ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOKTORA TEZİ

Erkut TEKELİ

BİLİŞİM SİSTEMLERİNDE GÜVENİLİRLİK ANALİZİ

İSTATİSTİK ANABİLİM DALI

ADANA, 2020

ÖZ

DOKTORA TEZİ

BİLİŞİM SİSTEMLERİNDE GÜVENİLİRLİK ANALİZİ

Erkut TEKELİ

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
İSTATİSTİK ANABİLİM DALI

Danışman : Prof. Dr. Güzin YÜKSEL
Yıl: 2020, Sayfa: 118
Jüri: : Prof. Dr. Güzin YÜKSEL
: Prof. Dr. Sadullah SAKALLIOĞLU
: Prof. Dr. Selahattin KAÇIRANLAR
: Prof. Dr. Hamza EROL
: Dr. Öğr. Üyesi Ayça Hatice ATLI

Günümüzde bilgisayar ve yazılım tüm alanlarda giderek daha yaygın bir şekilde kullanılmaktadır. Kritik görevler üstlenen bilgisayar ve yazılım sistemlerinin görevlerini gerçekleştirirken uğrayacağı başarısızlıkların giderek daha ciddi hasarlar oluşturacağı aşikardır. Bu sebeple bir bilişim sisteminin ileride oluşturacağı başarısızlıkları gerçeğe en yakın şekilde tahmin edebilmek önemlidir.

Bu tezin amacı bir yazılım sisteminin güvenilirlik modellerini incelemek ve bir makine öğrenmesi tekniği olan genetik algoritma kullanarak kayıp fonksiyonu minimize edecek modelin parametrelerini belirlemektir.

Bu amacı gerçekleştirmek için iki farklı uygulama yapılmıştır. Karma model ve sağdan sansürlü karma Weibull modelleri için Genetik Algoritma ile model parametreleri optimize edilmiştir. Literatürdeki yöntemler ile karşılaştırılmış ve Genetik Algoritma ile daha performanslı sonuçlar elde edildiği gözlemlenmiştir.

Anahtar Kelimeler: Bilişim Güvenilirliği; Yazılım Güvenilirliği; Genetik Algoritma; Karma Model

ABSTRACT

Ph.D. THESIS

RELIABILITY ANALYSIS IN INFORMATION SYSTEMS

Erkut TEKELİ

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF STATISTICS

Supervisor : Prof. Dr. Güzin YÜKSEL
Year: 2020, Page: 118
Juri : Prof. Dr. Güzin YÜKSEL
: Prof. Dr. Selahattin KAÇIRANLAR
: Prof. Dr. Sadullah SAKALLIOĞLU
: Prof. Dr. Hamza EROL
: Asst. Prof. Dr. Ayça Hatice ATLI

Today, computers and software are used more and more in all fields. It is obvious that the failures of computer and software systems that undertake critical tasks while performing their tasks will cause increasingly serious damage. For this reason, it is important to be able to estimate the future failures of an informatics system.

The purpose of this thesis is to examine the reliability models of a software system and to determine the parameters of the model that will minimize the loss function using a machine learning technique, genetic algorithm.

Two different applications have been made to achieve this goal. For mixed model and right-censored mixture Weibull models, model parameters were optimized by Genetic Algorithm. It was compared with the methods in the literature and it was observed that more performance results were obtained with the Genetic Algorithm.

Keywords: Information Reliability; Software Reliability; Genetic Algorithm; Mixed Model

GENİŞLETİLMİŞ ÖZET

Bilgisayarlar hem güvenlik açısından kritik olan sistemleri hem de günlük yaşamdaki sistemleri izlemek ve kontrol etmek için giderek daha fazla kullanıldığından, yüksek kaliteli ve mümkün olduğunca az hatalı yazılım ürünlerine büyük talep vardır. Güvenilirlik hem yazılım geliştiricileri hem de yazılım kullanıcıları için birincil bir sorundur.

Yazılım güvenilirlik analizi, yeni yazılım ürünlerinin geliştirilmesi ve test edilmesinde giderek daha fazla önem kazanmaktadır. Yeni geliştirilen yazılım, kullanıcının hizmetine sunulmadan önce, geliştirme sırasında ortaya çıkabilecek hatalar için kapsamlı bir şekilde test edilir. Tespit edilen hatalar hemen kaldırılsa da hata ayıklama sırasında yeni hatalar eklenebilir. Hata içeren ve piyasaya sürülen yazılım yüksek bakım maliyetleri doğurur. Hata ayıklama ve test, hata içeriğini azaltır ancak geliştirme maliyetlerini artırır. Bu nedenle, yazılım testini durdurmak için en uygun sürenin belirlenmesine ihtiyaç vardır. Sistem testi sırasında güvenilirlik ölçümü, yazılımın ne zaman piyasaya sürüleceğine karar vermede önemli bir kriterdir. En uygun test süresini belirlemek için, kalan hata sayısı, başarısızlık oranı, güvenilirlik gereksinimleri veya toplam sistem maliyeti gibi diğer bazı kriterler kullanılabilir.

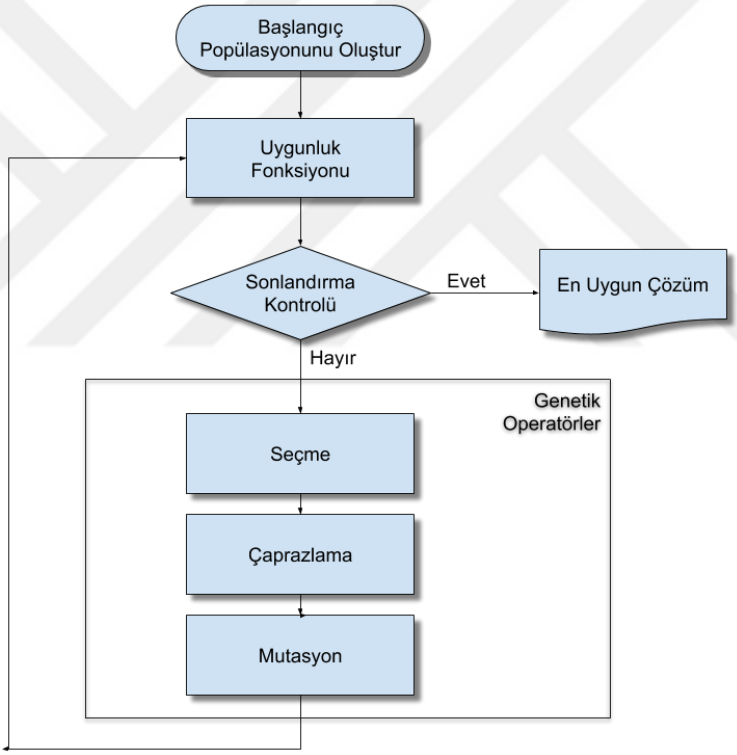
Güvenilirlik fonksiyonunu aşağıdaki eşitlik ile ifade edebiliriz.

$$R(t) = P(T > t) \qquad t \geq 0$$

Burada T, başarısızlık zamanını gösteren rasgele değişkendir.

Yazılım güvenilirliğini kestirebilmek için literatürde kullanılan istatistiksel dağılımlara, çeşitli yöntemlere ve olasılıksal yazılım güvenilirlik modellerinin bazılarına 2. Bölümde detaylı bir şekilde yer verilmiştir.

Bu tez çalışmasında, genetik algoritma yöntemi güvenilirlik tahmini yapmak için kullanılmıştır. Genetik algoritma (GA) yaklaşımı John Holland (1975) tarafından ortaya atılan bir stokastik arama ve optimizasyon tekniğidir. Biyolojik evrimin doğal seleksiyon ve genetik miras özelliklerinden esinlenilerek geliştirilen GA genellikle arama uzayının çok büyük veya analitik işlemlerin çok karmaşık olduğu durumlarda kullanılır. GA'nın çalışma mantığı Şekil 1'de sunulmuştur. Bölüm 3'te genetik algoritma ve bu algoritmanın çalışma mantığı hakkında detaylı bilgi verilmiştir.



Genetik Algoritma Akış Diyagramı (Scrucca, 2013)

Bölüm 4'te, gerçek veri setleri üzerinde yazılım güvenilirliği analizi için uygulama yapılmıştır. Bu uygulamada eğri uydurma model grubundan hata sayısı modelleme yöntemi kullanılmıştır. Literatürde sıkça kullanılan 4 farklı veri seti için

yazılımda zaman içinde bulunan hata sayıları, karma model kullanılarak tahmin edilmeye çalışılmıştır. Karma modeller, genetik algoritma kullanılarak en düşük HKO değerini verecek şekilde seçilmeye çalışılmıştır. Ayrıca, karma modellerde hangi dağılımların kullanılacağı da GA tarafından bir dağılım havuzundan seçilerek, seçilen dağılımların parametreleri optimize edilmiştir. Bu uygulamada çeşitli karma modeller kullanılarak ampirik verinin daha iyi modellendiği gözlemlenmiştir.

Bölüm 5'te, sağdan sansürlü güvenilirlik verisine en uygun iki katlı Weibull karma modelinin parametrelerini tahmin edebilmek için genetik algoritmaya dayanan bir yöntem sunulmuştur. Bu uygulamada eğri uydurma yöntemi ile güvenilirlik verisinin tahmini yapılmaya çalışılmıştır. İki katlı Weibull karma modelinin parametreleri sırasıyla HKO, MHO ve MYHO ölçülerini minimum yapacak şekilde GA kullanılarak optimize edilmiştir. Veri eğitim ve test olarak iki gruba ayrılmıştır. Eğitim veri seti ile çapraz doğrulama kullanılarak model optimize edilmiş ve test veri seti kullanılarak da modelin gelecek veriler için kestirim gücü bulunmuştur. Önerilen yöntem klasik yöntemler ile karşılaştırılmıştır. Gerçek veri seti için eğitim aşamasında EM algoritması GA'ya göre daha uygun model belirlerken, test veri seti için GA modellerinin daha iyi performans sergilediği gözlemlenmiştir. Uygulama bir Monte Carlo simülasyonu ile de desteklenmiştir. Simülasyon çalışmasında da farklı senaryolar kullanılmış ve önerilen GA yönteminin birçok durumda diğer yöntemlerden daha iyi performans gösterdiği gözlemlenmiştir.

TEŞEKKÜR

Her şeyden önce, danışmanım Prof. Dr. Güzin YÜKSEL'e araştırmalarım boyunca yaptığı rehberliği, beni yüreklendirmesi ve faydalı eleştirilerinden dolayı derin şükranlarımı sunuyorum. Bu tez onun desteği ve rehberliği olmadan tamamlanamazdı.

Ayrıca tez izleme komitesi üyeleri Prof. Dr. Sadullah SAKALLIOĞLU'na ve Prof. Dr. Selahattin KAÇIRANLAR'a tezimle ilgili gösterdikleri alakaları, eleştirel önerileri ve incelemeleri için teşekkür ederim.

Son olarak, aileme yıllar içinde gösterdikleri sevgi, anlayış ve özellikle çalışmalarımda kayıtsız şartsız destek oldukları için çok teşekkür ediyorum.

İÇİNDEKİLER **SAYFA**

ÖZ I
ABSTRACT II
GENİŞLETİLMİŞ ÖZET III
TEŞEKKÜR VII
İÇİNDEKİLER VIII
ÇİZELGELER DİZİNİ XII
ŞEKİLLER DİZİNİ XIV
SİMGELER ve KISALTMALAR XVI
1. GİRİŞ 1
2. YAZILIM GÜVENİLİRLİĞİ 7
2.1. Güvenilirlik Ölçümleri 7
2.1.1. Güvenilirlik 7
2.1.2. Güvenilmezlik 7
2.1.3. Başarısızlığa Kadar Geçen Ortalama Süre 8
2.1.4. Başarısızlık Oranı ve Risk Fonksiyonu 8
2.1.5. Bakım Kolaylığı Fonksiyonu 9
2.1.6. Kullanılabilirlik 10
2.2. Güvenilirlik Analizinde Kullanılan Dağılım Fonksiyonları 11
2.2.1. Binom Dağılımı 11
2.2.2. Poisson Dağılımı 11
2.2.3. Üstel Dağılım 12
2.2.4. Normal Dağılım 13
2.2.5. LogNormal Dağılım 14
2.2.6. Weibull Dağılımı 15
2.2.7. Gamma Dağılımı 16
2.2.8. Beta Dağılımı 16
2.2.9. Pareto Dağılımı 17

2.2.10. Rayleigh Dağılımı 17
2.2.11. Loglog Dağılımı 18
2.2.12. İki Parametreli Risk Oranı Dağılımı 19
2.2.13. Üç Parametreli Risk Oranı Dağılımı 19
2.3. Markov Süreçleri 19
2.3.1. Durum Uzayı Diyagramı 20
2.3.2. Geçiş Matrisi 21
2.3.3.Paralel Konumdaki İki Eş Bileşenin BKOS'u 22
2.4. Sayma Süreçleri 24
2.4.1. Poisson Süreçleri 24
2.4.2. Yenileme Süreçleri 25
2.4.3. Homojen Olmayan Poisson Süreçleri 25
2.5. Yazılım Geliştirme Yaşam Çevrimi 27
2.5.1. Yazılım ve Donanım Güvenilirliği Arasındaki Farklar 28
2.5.2. Yazılım Güvenilirliği ve Test Kavramları 29
2.5.3. Yazılım Testleri 31
2.5.3.1. Kara Kutu Testi 31
2.5.3.2. Beyaz Kutu Testi 32
2.5.3.3. Performans Testi 33
2.5.3.4. Sağlamlık ve Stres (Yük) Testi 34
2.5.3.5. Güvenlik Testi 35
2.6. Yazılım Güvenilirliği Modelleri 35
2.6.1. Hata Tohumlama Modelleri 36
2.6.1.1. Mills Hata Tohumlama Modeli 36
2.6.1.2. Cai Modeli 37
2.6.1.3. Hipergeometrik Dağılım Modeli 38
2.6.2. Başarısızlık Oranı Modelleri 38
2.6.2.1. Jelinski-Morenda Modeli 39
2.6.2.2. Schick-Wolverton Modeli 40

