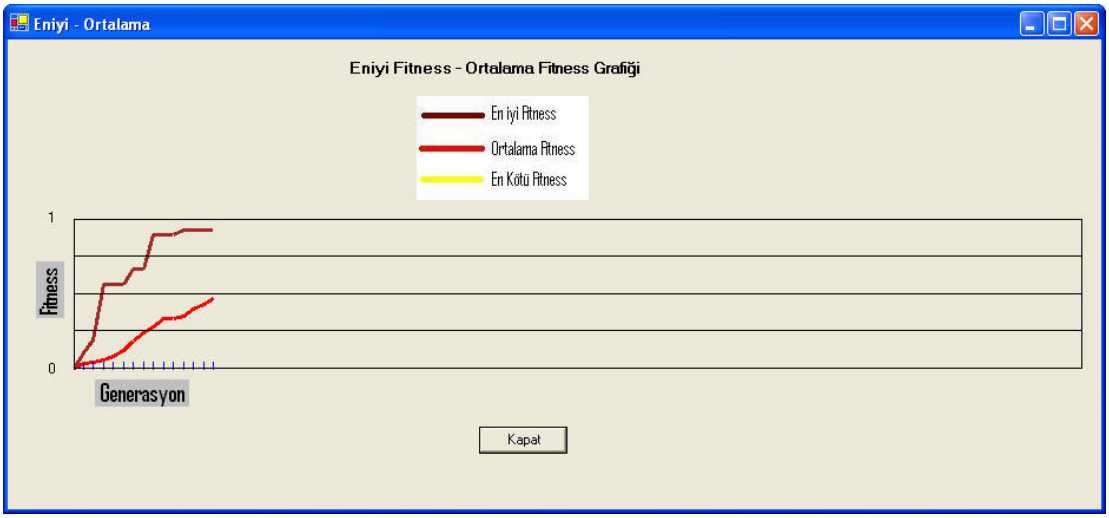


Şekil A.7 Başlangıç popülasyonunun oluşması.

Sonra, "Çalıştır" tabından "Başlat" sekmesi tıklanarak GenPRO'nun olası çözümleri üretmeye başlaması sağlanır (Şekil A.8). Bu sırada "Grafikler" tabından görmek istenilen grafik tıklanarak adım adım programın nasıl ilerlediğinin grafiği görülebilir (Şekil A.9).

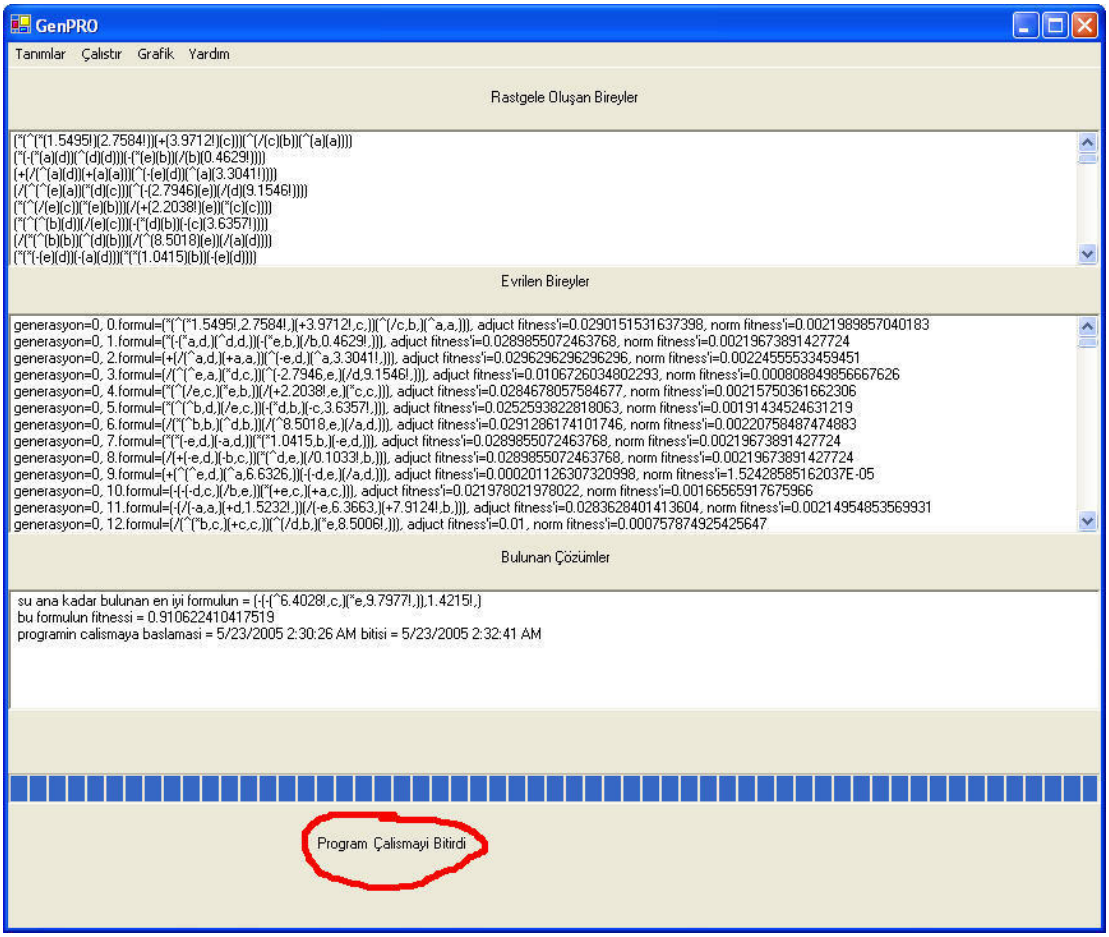


Şekil A.8 Bireylerin evrilmesi.



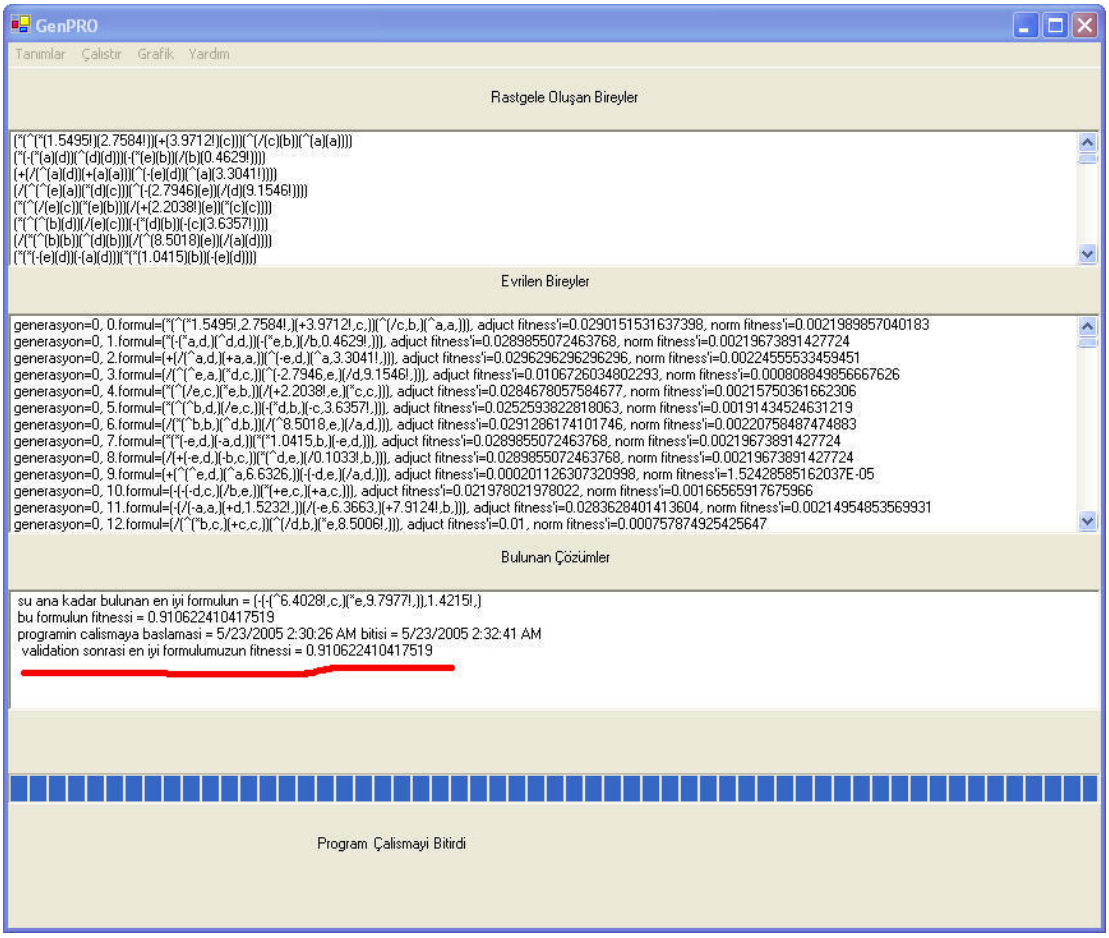
Şekil A.9 Grafiklerin gösterilmesi.