2.6.2.3. Jelinski-Morenda Geometrik Modeli 41
2.6.2.4. Düzenlenmiş Schick-Wolverton Modeli 42
2.6.2.5. Goel-Okumoto Kusurlu Hata Ayıklama Modeli 42
2.6.3. Eğri Uydurma Modelleri 43
2.6.3.1. Hata Sayısının Tahmini Modelleri 43
2.6.3.2. Yazılım Karmaşıklığının Tahmini Modelleri 43
2.6.3.3. Başarısızlık Oranının Tahmini Modelleri 44
2.6.4. Güvenilirlik Büyütme Modelleri 45
2.6.4.1. Coutinho Modeli 45
2.6.4.2. Wall ve Ferguson Modeli 45
2.6.5. Markov Modelleri 45
2.6.5.1. Kusurlu Hata Ayıklama ile Markov Modeli 46
2.6.5.2. Littlewood Markov Modeli 46
2.6.5.3. Yazılım Güvenlik Modeli 47
2.6.6. Zaman Serisi Modelleri 48
2.6.7. Homojen Olmayan Poisson Süreci Modelleri 50
2.7. Yazılım Güvenilirlik Analizinde Yeni Yaklaşımlar 51
3. GENETİK ALGORİTMALAR 55
3.1. Problemin Temsili: Kromozomların Kodlanması 54
3.2. Başlangıç Popülasyonu 56
3.3. Uygunluk Fonksiyonu ve Değerlendirme 57
3.4. Genetik Operatörler 58
3.4.1. Seçim operatörü 59
3.4.1.1. Rulet Tekerleği Seçimi 59
3.4.1.2. Turnuva Seçimi 60
3.4.2. Elitizm 60
3.4.3. Çaprazlama Operatörü 61
3.4.3.1. Tek Noktalı Çaprazlama 61
3.4.3.2. İki Noktalı Çaprazlama 62

3.4.3.3. Düzgün Çaprazlama 63
3.4.4. Mutasyon Operatörü 64
3.4.5. Sonlandırma Kriteri 65
3.4.6. GA Paremetreleri 66
4. KARMA MODELDE GA KULLANIMI 67
4.1. Çalışmada Kullanılan Veri Setleri 68
4.2. Model Oluşturulurken Kullanılan Dağılımlar 69
4.3. Bulgular 70
4.3.1. GA ile Dağılım Parametrelerin Belirlenmesi 70
4.3.2. İki Dağılımlı Karma Modelde GA Kullanımı 75
4.3.3. Üç Dağılımlı Karma Modelde GA Kullanımı 78
5. SAĞDAN SANSÜRLÜ VERİLERDE GA KULLANIMI 81
5.1. Giriş 81
5.2. İki Katlı Weibull Dağılımı 83
5.3. Önerilen Yöntem ve Uygulama 87
5.4. Simülasyon Çalışması 91
6. SONUÇLAR ve ÖNERİLER 97
KAYNAKÇA 99
ÖZGEÇMİŞ 107
EKLER 109

ÇİZELGELER DİZİNİ SAYFA

Çizelge 2.1 Yazılım ve donanım güvenilirliklerinin karşılaştırılması 29
Çizelge 2.2 Bazı HOPS yazılım güvenilirlik modelleri 52
Çizelge 3.1 İkili kodlama 56
Çizelge 3.2 Her kromozomu 5 bitle kodlanan rasgele bir başlangıç popülasyonu 57
Çizelge 4.1 Kullanılan veri setlerinin özellikleri 69
Çizelge 4.2 Uygulamada kullanılan dağılımlar 69
Çizelge 4.3 Her bir dağılım için bulunan en düşük HKO değerleri 70
Çizelge 4.4 GA ile iki dağılımlı karma model kullanılarak bulunan sonuçlar . 76
Çizelge 4.5 GA ile üç dağılımlı karma model kullanılarak bulunan sonuçlar.. 78
Çizelge 5.1 Aircraft uçağı ön cam arıza verileri 87
Çizelge 5.2 GA Parametreleri 89
Çizelge 5.3 GA yöntemi ile literatürdeki diğer yöntemlerin karşılaştırılması.. 90

ŞEKİLLER DİZİNİ **SAYFA**

Şekil 1.1 Çalışma zamanının bir fonksiyonu olarak donanım başarısızlık oranı 2
Şekil 1.2 Çalışma zamanının fonksiyonu olarak yazılım başarısızlık oranı ... 4
Şekil 2.1 Paralel konumlanmış iki eş bileşenden oluşan sistem 20
Şekil 2.2 Paralel konumlanmış iki eş bileşenin durum uzayı diyagramı 21
Şekil 2.3 Jelinski-Morenda modeli risk fonksiyonu grafiği 40
Şekil 3.1 Bir rulet tekerleği örneği 60
Şekil 3.2 Tek noktalı çaprazlama 62
Şekil 3.3 İki noktalı çaprazlama 63
Şekil 3.4 Düzgün çaprazlama 63
Şekil 3.5 Mutasyon operatörü 65
Şekil 4.1 GA'da kullanılan kromozom yapısı 68
Şekil 4.2 (a) Gumbel, (b) Gompertz, (c) Genelleştirilmiş Üstel, (d) Genelleştirilmiş Weibull, (e) Üstel Lojistik, (f) Log Gama dağılımları kullanarak veri seti-1 için algoritma tarafından yapılan en iyi tahminler 71
Şekil 4.3 (a) Gumbel, (b) Gompertz, (c) Genelleştirilmiş Üstel, (d) Genelleştirilmiş Weibull, (e) Üstel Lojistik, (f) Log Gama dağılımları kullanarak veri seti-2 için algoritma tarafından yapılan en iyi tahminler 72
Şekil 4.4 (a) Gumbel, (b) Gompertz, (c) Genelleştirilmiş Üstel, (d) Genelleştirilmiş Weibull, (e) Üstel Lojistik, (f) Log Gama dağılımları kullanarak veri seti-3 için algoritma tarafından yapılan en iyi tahminler 73
Şekil 4.5 a) Gumbel, (b) Gompertz, (c) Genelleştirilmiş Üstel, (d) Genelleştirilmiş Weibull, (e) Üstel Lojistik, (f) Log Gama

dağılımları kullanarak veri seti-4 için algoritma tarafından yapılan en iyi tahminler 74
Şekil 4.6 İki dağılımlı karma modellemede, (a) veri seti-1, (b) veri seti-2, (c) veri seti-3, (d) veri seti-4 için algoritma tarafından yapılan en iyi tahminler 77
Şekil 4.7 Üç dağılımlı karma modellemede, (a) veri seti-1, (b) veri seti-2, (c) veri seti-3, (d) veri seti-4 için algoritma tarafından yapılan en iyi tahminler 79
Şekil 5.1 Karma oranına göre yöntemlerin THKO değerlerinin karşılaştırılması 93
Şekil 5.2 Örneklem büyüklüğüne göre yöntemlerin THKO değerlerinin karşılaştırılması 94
Şekil 5.3 Sansür oranına göre yöntemlerin THKO değerlerinin karşılaştırılması 95

SİMGELER ve KISALTMALAR

ARIMA	: Otoregresif Entegre Hareketli Ortalama
ATM	: Automatic Teller Machine
BAOS	: Başarısızlıklar Arasındaki Ortalama Süre
BKOS	: Başarısızlığa Kadar Geçen Ortalama Süre
BLUE	: En İyi Doğrusal Yansız Tahmin
EM	: Beklenti-Makimizasyon
GA	: Genetik Algoritma
GA-HKO	: Uygunluk Fonksiyonu olarak HKO kullanan GA yöntemi
GA-MHO	: Uygunluk Fonksiyonu olarak MHO kullanan GA yöntemi
GA-MYHO	: Uygunluk Fonksiyonu olarak MYHO kullanan GA yöntemi
HKO	: Hata Kareler Ortalaması
HOPS	: Homojen Olmayan Poisson Süreci
J-M	: Jelinski-Morenda
kdf	: Kümülatif Dağılım Fonksiyonu
MHO	: Mutlak Hata Ortalaması
MLE	: Maksimum Olabilirlik Tahmini
MYHO	: Mutlak Yüzde Hata Ortalaması
OKOS	: Onarıma Kadar Geçen Ortalama Süre
oyf	: Olasılık Yoğunluk Fonksiyonu
PSO	: Parçacık Sürü Optimizasyonu
SEM	: Stokastik Beklenti-Maksimizasyon
S-W	: Schick-Wolverton
THKO	: Tahmin Edilen HKO
TYAN	: Tahmin Edilen Yanlılık
WPP	: Weibull Olasılık Kâğıdı
YZ	: Yapay Zekâ

1. GİRİŞ

Güvenilirlik nedir? Sözlükler ve ansiklopediler bu kelimenin çeşitli yorumlarına sahiptir ve onu iyilik, güzellik, dürüstlük gibi soyut kavramlar kategorisinde gösterir. Soyut kavramlar, farklı insanlar için farklı şeyler ifade edebilir ve tanımlanması çok zordur. Bir insana güzel görünen bir şey, diğerine tuhaf görünebilir. Tanımlanması zor olan kavramların ölçülmesi daha da zorlaşır.

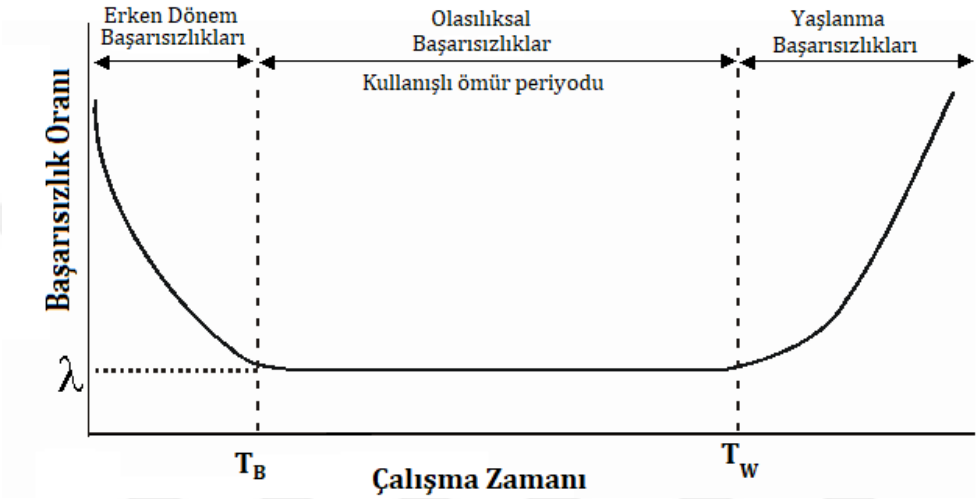
Bununla birlikte, istatistikte ve mühendislikte güvenilirliğin tam bir anlamı vardır. Bu kavram sadece tanımlanmaz, aynı zamanda bir mühendislik ekipmanı parçası olarak da kabul edilebilir. Bu nedenle, mühendisler için güvenilirlik soyut bir kavram olmaktan uzaktır. Aksine, güvenilirlik çok katı bir gerçektir. Bir ekipmanın performansı ile aynı seviyede yer alır ve hatta çoğu zaman performanstan daha da önemlidir.

En basit ve en genel haliyle, güvenilirlik bir başarı olasılığıdır (Bazovsky, 2004). Knight ve ark.'nın (1955) yaygın olarak kabul edilen tanımları ise şöyledir: "Güvenilirlik, bir cihazın karşılaşılan çalışma koşulları altında amaçlanan süre boyunca amacını yeterli bir şekilde gerçekleştirme olasılığıdır". Bu tanım, güvenilirliğin, cihazın belirli bir süre boyunca gerektiği gibi çalışma (başarısızlık ile karşılaşmama) olasılığı olduğunu ima eder. Böyle bir olasılığa, hayatta kalma olasılığı da denir. Çoğu durumda, bir cihazın işlevini gerçekleştirme olasılığı ilk başta bilinmemektedir.

Olasılık bağlamında, tatmin edici performans doğrudan başarısızlık veya arıza kavramlarıyla bağlantılıdır. Cihaz çalışırken tatmin edici şekilde çalışabilir veya başarısızlığa uğrayabilir, arızalanabilir. Neyin "başarısızlık" veya "arıza" olarak kabul edildiğini belirlemek daha basit olabilir ve tatmin edici performans, bir başarısızlık veya arıza durumunda olmayan diğer her türlü çalışma koşuludur da denilebilir.

Mekanik sitemlerde başarısızlık oranının (olasılığı) zamana göre üç farklı periyot oluşturduğu söylenebilir (Bazovsky, 2004). Bunlar Şekil 1.1'de de görüldüğü

gibi üretim hatalarının yoğunlukla başarısızlığa sebep olduğu erken dönem periyodu, başarısızlığın küçük sabit bir olasılıkla gerçekleştiği kullanışlı ömür periyodu, cihazın aşınması sebebiyle başarısızlık oranının yükseldiği yaşlanma periyodudur.



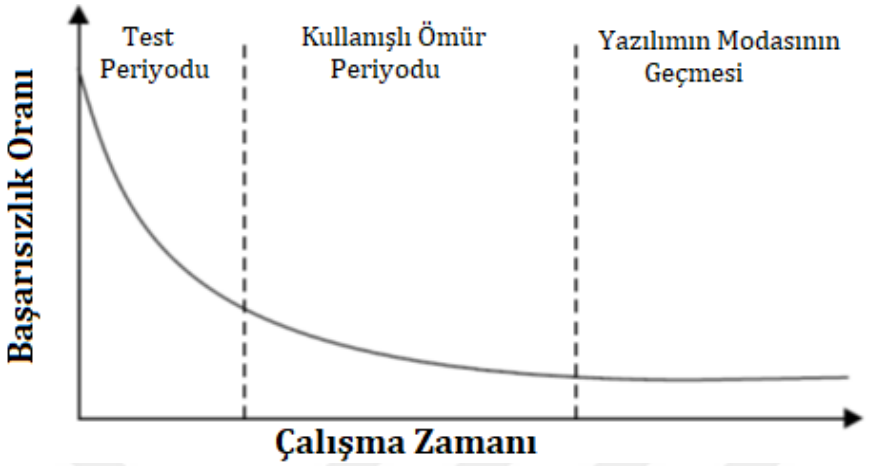
Şekil 1.1. Çalışma zamanının bir fonksiyonu olarak "Donanım" başarısızlık oranı (Bazovsky, 2004)

Bugün, dünyadaki hemen hemen herkes bilgisayar sistemlerinden doğrudan ya da dolaylı olarak etkilenmektedir. Bilgisayarlar, sağlık hizmetleri, hava trafik kontrolü, nükleer reaktörler, gerçek zamanlı sensor ağları, endüstriyel süreç kontrolü, güvenlik gibi milyonlarca insanı etkileyen çeşitli alanlarda kullanılmaktadır. Bir bilgisayar sistemi donanım ve yazılım olmak üzere iki ana bileşenden oluşur. Donanım güvenilirliği konusunda birçok araştırma yapılmıştır. Fakat yazılımların karmaşık uygulamalardaki artan önemi, çalışmaların odak noktasını sistem yazılımı güvenilirliği ve maliyet analizi konularına yöneldiğini göstermektedir. Yazılım güvenilirliğinde, yıpranma veya yaşlanma olmadığı için donanım güvenilirliğinden farklıdır. Yazılım içindeki kusurlar, bağımlı sistemde bir başarısızlığa neden olmadıkça, yazılımın kendisi başarısız olmaz.