Program çalışmasını bitirdiğinde ana menünün en altında “Program Çalışmayı Bitirdi” diye bir uyarı alınır ve GenPRO’nun sonlandığı anlaşılır (Şekil A.10).



Şekil A.10 GenPRO’nun sonlanması.

GenPRO çalışmasını bitirdikten sonra en son olarak doğrulama yapılır. “Çalıştır” tabından “Doğrula” sekmesi tıklanarak bu gerçekleştirilir (Şekil A.11). GenPRO doğrulamada, doğrulama verilerini o ana kadar bulduğu en iyi sonuca uygulayarak yapar. Eğer doğrulama verileri öğrenme verilerinden daha iyi sonuç elde etti ise o zaman GenPRO buldugu çözümün uyumluluğunu güncelleştirir. Doğrulama verileriyle bulunan çözüm çok kötü sonuçlar elde ederse o zaman çözüm tutarsızdır. GenPRO yeniden çalıştırılır.



Şekil A.11 Doğrulamanın yapılması.

A.2.3 GenPRO'nun Çıktıları

GenPRO çalışmasını bitirdiğinde kullanıcılara text dosyalar halinde sonuçlar sunar. Bunlar aşağıda açıklanmıştır.

eniyiler.txt : Bu dosyaya GenPRO evrilen her jenerasyon sonrasında oluşan uyumluluğu en yüksek bireyi kaydeder.

grafik.txt : Bu dosyaya GenPRO Grafikler tabında yer alan grafiklerin sayısal ifadesini kaydeder.

sonuc.txt : Bu dosyaya GenPRO çalışmasını bitirdikten sonra çalışmayla ilgili istatistikleri kaydeder.

hesaphatalari.txt : Bu dosyaya GenPRO çalışması sırasında ortaya çıkan hesap hatalarını kaydeder. Bunlar sıfıra bölme hataları ve sonsuza yakınsama hataları gibi hatalardır. Bu tür hatalar sonrası GenPRO çalışmasını durdurmaz bu tür bir hataya düştüğü bireyin uyumluluğunu sıfır yaparak o bireyin bir sonraki jenerasyonda seçilme şansını yok etmeye çalışır.

EK AÇIKLAMALAR B
FORMÜLÜN DOĞRULANMASI

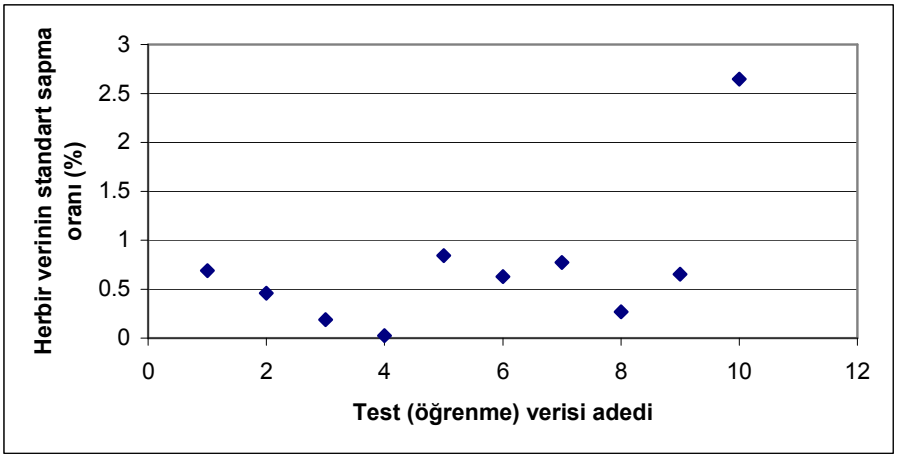
B. FORMÜLÜN DOĞRULANMASI

B.1 TEST (ÖĞRENME) VERİLERİNİN FORMÜLE UYGULANMASI

Yapılan çalışmanın 2.Örneğinin 3.Denemesinin sonucunda verilen tersine problem için GenPRO 0.984 başarı oranına sahip en iyi formülü bulmuştur. Bulunan bu en iyi formüle (B.1.) başlangıçta belirtilen öğrenme verileri uyguladığında ortaya çıkan sonuçların standart sapma dağılımı aşağıda Şekil B. 1'de gösterilmiştir.