Lyu (1996)'ya göre, yazılım güvenilirliği şu şekilde tanımlanır: yazılımın belirli bir ortamda belirli bir süre boyunca hatasız çalışması olasılığıdır. Yazılım güvenilirliği olasılıklı bir fonksiyon olarak tanımlansa ve zaman kavramıyla birlikte gelse de, geleneksel donanım güvenilirliğinden farklı olarak, yazılım güvenilirliğinin zamanın doğrudan bir fonksiyonu olmadığını unutmamalıyız. Elektronik ve mekanik parçalar eskiyebilir ve zamanla, kullanıldıkça aşınabilir. Ancak yazılım yaşam döngüsü boyunca paslanmayacak veya aşınmayacaktır. Yazılım, kasıtlı olarak değiştirilmedikçe veya sürümü yükseltilmedikçe zaman içinde değişmeyecektir.

Yazılım başarısızlıkları, yazılımın karşılaması gereken niteliklerin belirlenmesindeki hatalardan, bu niteliklerin belirsizliğinden, bazı gerekli niteliklerin gözden kaçırılmasından, yanlış yorumlanmasından veya kodlama yaparken dikkatsizlik veya yetersizlikten, yetersiz test, yazılımın yanlış veya beklenmedik kullanımı veya diğer öngörülemeyen sorunlardan kaynaklanabilir (Keiller ve Miller, 1991). Yazılım güvenilirliği ve donanım güvenilirliği için şunlar söylenebilir: Yazılım ve donanımın başarısızlık mekanizmalarında kendilerini farklı kılan temel özellikler vardır. Donanım hataları çoğunlukla fiziksel hatalardır, yazılım hataları ise görselleştirilmesi, sınıflandırılması, tespit edilmesi ve düzeltilmesi daha zor olan tasarım hatalarıdır (Lyu, 1996). Tasarım hataları, belirsiz insan faktörleri ve sağlam bir anlayışa sahip olmadığımız tasarım süreciyle yakından ilgilidir. Bu nedenle, yazılımın kalitesi depolama alanına yüklendikten ve çalışmaya başladıktan sonra değişmeyecektir.

Donanım sistemlerinde başarısızlık oranının zamana göre değişimi Şekil 1.1’de gösterildiği gibiyken yazılım güvenilirliği, donanıma benzer özellikleri göstermez. Aynı eksenler üzerinde yazılım güvenilirliğinin olasılık grafiği Şekil 1.2'de gösterilmektedir (RAC, 1996). Donanım ve yazılım eğrileri arasındaki en önemli fark, son aşamada, yazılımın donanımda olduğu gibi artan bir başarısızlık oranına sahip olmamasıdır. Yazılımda başarısızlık oranı, bulunan ve düzeltilen kusurlar nedeniyle kademeli olarak düşer. Son aşamada kusurların büyük bir çoğunluğu bulunduğu için başarısızlık oranı fazla değişmeyecektir.



Şekil 1.2. Çalışma zamanının bir fonksiyonu olarak "Yazılım" başarısızlık oranı

Bilgisayar uygulamalarının önemi, boyutu ve karmaşıklığı arttıkça, çeşitli işletim ortamlarında bilgisayar sistemlerinin güvenilirliğini ölçmek ve tahmin etmek daha büyük bir ihtiyaç haline gelmiştir. (Pham, 2005). Yazılım tasarımında, hatalar, özellikle de mantıksal hatalar, çözümünü daha zor hale getirir. Genellikle yazılımdaki mantıksal hataları düzeltmek zor değildir, ancak mantık hatalarının teşhis edilmesi birçok nedenden dolayı oldukça zorlayıcıdır.

Mantıksal hatayı bir örnek ile açıklayalım. Evli bir çiftin bankadaki ortak hesabında 1000 ₺ para bulunsun. Erkek, bir ATM cihazından 100 ₺ çekmeye çalışırken aynı zamanda karısının başka bir cihazdan hesaba 300 ₺ yatırdığını düşünelim. Her ikisi de işleme başlarken hesaplarında 1000 ₺ olduğunu görecekler. ATM yazılımının iki işlemin aynı anda yapılmasına izin vermek gibi bir hata barındırdığını varsayalım. Para yatırma işleminin önce bittiğini varsayarsak kadın işlemini bitirdiğinde hesapta 1300 ₺ olacaktır. Fakat kocasının ATM'si hesaptaki tutarın değiştiğini göz önüne almazsa işlem bittiğinde hesapta 1200 ₺ yerine 900 ₺ bulunacaktır.

Genellikle insanlar, yazılımın bir kere doğru çalıştıktan sonra sonsuza kadar doğru çalışacağını düşünürler. Yazılımların neden olduğu bir dizi trajedi ve kaos

bunun yanlış olduğunu kanıtlar. Yazılım hatalarından kaynaklanan bazı önemli olaylar aşağıda verilmiştir.

İngiliz destroyeri (savaş gemisi) Sheffield, radar sisteminin gelen bir füzeyi dost olarak tanımlaması sonucunda batmıştır (Lin, 1985). Yazılımlar ayrıca, bir felaketle sonuçlanabilecek fark edilemeyen küçük hatalara sahip olabilir. 25 Şubat 1991'de Körfez Savaşı sırasında saniyenin 10'da birini 0,000000095 saniye ile kaçıran ve 100 saat boyunca biriken bu hata, Patriot füzesinin bir Scud füzesini yakalayamamasına neden olmuş ve 28 kişi hayatını kaybetmiştir (Arnold, 2020). Bir kez mükemmel çalışan yazılım, çalışılan ortam değişirse de bozulabilir. Ariane 4 roketinin başarısından sonra, Ariane 5'in ilk uçuşu alevler içinde sonuçlanırken, kontrol yazılımındaki tasarım kusurları, yeni roketin daha hızlı yatay sürüklenmesiyle ortaya çıkmıştır (Lions, 1996).

Bu sebeplerden dolayı araştırmamızda daha çok yazılım güvenilirliği analizi üzerinde durulacaktır.

2. YAZILIM GÜVENİLİRLİĞİ

2.1. Güvenilirlik Ölçümleri

2.1.1. Güvenilirlik

Literatürde güvenirlik için çeşitli tanımlar verilmektedir. Pham (2006)'ya göre güvenilirlik, başarı olasılığı veya sistemin belirtilen işlevi belirtilen tasarım limitleri altında gerçekleştirme olasılığıdır. Bir başka ifadeyle güvenilirlik, bir ürünün veya parçanın, tasarım çalışma koşulları altında (sıcaklık, volt, vb.) belirli bir süre boyunca hatasız olarak doğru şekilde çalışması olasılığıdır. Güvenilirlik, sistemin işlevini doğru bir şekilde sağlamadaki başarısının bir ölçüsü olarak kullanılabilir. Güvenilirlik, tüketicilerin ürünün üreticisinden talep ettiği kalite özelliklerinden biridir.

Güvenilirlik aşağıdaki eşitlik ile matematiksel olarak ifade edilir.

$$R(t) = P(T > t) \qquad t \geq 0 \tag{2.1}$$

Burada T, başarısızlık zamanını gösteren rasgele değişkendir.

2.1.2. Güvenilmezlik

Sistemin t anında başarısızlık olasılığı olarak tanımlanır. Güvenilmezlik,

$$F(t) = P(T \leq t) \qquad t \geq 0 \tag{2.2}$$

eşitliği ile ifade edilir. Burada T, başarısızlık zamanını gösteren rasgele değişkendir. Diğer bir değişle F(t) başarısızlıkların kümülatif dağılım fonksiyonudur. f(t), T'nin olasılık yoğunluk fonksiyonu olmak üzere güvenilirliği ifade etmek için,

$$R(t) = \int_{t}^{\infty} f(x)dx = 1 - F(t)$$

veya benzer şekilde f(t) fonksiyonunu güvenilirliğe bağlı ifade etmek için

$$f(t) = -\frac{d}{dt}[R(t)] \tag{2.3}$$

eşitlikleri kullanılır.

2.1.3. Başarısızlığa Kadar Geçen Ortalama Süre

Bir sistemde iki başarısızlık arasında geçmesi beklenen süreye Başarısızlığa Kadar Geçen Ortalama Süre (BKOS) denir ve

$$BKOS = \int_{0}^{\infty} tf(t)dt$$

eşitliği ile ifade edilir. Bu ifadede (2.3) nolu eşitliği yerine koyarsak,

$$BKOS = -\int_{0}^{\infty} td[R(t)] = \int_{0}^{\infty} R(t)dt$$

eşitliği elde edilir.

2.1.4. Başarısızlık Oranı ve Risk Fonksiyonu

Başarısızlık oranı, birim zamandaki başarısızlık olarak tanımlanır. Bir sistemin t anında çalışıyorken $(t, t + \Delta t]$ zaman aralığında başarısız olması olasılığını Rausand ve Hoyland (2004) şu şekilde vermiştir.

$$P(t < T \leq t + \Delta t | t > T) = \frac{P(t < T \leq t + \Delta t)}{P(t > T)}$$

Bu eşitlikte (2.1) ve (2.2) nolu eşitlikler yerine konularak

$$P(t < T \leq t + \Delta t | t > T) = \frac{F(t + \Delta t) - F(t)}{P(t > T)} = \frac{F(t + \Delta t) - F(t)}{R(t)}$$

elde edilir.

Bu olasılığı zaman aralığı olan Δt'ye bölersek başarısızlık oranı elde edilir. Risk fonksiyonu ise zaman aralığı sıfıra yaklaştığında başarı oranının limiti olarak tanımlanır. Yani risk fonksiyonu anlık başarısızlık oranı olarak da düşünülebilir ve aşağıdaki eşitlikle tanımlanır,

$$h(t) = \lim_{\Delta t \to \infty} \frac{F(t + \Delta t) - F(t)}{\Delta t R(t)} = \frac{1}{R(t)} \left[-\frac{d}{dt} R(t) \right] = \frac{f(t)}{R(t)}$$

Risk fonksiyonu bir sistemin ömrü boyunca başarısızlık oranındaki değişimi göstermesi bakımından önemlidir. Örneğin iki sistem tasarımı belirli bir zamanda aynı güvenilirliğe sahip olabilir ama bu zamana gelene kadar başarısızlık oranları farklılık gösterebilir.

2.1.5. Bakım Kolaylığı Fonksiyonu

Bir sistem tatminkâr bir şekilde sonuç vermediğinde, arızayı bulmak ve düzeltmek için onarım yapılır. Sistem, bir ayarlama yapılarak veya bir bileşeni değiştirilerek tekrar çalışır hale getirilir.

Bakım kolaylığı, belirlenen prosedürlere ve kaynaklara göre bakım yapıldığında, belirli bir süre içinde, başarısız bir sistemin belirli koşullara geri

getirilme olasılığı olarak tanımlanır. Diğer bir deyişle, bakım kolaylığı, belirli bir süre içinde bir sistemdeki bir arızanın yalıtılması ve onarılması olasılığıdır.

T_k, toplam kesinti süresi veya onarım süresi rastgele değişkenini göstersin. Onarım süresi yoğunluk fonksiyonu g(t) olması durumunda, bakım kolaylığı, başarısız sistemin, t süresinde hizmete geri dönme olasılığı olarak tanımlanır.

$$V(t) = P(T_k \leq t) = \int_0^{\infty} g(x)dx$$

eşitliği ile gösterilir. Bakım çalışmalarında sıklıkla kullanılan önemli bir ölçüm, onarıma kadar geçen ortalama süre (OKOS) veya ortalama kesinti süresidir ve

$$OKOS = \int_0^{\infty} tg(x)dx$$

eşitliği ile gösterilir.

2.1.6. Kullanılabilirlik

Kullanılabilirlik, öncelikle onarılabilir sistemler için kullanılan bir başarı ölçüsüdür. Onarılamayan sistemler için, kullanılabilirlik, güvenilirliğe eşittir. Onarılabilen sistemlerde, kullanılabilirlik güvenilirliğe eşit veya daha büyük olacaktır.

Güvenilirlik, tüm görev süresi boyunca sistem başarısını gerektiren bir ölçüdür. Hiçbir arıza veya onarıma izin verilmez. Uzay uçuşları ve uçak uçuşları, arızaların veya onarımların izin verilmediği sistemlerin örnekleridir. Kullanılabilirlik, bir hatanın meydana gelmesi durumunda bir sistemin onarılmasını sağlayan bir ölçümdür.

Onarılabilen bir sistemin kullanılabilirliği, sistemin t zamanında başarılı olması olasılığı olarak tanımlanır ve

$$A(t) = \frac{Çalışma\ Süresi}{Çalışma\ Süresi + Kesinti\ Süresi} = \frac{BKOS}{BKOS + OKOS}$$

eşitliği ile verilir.

2.2. Güvenilirlik Analizinde Kullanılan Dağılım Fonksiyonları

2.2.1. Binom Dağılımı

Binom dağılımı, güvenilirlik ve kalite kontrolünde en yaygın kullanılan kesikli dağılımlardan biridir. Güvenilirlik analizinde, örneğin bir olayın başarılı veya başarısız olduğu durumlarda kullanılabilir.

Binom dağılımının olasılık yoğunluk fonksiyonu (oyf) aşağıdaki eşitlikte verilmiştir.

$$P(X = x) = \binom{n}{x} p^x (1-p)^{n-x} \qquad x = 0,1,2,3,\dots,n \quad ,$$

Burada n deneme sayısını, x başarılı deneme sayısını ve p tek bir denemenin başarılı olma olasılığını göstermektedir.

Güvenilirlik fonksiyonu R(k) (n denemeden k tanesinin başarılı olması) ise

$$R(k) = \sum_{x=k}^{n} \binom{n}{k} p^x (1-p)^{n-x} \quad ,$$

eşitliği ile elde edilir.