F(a,b,c,d) =

(^(+(*(^(/(^(*a,5.1595,)(-(/(+c,b,)(-a,(/(-b,a,)(+4.3419!,(+(*(^(/(^(^(/(^(*a,(-a,(/(-b,c,)(+c,a,)),),)(-(/(+c,b,)(-a,(/(-b,a,)(+4.3419!,(+(*(^(/(^(*d,b,)(-(/(+9.6697,b,)(-a,(/(-b,c,)(+7.6396,a,)),)),c,)),d,)(/4.3419!,5.1595,)),c,),(-d,0.4982!,)),)),)),c,)),d,)(/4.3419!,9.6697,))(-(/(+c,b,)(-a,(/(-b,c,)(+7.6396,a,)),)),c,)),d,)(/4.3419!,9.6697,)),c,),(-d,0.4982!,)),)),)),c,)),d,)(/4.3419!,9.6697,)),c,),(-d,0.4982!,))(/(+0.4982!,b,)(-a,(/5.1595,(^(/(^(*a,5.1595,)(-(/(+c,b,)(-a,(/(-b,(-(-a,(/(-b,(/(+(+(*(^(/(^a,(-(/(+c,b,)(-a,(/(-b,a,)(+4.3419!,(+(*(^(/1.4353,d,)(/4.3419!,9.6697,)),c,),(-d,0.4982!,)),)),)),)),c,)),d,)(/(+c,b,),9.6697,)),c,),(-d,0.4982!,)),(-a,(/(-a,(/(+(*c,4.3419!,),(-d,0.4982!,))(-a,c,)),),c,),)),a,),),(-b,(/(/(+c,b,)(-a,c,)),(-(+7.6396,a,),b,),),),),),b,),),(-b,(/(/(+c,b,)(-a,9.6697,)),(-a,b,),),),),)),c,)),d,)(/(^d,c,)(/(+(/(+4.3419!,(-a,4.3419!,)),a,),(-a,b,)),a,))),),))) (B.1.)

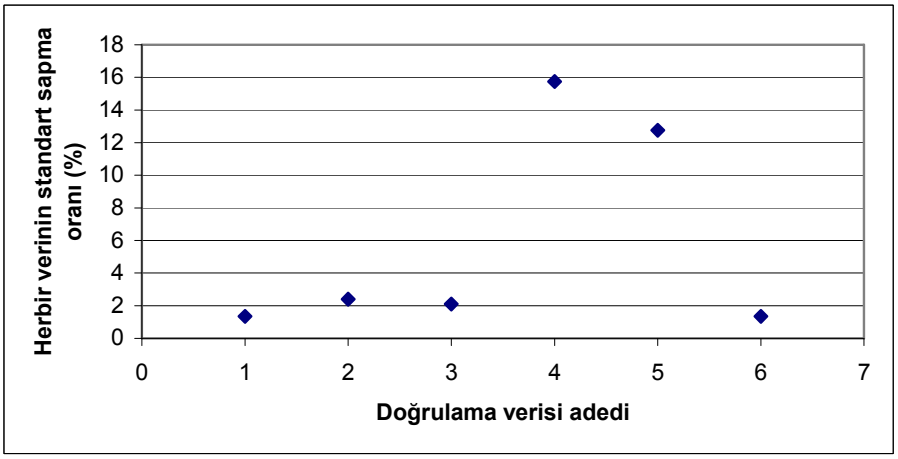


Şekil B.1 Formüle girilen öğrenme verilerinin sonuçlarının standart sapması.

Burada formülün öğrenme verilerine çok büyük oranda uyum sağladığı görülmektedir. Sonuçların standart sapma ortalaması %1'in altında yoğunlaşmaktadır.

B.2 DOĞRULAMA VERİLERİNİN FORMÜLE UYGULANMASI

Yukarıda anılan B.1. formülünün doğrulama verilerine uygulanması sonucu ortaya çıkan değerlerin standart sapması Şekil B. 2. de gösterilmiştir.



Şekil B.2 Formüle girilen doğrulama verilerinin standart sapması.

Grafikten anlaşılacağı üzere formül, doğrulama verileri için test verilerinde olduğu kadar bir uyum sağlamamıştır. Bunun yanında ortalama standart sapma yinede %10'nun altındadır. Bu gibi durumlar iki sebeple açıklanabilir;

Genetik Program başlangıçta verilen öğrenme verilerine aşırı derecede uyum sağlamış (overfitting) olabilir.

Seçilen test verileri ile doğrulama verilerinin, farklı örneklemlerden (girdileri bakımından) seçilmesi nedeniyle olabilir.