2.2.2. Poisson Dağılımı

Kesikli bir dağılım olan Poisson dağılımı Binom dağılımına benzer bir şekilde kullanılabilse de örneklem büyüklüğünün bilinmediği olaylarda için kullanılır. Poisson dağılımının oyf'u aşağıdaki eşitlikte verilmiştir.

$$P(X = x) = \frac{(\lambda t)^x e^{-\lambda t}}{x!} \qquad x = 0, 1, 2, \ldots \ ,$$

Burada λ başarısızlık oranını, x olay sayısını göstermektedir. P(X=x) ise t süresince x adet başarısız olayın olma olasılığıdır. Poisson dağılımında güvenilirlik fonksiyonu R(k) (k veya daha az başarısızlığın olasılığı) aşağıdaki eşitlik ile verilir.

$$R(k) = \sum_{x=0}^{k} \frac{(\lambda t)^x e^{-\lambda t}}{x!}$$

2.2.3. Üstel Dağılım

Üstel dağılım, sabit bir başarısızlık oranına sahip olduğundan güvenilirlik analizinde önemli bir rol oynar. Üstel dağılım, yazılım sistemlerin ömrünü modellemek için sıklıkla kullanılmıştır. Üstel dağılımın hafızasızlık özelliği vardır. Henüz başarısız olmayan kullanılmış bir bileşenin yeni bir bileşen kadar iyi olduğu durumlara uygundur. Üstel dağılım için oyf ve güvenilirlik fonksiyonu aşağıdaki eşitliklerde verilmiştir.

$$f(t) = \frac{1}{\theta} e^{-t/\theta} = \lambda e^{-\lambda t} \ , \quad t \geq 0$$

$$R(t) = e^{-t/\theta} = e^{-\lambda t} \ , \quad t \geq 0$$

Burada $\theta = 1/\lambda \geq 0$ olup BKOS parametresini belirtir, $\lambda \geq 0$ ise sabit başarısızlık oranını göstermektedir.

Üstel dağılımın risk fonksiyonu ise aşağıda görüldüğü gibi sabittir.

$$h(t) = \frac{f(t)}{R(t)} = \frac{\lambda e^{-\lambda t}}{e^{-\lambda t}} = \lambda$$

Bu dağılımın başarısızlık oranının sabit olması sebebiyle yaygın olarak kullanılır. Üstel dağılım, küvet eğrisinin uzun düz "içsel başarısızlık" kısmı için mükemmel bir modeldir. Bileşen ve sistemler ömürlerinin çoğunu küvet eğrisinin bu bölümünde geçirir.

2.2.4. Normal Dağılım

Normal dağılım, Merkezi Limit Teoremi sayesinde klasik istatistiksel yöntemlerde önemli bir rol oynamaktadır. Güvenilirlik mühendisliğinde, normal dağılım öncelikle ürün duyarlılığı ve dışsal stres ölçümleri için kullanılan bir dağılımdır. Bu iki parametreli dağılım, birçok mekanik sistemde bazı aşınma etkilerinden dolayı başarısızlıkla sonuçlanabilen sistemleri modellemek için kullanılır.

Normal dağılım ortalama etrafında simetriktir ve yayılma varyans ile ölçülür. Normal dağılımın oyf'nu ve kdf'nu aşağıda verilmiştir.

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2} \quad , \quad -\infty < t < \infty$$

$$F(t) = \int_{-\infty}^{t} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx$$

Burada μ ortalamayı, σ ise standart sapmayı göstermektedir. Güvenilirlik fonksiyonu ise;

$$R(t) = \int_{t}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx$$

eşitliği ile hesaplanabilir.

2.2.5. Log Normal Dağılım

Log-normal yaşam dağılımı, birçok arıza verisi türüne ampirik olarak uyan çok esnek bir dağılımdır. Bu dağılım, sürdürülebilirlik mühendisliğindeki uygulamaları ile onarılabilir sistemlerin başarısızlık olasılıklarını ve başarısızlık oranındaki belirsizliği modelleyebilir. Log-normal yoğunluk fonksiyonu;

$$f(t) = \frac{1}{\sigma t\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\ln t-\mu}{\sigma}\right)^2} \quad , \; t \geq 0$$

burada μ ve σ dağılımın parametreleri olup ortalama ve standart sapmayı göstermezler. Log-normal dağılımının kdf'nu,

$$F(t) = \int_0^t \frac{1}{\sigma x\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\ln x-\mu}{\sigma}\right)^2} dx$$

eşitliği ile tanımlar. Burada Z standart normal dağılıma bağlı olarak kdf'nu

$$F(t) = P(T \leq t) = P(\ln T \leq \ln t) = P\left[Z \leq \frac{\ln t - \mu}{\sigma}\right]$$

eşitliği ile bulunur. Böylece güvenilirlik ve risk dağılımlarını,

$$R(t) = P\left[Z > \frac{\ln t - \mu}{\sigma}\right]$$

$$h(t) = \frac{f(t)}{R(t)} = \frac{\Phi\left(\frac{\ln t - \mu}{\sigma}\right)}{\sigma t R(t)}$$

eşitlikleri yardımı ile buluruz. Burada Φ, standart normal dağılımın kdf'dur.

2.2.6. Weibull Dağılımı

Üstel dağılım, hafızasızlık özelliği sayesinde uygulanabilirlikte genellikle sınırlıdır. Weibull dağılımı (Weibull, 1951) üstel dağılımın genelleştirilmesidir ve genellikle yorgunluk ömrünü, elektrik ve mekanik aksamların ömrünü temsil etmek için kullanılır. Weibull dağılımı, son derece esnektir ve bileşen ömürlerini modellemek ve çeşitli mühendislik uygulamaları için uygundur. Weibull dağılımının birçok çeşidi ve özel durumu vardır. Detaylı bilgi için bkz. Murthy ve ark. (2004).

Üç parametreli Weibull dağılımının oyf'nu;

$$f(t) = \frac{\beta(t-\gamma)^{\beta-1}}{\theta^{\beta}} e^{-\left(\frac{t-\gamma}{\theta}\right)^{\beta}} \quad , \qquad t \geq \gamma \geq 0,\ \theta > 0,\ \beta > 0$$

eşitliği ile tanımlanır. Burada θ ve β sırasıyla ölçek ve şekil parametreleridir. γ ise konum parametresi olarak bilinir. $t \geq \gamma$ için Weibull dağılımının güvenilirlik ve risk fonksiyonları;

$$R(t) = e^{-\left(\frac{t-\gamma}{\theta}\right)^{\beta}} \quad , \qquad t \geq \gamma \geq 0,\ \theta > 0,\ \beta > 0$$

$$h(t) = \frac{\beta(t-\gamma)^{\beta-1}}{\theta^{\beta}} \quad , \qquad t \geq \gamma \geq 0,\ \theta > 0,\ \beta > 0$$

eşitlikleri ile tanımlanır.

2.2.7. Gamma Dağılımı

Gama dağılımı, dağılımı çarpık olan bileşenler için bir başarısızlık olasılık fonksiyonu olarak kullanılabilir. Gama dağılımının oyf'nu aşağıdaki eşitlik ile gösterilir;

$$f(t) = \frac{t^{\alpha-1}}{\beta^{\alpha}\Gamma(\alpha)} e^{-\frac{t}{\beta}} \quad , \quad t \geq 0, \quad \alpha, \beta > 0$$

Burada α şekil parametresi ve β ise ölçek parametresidir. Böylece güvenilirlik fonksiyonu;

$$R(t) = \int_{t}^{\alpha} \frac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{-\frac{x}{\beta}} dx$$

şeklinde elde edilir. Eğer α bir tamsayı ise güvenilirlik ve risk fonksiyonları;

$$R(t) = e^{-\frac{t}{\beta}} \sum_{i=0}^{\alpha-1} \frac{\left(\frac{t}{\beta}\right)^{i}}{i!}$$

$$h(t) = \frac{f(t)}{R(t)} = \frac{t^{\alpha-1}}{\beta^{\alpha}\Gamma(\alpha) \sum_{i=0}^{\alpha-1} \frac{\left(\frac{t}{\beta}\right)^{i}}{i!}}$$

eşitlikleri ile tanımlanır.

2.2.8. Beta Dağılımı

İki parametreli Beta dağılımında oyf'nu aşağıdaki eşitlik ile tanımlanır,

$$f(t) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} t^{\alpha}(1-t)^{\beta} \quad , \quad 0 < t < 1, \ \alpha > 0, \ \beta > 0$$

Burada α ve β dağılım parametreleridir. Beta dağılımı birçok güvenilirlik mühendisliği uygulamasında sıklıkla kullanılan bir dağılımdır.

2.2.9. Pareto Dağılımı

Pareto dağılımı başlangıçta bir popülasyondaki geliri modellemek için geliştirilmiştir. Şehir nüfusu büyüklüğü, hisse senedi fiyat dalgalanmaları ve kişisel gelirler gibi olayların çok uzun sağ kuyruklu dağılımları vardır. Pareto dağılımı bu uzun kuyruğu modele uydurmak için uygundur.

Pareto dağılımının oyf'nu ve güvenilirlik fonksiyonu,

$$f(t) = \frac{\alpha k^{\alpha}}{t^{\alpha+1}} \quad , \quad k < t < \infty$$

$$R(t) = \left(\frac{k}{t}\right)^{\alpha}$$

eşitliği ile tanımlanır.

2.2.10. Rayleigh Dağılımı

Rayleigh dağılımı, birçok bozulma işlemi başarısızlık durumlarına uygulanabilen esnek bir yaşam dağılımıdır. Rayleigh dağılımı oyf'nu ve güvenilirlik fonksiyonu aşağıda verilen eşitlik ile tanımlanır.

$$f(t) = \frac{t}{\sigma^2} e^{\left(\frac{-t^2}{2\sigma^2}\right)}$$

$$R(t) = e^{\frac{-\sigma t^2}{2}}$$

2.2.11. Loglog Dağılımı (Vtub-şekilli Risk Oranı Dağılımı)

Vtub şekilli loglog dağılımı, küvet eğrisi şeklinde arıza oranına sahip olan Weibull dağılımına benzemekle birlikte farklılıkları da vardır. Weibull dağılımında, sistemin erken dönemindeki yüksek orandaki başarısızlıklardan sonra, sistemin yararlı ömrü başlar. Kullanışlı ömrü boyunca, sistem sabit bir başarısızlık oranı ile devam eder. Bu sürenin ardından, sistemde aşınmanın başlamasıyla birlikte başarısızlık oranının arttığı bir aşınma süresi gelir. Vtub şekilli loglog dağılımında ise erken dönemden sonra, sistem nispeten düşük bir artış hızında yaşamaya başlar, ancak bu sabit değildir ve daha sonra yaşlanmaya bağlı başarısızlıklarla oran artar (Pham, 2002).

Loglog dağılımının oyf'nu, güvenilirlik fonksiyonu ve risk fonksiyonu sırasıyla aşağıda verilmiştir.

$$f(t) = \alpha \ln a \, t^{\alpha-1} a^{t^\alpha} e^{1-a^{t^\alpha}} \quad , \quad t > 0, \;\; a > 0, \;\; \alpha > 0$$

$$R(t) = e^{1-a^{t^\alpha}}$$

$$h(t) = \alpha \ln(a) \, t^{\alpha-1} a^{t^\alpha}$$

2.2.12. İki parametreli Risk Oranı Dağılımı

Bu dağılım, artan ve azalan risk oranlarına sahip olabilen iki parametreli bir dağılımdır. İki parametreli risk oranı dağılımının oyf'nu, güvenilirlik fonksiyonu ve risk fonksiyonu sırasıyla aşağıda verilmiştir.

$$f(t) = \frac{n\lambda t^{n-1}}{\lambda t^n + 1} e^{-\ln(\lambda t^n + 1)} \qquad n \geq 1, \lambda > 0, t \geq 0$$

$$R(t) = e^{-\ln(\lambda t^n + 1)}$$

$$h(t) = \frac{n\lambda t^{n-1}}{\lambda t^n + 1}$$

Burada n şekil parametresi, λ ölçek parametresidir.

2.2.13. Üç parametreli Risk Oranı Dağılımı

Bu dağılım, artan ve azalan risk oranlarına sahip olabilen üç parametreli bir dağılımdır. Üç parametreli risk oranı dağılımının oyf'u, güvenilirlik fonksiyonu ve risk fonksiyonu sırasıyla aşağıda verilmiştir.

$$f(t) = e^{-[\ln(\lambda t+\alpha)]^{b+1}} \frac{\lambda(b+1)[\ln(\lambda t+\alpha)]^b}{\lambda t+\alpha} \quad , \quad b \geq 0,\ \lambda > 0,\ \alpha \geq 0,\ t \geq 0$$

$$R(t) = e^{-[\ln(\lambda t+\alpha)]^{b+1}}$$

$$h(t) = \frac{\lambda(b+1)[\ln(\lambda t+\alpha)]^b}{\lambda t+\alpha}$$

Burada b şekil parametresi, λ ölçek parametresi ve α ise konum parametresidir.

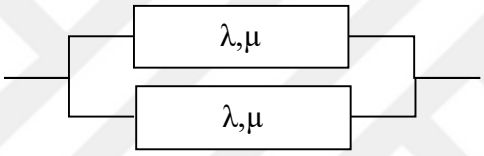
2.3. Markov Süreçleri

Stokastik süreçler, bir sistemde işlemlerin zaman içinde tanımlanması için kullanılır. İki ana stokastik süreç türü vardır: sürekli ve ayrık. Karmaşık sürekli süreç, bir durumdan başka bir duruma geçişi açıklayan bir süreçtir. Burada tartışılacak en basit süreç bir Markov sürecidir. Sürecin mevcut durumu göz önüne alındığında, gelecekteki davranışları geçmişe bağlı değildir.

Markov analizi yöntemi karmaşık sistemlerin analiz edilmesine imkân tanıyan bir yöntemdir. Bu yöntem Markov Zincirleri teorisine dayanır. Sistem bileşenlerinin çalışma, hata veya tamir durumunda olmasına göre analiz yapılır.

2.3.1. Durum Uzayı Diyagramı

Durum uzayı diyagramı bir sistemin güvenilirlik durumlarının grafiksel gösterimidir (Ireson ve ark., 1995). Örnek olarak Şekil 2.1'de başarısızlık oranı λ, tamir oranı μ olan ve paralel olarak konumlanmış iki eş bileşen gösterilmektedir. Bu sistemin durum uzay diyagramını oluşturmaya çalışalım.

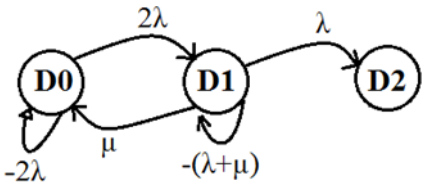


Şekil 2.1. Paralel konumlanmış iki eş bileşenden oluşan sistem

Bu sistemde 3 durum oluşabilir.

Durum 0 (D_0) : İki bileşen de çalışıyor.
Durum 1 (D_1) : Bileşenlerden biri çalışıyorken diğeri hatalı durumda
Durum 2 (D_2) : Her iki bileşen de hatalı durumda

Durum uzayı diyagramında her bir durum bir düğüm noktası ile belirtilir. Bir durumdan diğer bir duruma geçiş ise durumlar arasına çizilen oklar ile gösterilir. Şekil 2.2'de paralel konumlanmış iki eş bileşenin durum uzayı diyagramı görülmektedir.



Şekil 2.2. Paralel konumlanmış iki eş bileşenin durum uzayı diyagramı

Sistem herhangi bir zamanda bu durumlardan yalnızca bir tanesinde bulunabilir. Ayrıca bir durumdan diğer bir duruma geçiş sadece 1 adet bileşenin başarısızlığı ile veya tamiri ile mümkündür.

D_0 durumundan D_1 durumuna geçilirken iki bileşenden herhangi birisinin başarısızlığı söz konusu olduğunda geçiş oranı $\lambda+\lambda = 2\lambda$ 'dır. D_1 durumunda hatalı çalışan bileşen tamir edildiğinde tekrar D_0 durumuna geçilir ki bu durumda geçiş oran μ'dür. Yine D_1 durumunda iken çalışan tek bileşen de başarısızlığa uğrarsa D_2 durumuna geçilir ve geçiş oranı λ'dır.

Bir durumda kararlılığı göstermek için diyagramda durumların üzerine çevrimler çizilir. Bu çevrimler negatif bir sayıyla gösterilir ve değeri ilgili durumdan ayrılan geçiş oranlarının toplamıdır.

2.3.2. Geçiş Matrisi

Durum uzayı diyagramından faydalanılarak oluşturulan matrise geçiş matrisi denir. Matrisin elemanı olan a_{ij} , durum j'den durum i'ye geçiş oranını gösterir. Eğer geçiş yoksa $a_{ij}=0$ dır. Geçiş matrisinin özellikleri şunlardır.

i. Geçiş matrisi kare matristir.
ii. Sütunların toplamı 0'dır.
iii. Matrisin köşegen elemanları durum diyagramındaki çevrimleri gösterir.

İki eş bileşenin paralel olması durumunda geçiş matrisi 3x3 boyutunda olacaktır.

$$T = \begin{bmatrix} -2\lambda & \mu & 0 \\ 2\lambda & -(\lambda+\mu) & 0 \\ 0 & \lambda & 0 \end{bmatrix}$$

2.3.3. Paralel Konumdaki İki Eş Bileşenin BKOS'u

$P_0(t)$, $P_1(t)$ ve $P_2(t)$, t anında sırasıyla D_0, D_1, D_2 durumlarında olma olasılığı olsun. t+Δt anında;

$$P_0(t+\Delta t) = P_0(t) - P_0(t)2\lambda\Delta t + P_1(t)\mu\Delta t \quad (2.2)$$

(2.2) eşitliğinde;

$P_0(t)2\lambda\Delta t$, Δt süresince D_1 durumuna geçme olasılığını,

$P_1(t)\mu\Delta t$ ise Δt süresince D_1 durumundan D_0 durumuna geçilme olasılığını gösterir.

Aynı şekilde $P_1(t+\Delta t)$ ve $P_2(t+\Delta t)$ olasılıkları da aşağıda gösterilmiştir;

$$P_1(t+\Delta t) = P_0(t)2\lambda\Delta t + P_1(t) - P_1(t)(\mu+\lambda)\Delta t \quad (2.3)$$

$$P_2(t+\Delta t) = P_1(t)\lambda\Delta t + P_2(t) \quad (2.4)$$

Sistemin paralel yapısı sebebiyle D_0 ve D_1 durumlarında sistem çalışır vaziyette olduğundan sistemin t anındaki güvenilirliği;

$$R(t) = P_0(t) + P_1(t)$$

Bu durumda BAOS (başarısızlıklar arasındaki ortalama süre);

$$BAOS = \int_0^\infty R(t) = \int_0^\infty P_0(t)dt + \int_0^\infty P_1(t)dt \quad (2.5)$$

denklemi ile hesaplanır.

Δt, 0'a yaklaştığında (2.2), (2.3) ve (2.4) eşitlikleri sırasıyla aşağıdaki hale dönüşür.

$$P_0'(t) = -2\lambda P_0(t) + \mu P_1(t)$$

$$P_1'(t) = 2\lambda P_0(t) - (\lambda + \mu) P_1(t)$$

$$P_2'(t) = \lambda P_1(t)$$

Bu üç eşitliği matris formunda yazarsak;

$$\begin{bmatrix} P_0'(t) \\ P_1'(t) \\ P_2'(t) \end{bmatrix} = \begin{bmatrix} -2\lambda & \mu & 0 \\ 2\lambda & -(\lambda+\mu) & 0 \\ 0 & \lambda & 0 \end{bmatrix} . \begin{bmatrix} P_0(t) \\ P_1(t) \\ P_2(t) \end{bmatrix}$$

Eşitliğin her iki tarafının integralini alıp çözersek;

$$\int_0^\infty P_0(t)dt = \frac{\lambda+\mu}{2\lambda^2} \quad (2.6)$$

$$\int_0^\infty P_1(t)dt = \frac{1}{\lambda} \quad (2.7)$$

sonuçları bulunur. (2.6) ve (2.7) eşitliklerini (2.5) eşitliğinde yerine koyarak BAOS bulunur.

$$BAOS = \int_0^\infty P_0(t)dt + \int_0^\infty P_1(t)dt = \frac{\lambda+\mu}{2\lambda^2} + \frac{1}{\lambda} = \frac{3\lambda+\mu}{2\lambda^2}$$

2.4. Sayma Süreçleri

Ayrık stokastik süreçler arasında güvenilirlik mühendisliğindeki sayım süreçleri, zaman içinde olayların görünüşünü, örneğin arızaları, mükemmel onarımların sayısını, vs. tanımlamak için yaygın olarak kullanılmaktadır. En basit sayma işlemi bir Poisson sürecidir. Poisson süreci, güvenilirlikteki birçok uygulamada özel bir rol oynar (Pham, 2000). Böyle bir uygulamanın klasik bir örneği uranyumun çöküşüdür. Nükleer maddeden radyoaktif parçacıklar, belirli bir yoğunluktaki bir Poisson sürecine göre belirli bir hedefe çarpmaktadır. İyi bilinen bir sayım süreci, sözde yenilenme sürecidir. Bu süreç, birbirinden bağımsız ve aynı şekilde dağıtılmış rastgele değişkenler olan aralıklar olarak tanımlanmaktadır. Güvenilirlik teorisinde, bu tür bir matematiksel model, zaman aralığındaki bir olayın olay sayısını tanımlamak için kullanılır. Bu bölümde aynı zamanda yarı yenileme sürecini ve homojen olmayan Poisson sürecinden bahsediyoruz.

Negatif olmayan, tamsayı değerli bir stokastik süreç olan N(t) 'nin [0, t] zaman aralığındaki olayın toplam sayısını temsil etmesi ve bu iki özelliği karşılaması durumunda bir sayım işlemi olarak adlandırılır.

2.4.1. Poisson Süreçleri

En önemli sayım süreçlerinden biri Poisson sürecidir. N(t) sayma süreci, aşağıdaki koşulları yerine getirirse λ yoğunluğunda bir Poisson süreci olduğu söylenir.

1) Başarısızlık süreci N (t), bağımsız artışlara sahiptir
2) s uzunluğunda herhangi bir zaman aralığında başarısızlıkların sayısı, λs ortalama ile Poisson dağılımına sahiptir, yani,

$$P\{N(t+s)-N(t)=n\}=\frac{e^{-\lambda s}(\lambda s)^n}{n!} \qquad n=0,1,2,\cdots$$

3) Başlangıçta N(0) = 0'dır.

Bu model aynı zamanda, λ başarısızlık oranının t'ye bağlı olmadığını belirten homojen bir Poisson süreci olarak da adlandırılır. Diğer bir deyişle, zaman aralığı (t, t + s] sırasında meydana gelen başarısızlık sayısı, t zamanına değil, sadece zaman aralığının uzunluğuna bağlıdır. Bir sayım sürecinin, ayrık zaman aralıklarındaki olayların sayısının bağımsız olması halinde bağımsız artışlara sahip olduğu söylenir.

2.4.2. Yenileme Süreçleri

Bir yenileme süreci, başarısızlıklar arasındaki sürenin üstel dağılımı takip etmediği, Poisson sürecinin daha genel bir örneğidir. Kolaylık sağlamak için, bir etkinliğin meydana gelmesini bir yenileme, yenileme süresi arasındaki varış zamanı ve yenileme süresini bekleme süresi olarak adlandıracağız.

Başarısızlıklar arasındaki süreler bağımsız ve eş dağılımlara sahip rasgele değişkenler ise, bu sayım süreci bir yenileme işlemi olarak adlandırılır.

2.4.3. Homojen Olmayan Poisson Süreçleri

Homojen olmayan Poisson süreci (HOPS) modeli t zamanına kadar yaşanan başarısızlık sayısını temsil eder. HOPS modelindeki ana sorun, belirli bir zamana kadar beklenen başarısızlık sayısını belirtmek için uygun bir ortalama değer fonksiyonu belirlemek gerekmektedir. Farklı varsayımlar ele alındığında, model ortalama değer fonksiyonunun farklı fonksiyonel formları ile sonuçlanacaktır.

HOPS modeli aşağıdaki varsayımlara dayanmaktadır:

1) Başarısızlık süreci geçmiş başarısızlıklardan bağımsızdır. Yani, (t, t+s) zaman aralığı sırasındaki başarısızlık sayısı, t zamanına ve zaman aralığı s'nin uzunluğuna bağlıdır ve sürecin geçmişine bağlı değildir.
2) Sürecin başarısızlık oranı,

$$P\{(t, t+\Delta t)\ aralığındaki\ başarısızlık\ sayısı = 1\}$$
$$= P\{N(t+\Delta t) - N(t) = 1\} = \lambda(t)\Delta t + o(\Delta t)$$

3) Küçük bir Δt aralığında, birden fazla başarısızlık olasılığı ihmal edilebilir, yani,

$$P\{N(t+\Delta t) - N(t) > 1\} = o(t)$$

4) Başlangıçta $N(0) = 0$

Bu varsayımlara dayanarak, HOPS için (0, t) zaman aralığında tam olarak n adet başarısızlık olasılığı

$$P\{N(t) = n\} = \frac{[m(t)]^n}{n!} e^{-m(t)} \qquad n = 0,1,2,\cdots$$

eşitliği tarafından verilir. Burada, $m(t) = E[N(t)] = \int_0^t \lambda(x)dx$ ve $\lambda(x)$ başarısızlık oranı fonksiyonudur.

(0, t) aralığında hiç başarısızlık olmaması olasılığını belirten güvenilirlik fonksiyonu ise aşağıdaki eşitlik ile elde edilebilir.

$$R(t) = P(N(t) = 0) = e^{-m(t)}$$

Güvenilirlik fonksiyonunu genelleştirerek (t, t+x) aralığında hiç başarısızlık olmaması olasılığını,

$$R(x|t) = P(N(t+x) - N(t) = 0) = e^{-[m(t+x)-m(t)]}$$

eşitliği ile elde edilir.

2.5. Yazılım Geliştirme Yaşam Çevrimi

Askeri savunma sanayi, nükleer reaktörler gibi karmaşık ve kritik işlevleri yerine getiren sistemlerde yazılım giderek daha önemli hale geldiğinden, yazılım kaynaklı arızaların da riskleri ve etkileri çok önemli hale gelmiştir. Artık yazılım geliştirme süreci sırasında oluşan hataları ortadan kaldırarak yazılım güvenilirliğini ve kalitesini artırma ihtiyacı gereklilik haline gelmiştir. Sanayi ve akademik kurumlar bu ihtiyaca, yazılım mühendisliği olarak bilinen teknolojik yöntemler geliştirerek ve gelişim süreci sırasında ve buna paralel olarak yazılım hatalarını tespit etmek için sistematik kontroller uygulayarak cevap vermiştir.

Birçok yazılım kuruluşu için hataların azaltılması en önemli kalite hedefidir. Bununla birlikte, tüketici elektroniği işi farklı bir hedefe yöneliktir: alandaki kusurların sayısını sıfırda tutmak. Elektronik ürünler showroom'dan ayrıldığında varış yerleri bilinmemektedir. Bu nedenle, ciddi bir yazılım arızasının tespiti ve düzeltilmesi, yüz binlerce ürünün geri çağrılmasını gerektirecektir.

Lyu'ya (1996) göre yazılım mühendisliği uygulamalı bir disiplin olma yolunda gelişmektedir. Yazılım kalitesi, yazılım geliştirme süreci, yazılım güvenilirlik modellemesi ve yazılım hata toleransı gibi alanlarda yüzlerce araştırma makalesi yayınlanmıştır.

Bir yazılım sisteminin kalitesini değerlendirmek için son yıllarda çok sayıda analitik model önerilmiş ve incelenmiştir. Her model, geliştirme süreci ve test ortamı hakkında bazı varsayımlar ortaya koymaktadır. Yazlım geliştirme çevresi, yazılım uygulamasına, yaşam döngüsü geliştirme sürecine ve mühendislik tasarım ekibinin yeteneklerine bağlı olarak değişebilir (Malaiya ve Srimani, 1990). Bu nedenle, yazılım geliştiricilerinin ve uygulayıcılarının, herhangi bir yazılım ürününün kalitesi hakkında bilinçli kararlar verebilmek için ilgili tüm modellere aşina olmaları önemlidir.

2.5.1. Yazılım ve Donanım Güvenilirliği Arasındaki Farklar

Donanım güvenilirliği teorisinin gelişimi yazılım güvenilirliğine göre daha uzun bir geçmişe sahiptir. Son yıllarda yazılım tarafında uygulamalarının boyutu ve karmaşıklığı giderek artarken donanım tarafında ise güvenilirlik büyük ölçüde artırılmıştır (Xie, 1991). Donanım güvenilirliği, ürün performansını olumsuz etkileyen sistem arızalarını azaltmak veya ortadan kaldırmak için sistematik olarak çabalayan geniş bir analiz yelpazesini kapsar. Güvenilirlik ayrıca güvenlik ve risk analizini değerlendirmek için temel bir yaklaşım sağlar.

Yazılım güvenilirliği ise, bir yazılım programının performansını olumsuz yönde etkileyen sistem hatalarını azaltmak veya ortadan kaldırmak için sistematik olarak çalışır. Yazılım sistemleri değiştirilmedikçe zamanla bozulmaz. Donanım ve yazılımın güvenilirliği arasında test kavram ve teknikleri arasında birçok fark vardır. Bu nedenle, yazılım ve donanım güvenilirliğinin karşılaştırılması, yazılım güvenilirliği modellemesinin geliştirilmesinde yararlı olacaktır.

Pham (2000), donanım veya yazılım tarafındaki arızaları ayrıntılı olarak göstermiştir. Çizelge 2.1 yazılım ve donanım güvenilirliği arasındaki farkları ve benzerlikleri göstermektedir. Sonuç olarak, geliştirme sürecinde yazılım kalitesi ve güvenilirliğinin yazılıma dahil edilmesi gereklidir.

Çizelge 2.1. Yazılım güvenilirliği ile donanım güvenilirliğinin karşılaştırılması

Yazılım Güvenilirliği	Donanım Güvenilirliği
Program gelişimi dikkate alınmaksızın, başarısızlık oranı istatistiksel olarak artmamaktadır.	Başarısızlık oranı bir küvet eğrisine sahiptir. Erken dönem, yazılım hata ayıklama dönemine benzer.
Yazılım kullanılmazsa asla hata oluşmaz	Sistem kullanılmasa bile malzeme bozulması (metal yorgunluğu) arızalara neden olabilir.
Çoğu güvenilirlik model analitik olarak varsayımlardan türetilmiştir. Önemli konular, modelin geliştirilmesi, model varsayımlarının yorumlanması ve parametrelerin fiziksel anlamıdır.	Başarısızlık verileri bazı dağılımlara uydurulur. Temel dağılımın seçimi, başarısızlık verilerinin ve deneyimlerinin analizine dayanır. Başarısızlık verilerinin analizine vurgu yapılır.
Hatalara yanlış mantık, yanlış ifadeler veya yanlış giriş verileri neden olur. Bu, karmaşık bir donanım sisteminin tasarım hatalarına benzer.	Başarısızlıklar, malzeme bozulması, rastgele arızalar, tasarım hataları, yanlış kullanım ve ortamdan kaynaklanır.
Yazılımın güvenilirliği, test aşamasında sar edilen eforu artırarak ve tespit edilen hataları düzelterek iyileştirilebilir. Güvenilirlik, yeni koddaki sorunların eklenmesi veya hata ayıklanmasıyla sorunların giderilmesi nedeniyle test sırasında sürekli olarak değişme eğilimindedir.	Donanım güvenilirliği, daha iyi tasarım, daha iyi malzeme, yedekli çalışma ve hızlandırılmış ömür testi uygulanarak geliştirilebilir.
Yazılım onarımları yeni bir yazılım parçası oluşturur.	Donanım onarımları sistemi orijinal duruma geri getirir.
Yazılım başarısızlıkları nadiren uyarılardan sonra gelir.	Donanım başarısızlıklarından önce genellikle uyarılar gelir.
Yazılım bileşenleri nadiren standartlaştırılmıştır.	Donanım bileşenleri standartlaştırılabilir.
Yazılım esasen sonsuz test gerektirir.	Donanım genellikle kapsamlı bir şekilde test edilebilir.

2.5.2. Yazılım Güvenilirliği ve Test Kavramları

Yazılım esasen bir girdi kümesini bir çıktı kümesine dönüştüren bir araçtır. Bir ifadeyi değerlendirmek ve sonucu geçici veya kalıcı bir yerde saklamak, hangi ifadenin yürütüleceğine veya hangi girdi/çıktı işlemlerini gerçekleştireceğine karar vermek için bir dizi kodlanmış ifade veya talimat içerir. Bu nedenle, yazılım, girdi alanını çıktı alanına eşleştiren bir f fonksiyonu olarak kabul edilebilir, burada girdi

alanı tüm girdi durumlarının kümesi ve çıktı alanı tüm çıktı durumlarının kümesidir (Goel ve Okumoto, 1985).

Yazılım güvenilirliği, belirli bir ortam koşulunda belirli bir süre boyunca yazılımın başarılı bir şekilde çalışabilme olasılığıdır. Diğer bir deterministik model, yazılım güvenilirliğini, belirli çalışma koşulları altında giriş alanından rastgele seçilen bir giriş durumunun başarılı yürütülmesi olasılığı olarak tanımlar. Başka bir tanım, belirtilen bir ortamda yazılımın belirli bir süre boyunca başarılı yürütülmesi olasılığıdır. Örneğin, ortalama bir kullanıcı için 8 saat boyunca %95 güvenilirliğe sahip bir işletim sistemi, 8 saatlik 100 dönemin 95'inde sorunsuz çalışmalıdır. Yazılım hatası, bir gereksinimle belirlenen bir görevi gerçekleştirememe anlamına gelir. Bir yazılım kusuru (veya bug), program kaynak metnindeki bir hatadır ve bu, program belirli koşullar altında yürütüldüğünde yazılımın başarısız olmasına neden olur. Bu nedenle, yazılım geliştirme sırasında yapılan yanlışlar yazılım hatası üretir.

Örneğin bir taşıt uygulamasında hatalar dört kategoriye ayrılabilir: kritik, yüksek, orta ve düşük:

Kritik: Kanuni olarak zorunlu bir maddeyi etkileyebilir. (Derhal bir değişiklik yapılması gerekir, örneğin fren sistemi.)

Yüksek: Taşıt işletiminin veya alt sisteminin gerekli bir işlevini, örneğin çalışmayan motor, klima, kilitler vb. etkileyebilir (potansiyel müşteri memnuniyeti açısından derhal bir değişiklik yapılması gerekir).

Orta: Taşıtın konfor veya kolaylık özelliğini etkileyebilir, örneğin ses sistemi, seyir kontrolü, mini yol bilgisayarı, far gecikmesi gibi (ilk fırsatta değiştirilmesi lazım).

Düşük: Bu hatanın taşıt veya sistem performansı üzerinde gerçek bir etkisi yoktur (hemen değişiklik yapılması gerekmez).

Genel olarak, bir yazılım başarısızlığının sebebi uygulamaya bağlı olduğu için tartışmaya açık bir alandır. Bir program çöktüğünde, tasarım, özellikler, kodlama veya testteki bir hata nedeniyle açıkça başarısız olduğunu söyleyebiliriz. Fakat sadece programın çöktüğü durumda yazılım başarısızdır demek doğru değildir.

Başarısızlığı, kullanıcının yazılım işlemiyle ilgili gereksinimlerini veya beklentilerini karşılamaması durumu olarak tanımlamak daha doğrudur. İyi tanımlanmış bir dizi başarı kriteri ile başarısızlıkları daha iyi tanımlayabiliriz. Örneğin, programın çalışma hızı veya hesaplamaların doğruluğu yazılımın başarısızlığı için kriter olabilir.

Yazılım testi, bir hatayı bulmak için bir program yürütme işlemidir. İyi bir test olayı, keşfedilmemiş hataları bulma olasılığının yüksek olduğu olaydır. Tüm olası girdilerin test edilmesi milyonlarca yıl gerektireceğinden, tüm hatalar tespit edilip giderilene kadar yazılımı test etmeye devam etmek imkansızdır. Bu nedenle, başarısızlık olasılıkları, girdi alanı adı verilen tüm olası girdi durumlarının bir örneğinin test edilmesinden çıkarılmalıdır. Başka bir deyişle, giriş alanı tüm olası giriş durumları kümesidir. Benzer şekilde, çıkış alanı, belirli bir yazılım ve giriş alanı için tüm olası çıkış durumları bir kümesidir.

Boyutsal problemler nedeniyle büyük bir bilgisayar programını kapsamlı bir şekilde test etmek genellikle çok zordur. Giriş alanı tek bir sınırsız değişkenden oluşuyorsa, programın kapsamlı bir testini sağlamak için sonsuz sayıda giriş senaryosu gerekecektir. Giriş alanı sınırlandırılmış ancak çok sayıda bağımsız değişken içeriyorsa, kapsamlı bir test için gereken girdi olaylarının sayısı, her bir giriş değişkeni için ayrık değerler kabul etse bile imkânsız derecede büyük olacaktır.

2.5.3. Yazılım Testleri

Test uygulayıcıları, test edilen yazılım modülünün iç ayrıntılarını biliyor veya bilmiyor olabilirler. Bu nedenle, test yazılımında beyaz kutu veya kara kutu bakış açısıyla yola çıkılabilir. Kara kutu ve beyaz kutu fikirlerinin yalnızca doğruluk testinde sınırlı değildir.

2.5.3.1. Kara Kutu Testi

Kara kutu yaklaşımı, test verilerinin programın yapısına bakılmaksızın belirtilen fonksiyonel gereksinimlerden türetildiği bir test yöntemidir (Perry, 1990).

Ayrıca veri odaklı, giriş/çıkış odaklı (Myers, 1979) veya gereksinimlere dayalı (Hetzel, 1988) test olarak da adlandırılır. Yalnızca yazılım modülünün işlevselliği önemli olduğu için, kara kutu testi de temel olarak işlevlerin yerine getirilmesi ve giriş/çıkış verilerinin incelenmesi üzerinde durulan bir test yöntemidir (Howden, 1987).

Test uygulayıcıları, test edilen yazılımı kara bir kutu olarak ele alır. Testte, çeşitli girdiler uygulanır ve doğrulukları sınamak için elde edilen çıktılar olması gereken çıktılar ile karşılaştırılır. Tüm test senaryoları istenilen çıktılara, kodun uygulama ayrıntıları dikkate alınmaz.

Giriş uzayını ne kadar büyütürsek, o kadar çok sorun bulacağımız ve bu nedenle yazılımın kalitesinden daha fazla emin olacağımız açıktır. İdeal olarak, girdi alanını kapsamlı bir şekilde test etmek istenilir. Ancak geçersiz girdiler, zamanlama, sıra ve kaynak değişkenleri göz önüne alınmaksızın, geçerli girdilerin kombinasyonlarının kapsamlı bir şekilde test edilmesi programların çoğu için imkânsız olacaktır. Kombinatoryal bir patlama, fonksiyonel testlerde önemli bir engeldir. Ayrıca, karşılaşılabilecek her durumu önceden tam olarak belirlemek mümkün olmayabilir. Gereksinimlerin belirlenmesindeki problemler yazılımdaki tüm hataların yaklaşık yüzde 30'una katkıda bulunur (Beizer, 1995).

İyi bir test planı sadece kara kutu testini değil, aynı zamanda beyaz kutu yaklaşımlarını ve bu ikisinin kombinasyonlarını da içermesi gerekecektir.

2.5.3.2. Beyaz Kutu Testi

Kara kutu testinin aksine, test edilen yazılımın yapısı ve akışı test uygulayıcıları tarafından görülebildiğinden, yazılım bu testte beyaz bir kutu veya cam bir kutu olarak görülür. Test planları, kullanılan programlama dili, mantığı ve akışı gibi yazılım uygulamasının ayrıntılarına göre yapılır. Test senaryoları programın yapısından türetilir. Beyaz kutu testine ayrıca cam kutu testi, mantık tabanlı test (Myers, 1979) veya tasarım tabanlı test de (Hetzel, 1988) denir.

Beyaz kutu testinde birçok teknik vardır. Yazılımın her kod satırı en az bir kez yürütülür, her dallanma ifadesi dikkatle irdelenir, doğru ve yanlış durum tahminlerinin tüm olası kombinasyonlarını kapsayan senaryolar üretilir. Kontrol akışı testi, döngü testi ve veri akışı testi kullanılan tekniklerden birkaçıdır. Test senaryoları, tüm düğümlerin veya yolların en az bir kez kaplandığı veya geçildiği kriterine göre dikkatle seçilir. Bunu yaparak, hiç kullanılmayan ya da hiç çalıştırılmayan kodlar yani gereksiz kodlar da keşfedilebilir (Parrington ve Roper, 1989).

Mutasyon testinde, orijinal program kodu bozulur ve her biri bir hata içeren birçok mutasyona uğramış program oluşturulur. Programın her hatalı sürümüne mutant denir. Test verileri, mutantların başarısız olmasının etkinliğine göre seçilir. Bir test vakası ne kadar çok mutant öldürebilirse, test vakası o kadar iyi kabul edilir. Mutasyon testiyle ilgili sorun, kullanımda çok hesaplama ihtiyacı doğurmasıdır. Kara kutu yaklaşımı ile beyaz kutu yaklaşımı arasındaki sınır kesin değildir. Yukarıda belirtilen birçok test stratejisi, kara kutu testi veya beyaz kutu testi olarak güvenli bir şekilde sınıflandırılamaz. Bu işlem akışı testi, sözdizimi testi, sonlu durum testi ve diğer birçok test stratejisi için de geçerlidir. Bunun bir nedeni, yukarıdaki tüm tekniklerin test edilen yazılımın özellikleri hakkında biraz bilgiye ihtiyaç duymasıdır (Pan, 1999).

2.5.3.3. Performans Testi

Tüm yazılım sistemlerinin performansla ilgili özel gereksinimleri açıkça bulunmayabilir. Ancak her sistemin mutlaka performans gereksinimleri de olacaktır. Yazılımın yürütülmesi sonsuz zaman veya sonsuz kaynak kullanmamalıdır. Performans hataları bazen yazılımdaki sistem performansının düşmesine neden olan tasarım problemlerini ifade etmek için kullanılır. Performans yazılımcılar için her zaman büyük bir endişe kaynağı ve bilgisayar evriminin itici gücü olmuştur. Bir yazılım sisteminin performans değerlendirmesi genellikle kaynak kullanımı, işlem hacmi, yanıt süresi ve seçilen kaynaklar tarafından hizmet verilmesini bekleyen

ortalama veya maksimum görev sayısını detaylandıran kuyruk uzunluklarını içerir (Smith, 1990).

Performans testinin amacı, performans darboğazı tanımlaması ve performans karşılaştırması yapmaktır. Performans testi yapmanın tipik yöntemi sistem kullanımını temsil edecek şekilde tasarlanmış bir program, iş yükü veya iz dosyasını temel alarak bir karşılaştırma ölçütü kullanmaktır (Vokolos ve Weyuker, 1998).

2.5.3.4. Sağlamlık ve Stres (Yük) Testi

Yazılım güvenilirliği, test süreci de dahil olmak üzere yazılımın birçok yönüyle ilgilidir. Yazılım güvenilirliğini doğrudan tahmin etmek zor olabilir. Test, yazılım güvenilirliğini ölçmek için etkili bir örnekleme yöntemidir. Operasyonel profilin rehberliğinde yazılım testi hata verilerini elde etmek için kullanılabilir ve ayrıca mevcut güvenilirliği tahmin etmek ve gelecekteki güvenilirliği kestirmek için verileri analiz ederek bir tahmin modeli oluşturulabilir. Bu nedenle, tahminlere dayanarak, geliştiriciler yazılımı piyasaya sürüp sürmemeye karar verebilir ve yazılımı kullanma riski güvenilirlik bilgisine dayanarak da değerlendirilebilir. (Hamlet, 1994), testin birincil amacının test edilen yazılımın güvenilirliğini ölçmek olduğunu savunmaktadır.

Güvenilir yazılım beklenmedik veya yıkıcı yollardan başarısız olmaz. Hamlet'e (1994) göre sağlamlık testi ve stres testi, bu basit kritere dayanan güvenilirlik testinin varyanslarıdır. Bir yazılım bileşeninin sağlamlığı, istisnai girdiler veya stresli ortam koşullarında doğru işlev görebilme derecesidir. Sağlamlık testi, yazılımın işlevsel doğruluğundan ziyade makine çökmeleri, süreç kilitlenmeleri veya anormal sonlandırma gibi sağlamlık sorunlarını izler (Koopman ve DeVale, 1999). Stres testi veya yük testi ise yalnızca yazılım yerine tüm sistemi test etmek için kullanılır. Tipik stres, kaynak tükenmesi, faaliyet patlamaları ve sürekli yüksek yükleri içerir (IEEE, 1990).

2.5.3.5. Güvenlik Testi

Yazılımın kalitesi, güvenilirliği ve güvenliği sıkı sıkıya birbirine bağlıdır. Yazılımdaki kusurlar, güvenlik açıklarını açmak için davetsiz misafirlerce kullanılabilir. İnternetin gelişmesiyle birlikte, yazılım güvenliği sorunları daha da ciddi hale gelmiştir.

Birçok kritik yazılım uygulaması ve hizmeti, kötü niyetli saldırılara karşı entegre güvenlik önlemlerine sahiptir. Bu sistemlerin güvenlik testinin amacı, potansiyel olarak güvenlik ihlallerine yol açabilecek yazılım hatalarını tanımlamayı ve kaldırmayı ve güvenlik önlemlerinin etkinliğini doğrulamayı içerir. Güvenlik açıklarını bulmak için simüle edilmiş güvenlik saldırıları gerçekleştirilebilir. (Felderer ve ark., 2016)

2.6. Yazılım Güvenilirliği Modelleri

Yazılım kalite ölçümü için, bir yazılım sisteminde kalan hataların bir tahmininin yapılması arzu edilir. Yazılımdaki ilk hataların ne olduğunu bilmeden böyle önemli bir bulguyu belirlemek zordur. Yazılımdaki hataların tahminini yapmak ve kalan hata sayısını kestirmek için birçok istatistiksel model ve çeşitli teknikler geliştirilmiştir.

İki tür yazılım güvenilirliği modeli vardır: deterministik ve olasılıksal modeller. Deterministik modeller, bir programdaki farklı operatörlerin ve işleçlerin sayısından yararlanarak programdaki hataların sayısını incelemek için kullanılır. Deterministik tipteki performans ölçümleri, programın dokusu analiz edilerek elde edilir ve herhangi bir rastgele olay içermez. Bilinen iki model Halstead'in yazılım metriği ve McCabe'nin siklomatik karmaşıklık metriğidir. Halstead'in yazılım metriği programdaki hata sayısını tahmin etmek için kullanılırken Halstead (1977), McCabe'nin siklomatik karmaşıklık metriği (McCabe 1976), kalan yazılım hatalarının sayısını tahmin etmek için model üzerinde bir üst sınır belirlemek için

kullanılır. Genel olarak, bu modeller bilgisayar yazılımının karmaşıklığı oranında artan nicel bir yaklaşımı temsil etmektedir.

Olasılıksal model, başarısızlık olaylarını ve başarısızlık tamirlerini olasılık olayları olarak kabul eder. Olasılıksal modellerin deterministik modellere göre daha başarılı olduğu söylenebilir (Pham, 2006). Farklı olasılıksal yazılım güvenilirlik modelleri izleyen konu başlıklarında incelenmiştir.

2.6.1. Hata Tohumlama Modelleri

Hata tohumlama grubu modelleri, çok aşamalı örnekleme tekniğini kullanarak bir programdaki hata sayısını tahmin eder. Hatalar yerleşik hatalara ve ortaya çıkan hatalara (tohumlanmış hatalar) ayrılır. Bilinmeyen yerleşik hata sayısı, ortaya çıkan hataların sayısından ve hata ayıklama verilerinden elde edilen iki tür hatanın oranından tahmin edilir.

2.6.1.1. Mills Hata Tohumlama Modeli

Mills hata tohumlama modelinde Mills (1970), bir programdaki hata sayısını tahmin etmek için programa yeni hatalar dahil etme (hata tohumlama) yöntemi önerdi. Doğal hatalar ve tohumlanmış hatalardan oluşan hata ayıklama verilerinden, bilinmeyen doğal hata sayısı tahmin edilebilir. Hem doğal hataların hem de uyarılan hataların eşit olarak algılanması muhtemelse, r adet düzeltilen hatada k tohumlanan hata bulma olasılığı, aşağıdaki eşitlikte verilen hipergeometrik dağılımı izler.

$$P(k; n, n_1, r) = \frac{\binom{n_1}{k}\binom{N}{r-k}}{\binom{n+n_1}{r}}, \qquad k = 1,2,3,\cdots,r$$

Bu eşitlikte N toplam doğal hata sayısını, n_1 toplam tohumlanmış hata sayısını, r toplam tespit edilip düzeltilmiş hata sayısını, k toplam tespit edilip düzeltilen tohumlanmış hata sayısını göstermektedir.

Huang'a (1984) göre n_1, k ve r bilindiğinde N'nin MLE'si

$$\widehat{N} = \left\lfloor \frac{n_1(r-k)}{k} \right\rfloor$$

olarak elde edilir.

2.6.1.2. Cai Modeli

Cai (1998), Mills'in modelini, yazılımı iki parçaya bölerek değiştirdi. Bu modelde aşağıdaki varsayımlar uygulanır:

1) Yazılım toplam N adet hata içerir. Bu hataların N_0 adedi bölüm-0'da ve N_1 adedi bölüm-1'de bulunmaktadır. $N = N_0 + N_1$
2) Kalan hataların hepsi aynı tespit edilebilme olasılığına sahiptir.
3) Tespit edilen her hata anında düzeltilir.
4) Her seferinde sadece bir hata düzeltilir ve düzeltme işleminde yeni hata oluşmaz.

t_i, düzeltilen i inci hatanın tespit edildiği zaman olsun ve Y_i rastgele değişkeni aşağıdaki gibi tanımlansın,

$$Y_i = \begin{cases} 0 \ ; & i\ inci\ hata\ bölüm - 0'da\ ise \\ 1 \ ; & i\ inci\ hata\ bölüm - 1'de\ ise \end{cases}$$

Bu durumda Cai (1998), N_0 ve N_1'in MLE tahminlerinin aşağıdaki iki eşitliğin çözülmesi ile elde edileceğini göstermiştir.

$$\sum_{i=1}^{n} \frac{1-y_i}{N_0 - i + 1 + \sum_{j=0}^{i-1} y_j} = \frac{1}{N_0 + N_1 - i + 1}$$

$$\sum_{i=1}^{n} \frac{y_i}{N_1 - \sum_{j=0}^{i-1} y_j} = \frac{1}{N_0 + N_1 - i + 1}$$

2.6.1.3. Hipergeometrik Dağılım Modeli

Tohma ve ark. (1991), hipergeometrik dağılıma dayalı olarak bir programda başlangıçta bulunan hataların sayısını tahmin etmek için bir model önermişlerdir. Tohma, Mills ve Cai'den farklı olarak modelinde tespit edilen hataların düzeltilmediğini varsaymıştır. C_{i-1} şimdiye kadar $t_1, t_2, \cdots, t_{i-1}$ zamanların tespit edilen toplam hata sayısı ve N_i ise t_i zamanında yeni algılanan hata sayısı olduğunu ve N_i'nin hipergeometrik dağılıma sahip olduğunu varsayalım. Tohma ve ark. (1991) belirli varsayımlar altında C_i'nin beklenen değerinin aşağıdaki eşitlik yardımıyla bulunabileceğini göstermişlerdir.

$$E(C_i) = m\left[1 - \sum_{j=1}^{i}(1 - p_j)\right]$$

Bu eşitlikte, $p_j = \frac{W_i}{m} \quad i = 1,2,\cdots$, m başlangıçtaki hata sayısı, W_i i zamanında tespit edilen hataların (tekrar tespit edilen hatalar da dahil) sayısıdır.

2.6.2. Başarısızlık Oranı Modelleri

Başarısızlık oranı modelleri, zaman aralıklarında başarısızlık oranlarının nasıl değiştiğini inceler. Kalan hataların sayısı değiştikçe, programın başarısızlık oranı da buna göre değişir. Programdaki hata sayısı ayrık bir fonksiyon olduğundan, programın başarısızlık oranı aynı zamanda başarısızlık zamanlarında süreksizlikleri olan ayrık bir fonksiyondur.

2.6.2.1. Jelinski-Morenda Modeli

Jelinski-Moranda (J-M) modeli (Jelsinki, 1972) en eski yazılım güvenilirlik modellerinden biridir. Mevcut birçok yazılım güvenilirliği modeli, bu temel modelin varyantları veya uzantılarıdır. Bu modelin varsayımları şunlardır,

1) Program, başlangıçta N adet bilinmeyen fakat sabit hata içeriyor.
2) Programdaki her hata bağımsızdır.
3) Başarısızlık oluşumları arasındaki zaman aralıkları birbirinden bağımsızdır.
4) Bir başarısızlık meydana geldiğinde, başarısızlığa karşılık gelen hata kesin olarak ortadan kaldırılır.
5) Başarısızlığa neden olan hatanın anında giderildiği varsayılır ve tespit edilen hatanın giderilmesi sırasında yeni hatalar oluşmaz.
6) İki başarısızlık arasındaki yazılım başarısızlık oranı sabittir ve programda kalan hata sayısıyla orantılıdır.

Jelinski-Morenda modelinde i inci başarısızlık aralığındaki başarısızlık oranı aşağıdaki eşitlik ile temsil edilir.

$$\lambda(t_i) = \phi[N - (i-1)] \quad , \quad i = 1,2,\cdots,N$$

Burada, ϕ herhangi bir hatanın programa yaptığı katkıyı belirten bir sabit, N başlangıçtaki toplam hata sayısı, t_i (i-1) inci ve i inci başarısızlıklar arasında geçen süredir. Modelin oyf'nu ve kdf'nu,

$$f(t_i) = \lambda(t_i)e^{-\int_0^{t_i}\lambda(x_i)dx_i} = \phi[N-(i-1)]e^{-\phi[N-(i-1)]t_i}$$

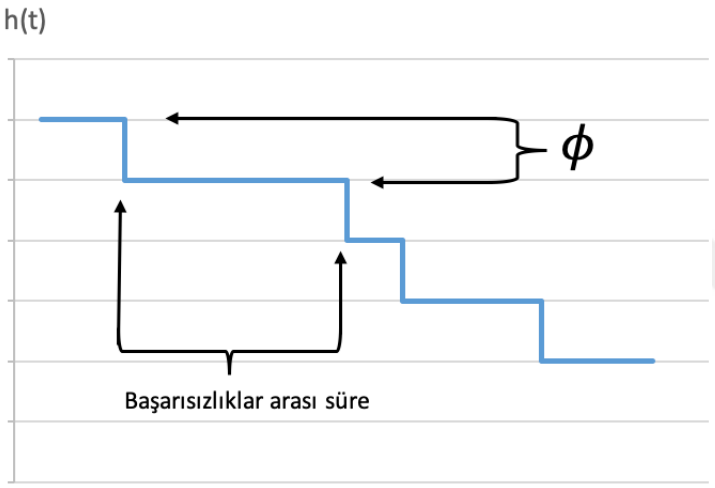
$$F(t_i) = \int_0^{t_i} f(x_i)dx_i = 1 - e^{-\phi[N-(i-1)]t_i}$$

eşitlikleri ile belirlenir. Modelin yazılım güvenilirlik fonksiyonu ve risk fonksiyonu ise,

$$R(t_i) = e^{-\phi[N-(i-1)]t_i}$$

$$h(t_i) = \phi[N-(i-1)]$$

eşitlikleri ile bulunur.



Şekil 2.3. Jelinski-Morenda modeli risk fonksiyonu grafiği

2.6.2.2. Schick-Wolverton Modeli

Schick-Wolverton (S-W) modeli (Schick ve Wolverton, 1978) J-M modelinden türetilen bir modeldir. S-W modelinde i inci zaman aralığında başarısızlık oranının t_i süresi ile orantılı olarak arttığı varsayılır. Modelde i inci başarısızlık aralığındaki başarısızlık oranı aşağıdaki eşitlik ile temsil edilir.

$$\lambda(t_i) = \phi[N-(i-1)]t_i \quad , \quad i = 1,2,\cdots,N$$

Burada, ϕ herhangi bir hatanın programa yaptığı katkıyı belirten bir sabit, N başlangıçtaki toplam hata sayısı, t_i (i-1) inci ve i inci başarısızlıklar arasında geçen süredir. Modelin oyf'nu,

$$f(t_i) = \phi[N-(i-1)]t_i e^{-\frac{\phi[N-(i-1)]t_i{}^2}{2}}$$

eşitliği ile belirlenir. Modelin yazılım güvenilirlik fonksiyonu ve risk fonksiyonu ise,

$$R(t_i) = e^{-\frac{\phi[N-(i-1)]t_i{}^2}{2}}$$

$$h(t_i) = \phi[N-(i-1)]t_i$$

eşitlikleri ile bulunur.

2.6.2.3. Jelinski-Moranda Geometrik Modeli

Jelinski-Moranda geometrik modeli (Moranda, 1979), program başarısızlık oranı fonksiyonunun başlangıçta D gibi bir sabit olduğunu ve başarısızlık zamanlarında geometrik olarak azaldığını varsayar. Modelde i inci başarısızlık aralığındaki başarısızlık oranı ve güvenilirlik fonksiyonu sırasıyla aşağıdaki eşitlikte gösterildiği gibidir.

$$\lambda(t_i) = Dk^{i-1}$$

$$R(t_i) = e^{-Dk^{i-1}t_i}$$

Burada D başlangıçtaki başarısızlık oranını, k ise geometrik fonksiyonun parametresini göstermektedir.

2.6.2.4. Düzenlenmiş Schick-Wolverton Modeli

Sukert (1977), S-W modelini her zaman aralığında birden fazla başarısızlığa izin verecek şekilde değiştirmiştir. Modelin başarısızlık oranı fonksiyonu,

$$\lambda(t_i) = \phi(N - n_{i-1})t_i$$

eşitliği ile tanımlanır. Burada n_{i-1} (i-1) inci aralıktaki kümülatif başarısızlık sayısıdır. Böylece bu modeldeki yazılım güvenilirlik fonksiyonu,

$$R(t_i) = e^{-\phi(N-n_{i-1})\frac{t_i^2}{2}}$$

eşitliği ile bulunabilir.

2.6.2.5. Goel-Okumoto Kusurlu Hata Ayıklama Modeli

Goel ve Okumoto (1979), bir başarısızlık oluştuğunda hatanın p olasılığı ile giderildiğini varsayarak J-M modelini genişlettiler. Modelde i inci başarısızlık aralığında başarısızlık oranı fonksiyonu,

$$\lambda(t_i) = \phi[N - p(i-1)]$$

ve güvenilirlik fonksiyonu,

$$R(t_i) = e^{-\phi[N-p(i-1)]t_i}$$

eşitlikleri ile tanımlanır.

2.6.3. Eğri Uydurma Modelleri

Bu modeller, yazılım karmaşıklığı, bir programdaki hata sayısı, değişiklik sayısı veya başarısızlık oranı arasındaki ilişkiyi incelemek için istatistiksel regresyon analizini kullanır. Bu model grubu, doğrusal regresyon, doğrusal olmayan regresyon veya zaman serisi analizi yöntemlerini kullanarak bağımlı ve bağımsız değişkenler arasında fonksiyonel bir ilişki bulur. Bağımlı değişken, bir programdaki hata sayısı olabilir. Bağımsız değişkenler bakım aşamasında değiştirilen modül sayısı, başarısızlıklar arasındaki süre, programcıların becerisi, program büyüklüğü gibi değişkenler olabilir.

2.6.3.1. Hata Sayısının Tahmini Modelleri

Bir programdaki hata sayısı doğrusal veya doğrusal olmayan regresyon modeli kullanılarak tahmin edilebilir. Programdaki N başlangıç hatalarının toplam sayısını tahmin etmek için basit bir doğrusal olmayan regresyon modeli aşağıdaki gibi sunulabilir:

$$N = \sum_i a_i X_i + \sum_i b_i {X_i}^2 + \sum_i c_i {X_i}^3 + \varepsilon$$

Burada X_i i inci hata faktörü, $a_i, b_i \; ve \; c_i$ model katsayıları ve ε ise istatistiksel hata terimidir. Sıklıkla kullanılan hata faktörleri, yazılım karmaşıklığı ölçütleri ve çevresel faktörlerdir. Çoğu eğri uydurma modeli yalnızca bir hata faktörü içerir.

2.6.3.2. Yazılım Karmaşıklığının Tahmini Modelleri

Belady ve Lehman (1976), zaman serisi yaklaşımını kullanarak yazılım karmaşıklığını tahmin etmek için bir model önermişlerdir. Yazılım karmaşıklığı modeli aşağıdaki gibi özetlenmiştir:

$$C_R = a_0 + a_1 R + a_2 E_R + a_3 M_R + a_4 I_R + a_5 D + \varepsilon$$

Bu eşitlikte R sürüm sıra numarasını, E_R R sürümü için çevresel faktörü, M_R R sürümündeki modül sayısını, I_R sürümler arasındaki zaman aralığını, D ilk sürüm başarısızlığından bu yana geçen gün sayısını, ε ise istatistiksel hata terimini belirtmektedir. Bu model, birden çok sürüme sahip ve uzun bir süre boyunca geliştirilen yazılımlar için geçerlidir.

2.6.3.3. Başarısızlık Oranının Tahmini Modelleri

Miller (1985) yazılımın başarısızlık oranını tahmin etmek için bir model önerdi. Miller çalışmasında başarısızlık oranının monoton olarak artmadığını varsayarak, λ_i'nin bir tahmini, en küçük kareler yöntemi kullanılarak elde edilebilir. Bu modele göre i inci başarısızlık aralığındaki başarısızlık oranının tahmini

$$\hat{\lambda}_i = \frac{1}{t_{i-1} - t_i}$$

olmaktadır.

2.6.4. Güvenilirlik Büyütme Modelleri

Güvenilirlik büyütme modelleri, test süreci boyunca güvenilirliğin iyileştirilmesini ölçer ve tahmin eder. Büyüme modeli, bir sistemin güvenilirliğini veya başarısızlık oranını zamanın fonksiyonu veya test senaryolarının sayısı olarak temsil eder.

2.6.4.1. Coutinho Modeli

Coutinho (1973), Duane büyüme modelini yazılım test sürecini temsil edecek şekilde uyarladı. Coutinho, test haftalarına göre keşfedilen kümülatif başarısızlık sayısını ve yapılan düzeltmelerin sayısını baz alan bir model oluşturdu. N(t) kümülatif başarısızlık sayısını belirtsin ve t toplam test süresi olsun. Model başarısızlık oranını,

$$\lambda(t) = \frac{N(t)}{t} = \beta_0 t^{-\beta_1}$$

eşitliği ile tanımlar. Burada β_0 ve β_1 model parametreleridir. Modelin parametrelerini tahmin edebilmek için en küçük kareler yöntemi kullanılabilir.

2.6.4.2. Wall ve Ferguson Modeli

Wall ve Ferguson (1977), test sırasında yazılımın başarısızlık oranını tahmin etmek için Weibull büyüme modeline benzer bir model önerdiler. Wall ve Ferguson bu modeli birkaç yazılım başarısızlık verisi kullanarak test etmiş ve başarısızlık verilerinin modelle uyumunun iyi olduğunu gözlemlemiştir. t anında yazılımdaki başarısızlıkların kümülatif sayısını $m(t)$ ile gösterirsek, Wall ve Ferguson modeli,

$$m(t) = a_0[b(t)]^{\beta}$$

eşitliği ile tanımlanabilir. Burada a_0 ve β bilinmeyen parametrelerdir. b(t) fonksiyonu ise toplam test olaylarının sayısından elde edilir. Benzer şekilde t anındaki başarısızlık oranı fonksiyonu ise,

$$\lambda(t) = m'(t) = a_0\beta[b(t)]^{\beta-1}$$

eşitliği ile bulunur.

2.6.5. Markov Modelleri

Bir Markov süreci, sürecin gelecekteki davranışının sadece mevcut duruma bağlı olduğu ve geçmişinden bağımsız olduğu varsayımına sahiptir (bkz. bölüm 2.3). Markov modelleri, yazılımın başarısızlık sürecini temsil etmenin genel bir yoludur. Bu modeller, yazılımın kendi içindeki modüllerin güvenilirliğini ve ilişkisini incelemek için de kullanılabilir. Modüllerin başarısızlıklarının birbirinden bağımsız

olduğu varsayılmaktadır. Bu varsayım, bağımsız olarak tasarlanabileceği, kodlanabileceği ve test edilebileceği için modül düzeyinde makul görünmektedir, ancak sistem düzeyinde doğru olmayabilir.

2.6.5.1. Kusurlu Hata Ayıklama ile Markov Modeli

Goel ve Okumoto (1979) kusurlu hata ayıklama ile doğrusal bir Markov modeli önermiş ve modelin geçiş olasılıkları

$$p_{ij} = \begin{cases} p & ,j = i-1 \; için \\ q & ,j = i \; için \\ 1 & ,i = j = 0 \\ 0 & ,diğer \end{cases}$$

olarak ifade edilebilir. Burada p başarılı hata ayıklama olasılığını, q=1-p yani bir başarısızlık oluştuğunda hata ayıklamanın gerçekleştirilememesi olasılığıdır. Goel'e (1979) göre sistemde k ıncı başarısızlık aralığında güvenilirlik fonksiyonu,

$$R_k(t) = \sum_{j=0}^{k-1} \binom{k-1}{j} p^{k-j-1} q^j e^{-[N-(k-j-1)]\phi t}$$

eşitliği ile bulunur.

2.6.5.2. Littlewood Markov Modeli

Littlewood'un modeli (Littlewood, 1979), Markov süreci olarak yazılımın çalışması sırasında program modülleri arasındaki geçişleri temsil eder. Modelde iki tür başarısızlık dikkate alınır. İlk başarısızlık tipi, her modülün kendi iç yapısında bir Poisson başarısızlık işleminden gelir. Modüller entegre edildiğinde yeni hataların ortaya çıkacağı kabul edilmektedir. İkinci başarısızlık tipi, modüller arasındaki arayüzden kaynaklanabilecek başarısızlıklardır.

Modüller ve arayüzlerdeki başarısızlıkların birbirinden bağımsız olduğu varsayılarak, Littlewood (1979), program başarısızlık sürecinin asimtotik olarak

$$\lambda = \sum_{i=1}^{n} \pi_i \left(\lambda_i + \sum_{i \neq j} a_{ij} q_{ij} \right)$$

başarısızlık oranına sahip bir Poisson süreci olduğunu göstermiştir. Burada n modüllerin sayısını, a_{ij} i modülünden j modülüne geçiş sürecini, λ_i i modülünün Poisson başarısızlık oranını, q_{ij} i modülünden j modülüne geçişin başarısız olma olasılığı, π_i i modülünün çalışma olasılığını göstermektedir.

2.6.5.3. Yazılım Güvenlik Modeli

Yamada ve ark. (1998), Markov süreçlerini kullanarak yazılımın zamana bağlı davranışını tanımlamak için yazılım güvenlik modelini önermiştir. Bu güvenlik modelindeki varsayımlar şunları içerir:

1) Yazılım sistemi çalışırken, güvenli ve güvensiz durumun bekleme süreleri sırasıyla $1/\theta$ ve $1/\eta$ ortalamalarıyla üstel dağılımları takip eder.
2) Bir yazılım başarısızlığı oluştuğunda bir hata ayıklama etkinliği gerçekleştirilir. Hata ayıklama faaliyetleri a olasılığı ile mükemmel olmakla birlikte b olasılığı ile kusurludurlar.
3) Mükemmel hata ayıklama durumlarında yazılım güvenilirliği artışı meydana gelir. Yazılım başarısızlıkları arasındaki zaman aralığı, ortalama 1/n ile üstel bir dağılımı takip eder; burada n=0,1,2,..., düzeltilen hataların kümülatif sayısını gösterir.
4) İki veya daha fazla yazılım başarısızlığının aynı anda meydana gelme olasılığı göz ardı edilebilir.

t anında yazılım sisteminin durumunu temsil eden stokastik bir {X(t), t>=0} sürecini düşünün. X(t)'nin durum uzayı W_n sistemin güvenli çalışma durumu ve U_n sistemin güvensiz duruma düşmesi alt kümeleri ile tanımlanır.

Yamada ve ark. (1998) yazılım güvenilirliği büyüme sürecini tanımlamak için Moranda'nın modelini kullanmıştır. N hata düzeltildiğinde, bir sonraki yazılım başarısızlığı oluşumunda başarısızlık oranı.

$$\lambda_n = Dk^n$$

olacaktır. Burada D>0 ve 0<k<1 sırasıyla başlangıçtaki başarısızlık oranı ve başarısızlık oranının azalma oranıdır.

Yamada, yazılım güvenliğini, sistemin t anında güvenli olmayan durumlara düşmeme olasılığı olarak tanımlanır ve çalışmasında aşağıdaki şekilde vermiştir:

$$S(t) = \sum_{n=0}^{\infty} P_n(t)$$

ve

$$P_n(t) = P\{X(t) = W_n\} = A_n e^{-(\lambda_n+\theta+\eta)^t} + \sum_{i=0}^{n} B_{ni} e^{-a\lambda_i t}$$

Burada A_n ve B_{ni} katsayıları,

$$A_n = \frac{-\theta \prod_{j=0}^{n-1} a\lambda_j}{\prod_{j=0}^{n}(a\lambda_j - \lambda_n - \theta - \eta)}$$

$$B_{ni} = \frac{(\lambda_n + \eta - a\lambda_i)\prod_{j=0}^{n-1}\lambda_j}{(\lambda_n + \theta + \eta - a\lambda_i)\prod_{\substack{j=0 \\ j\neq i}}^{n}(\lambda_j - \lambda_i)}$$

2.6.6. Zaman Serisi Modelleri

Box ve Reinsel (1994) tarafından Otoregresif Entegre Hareketli Ortalama (ARIMA) adı verilen genel bir zaman serisi modelini kullanarak yazılım güvenilirliği modeli önermişlerdir.

ARIMA modeli aşağıdaki eşitlik ile tanımlanır.

$$z_t = \delta + \phi_1 z_{t-1} + \phi_2 z_{t-2} + \cdots + \phi_p z_{t-p} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \cdots - \theta_q a_{t-q}$$

Burada ϕ_i modelin otoregresif kısmının parametreleridir, θ_i hareketli ortalama kısmının parametreleridir. p ve q parametreleri negatif olmayan tam sayılardır. p, otoregresif modelin sırası (zaman gecikmelerinin sayısı) ve q hareketli ortalama modelin sırasıdır.

Varsayalım ki $y_1, y_2, \cdots, y_n$ değişkenleri n=1, 2, … birim zamanlarında test aşamasında yazılımın kümülatif başarısızlık sayılarını göstersin. Durağan olmayan zaman serilerinde değerlerin birinci veya ikinci farkını alabilir ve dönüştürülen serinin sabit bir durum olup olmadığını incelenebilir.

z_t ise t inci test periyodunda meydana gelen başarısızlıkların sayısı olsun.

$$z_t = y_t - y_{t-1} \quad , t = 2,3,\cdots,n$$

şeklinde yazılabilir. w ise z'deki ilk fark (dolayısıyla y'deki ikinci fark) olsun. Böylece,

$$w_t = z_t - z_{t-1} = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) = y_t - 2y_{t-1} + y_{t-2}$$

eşitliğini elde ederiz. Buradan ARIMA fonksiyonunu y terimine göre elde edebiliriz. Örneğin p=0, d=2 ve q=3 iken y değişkeni için ARIMA(0,2,3) fonksiyonu,

$$y_t - 2y_{t-1} + y_{t-2} = \delta + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \theta_3 a_{t-3}$$

eşitliği ile bulunabilir. Burada d farklılaşma derecesidir (verilerin geçmiş değerlerin çıkarılma sayısı). Böylece bu eşitlikten yola çıkılarak yazılımın güvenilirliğini tahmin edilebilir.

2.6.7. Homojen Olmayan Poisson Süreci Modelleri

Homojen olmayan Poisson Süreçleri (HOPS) model grubu, test sırasında yazılım hatası olgusunu tanımlamak için analitik bir çerçeve sağlar. HOPS modelindeki ana konu, belirli bir zaman dilimine kadar yaşanan kümülatif başarısızlık sayısının ortalama değer fonksiyonunu tahmin etmektir.

Yazılım güvenilirliğini değerlendirmek için bir dizi HOPS yazılım güvenilirliği büyüme modeli önerilmiştir. Aslında, HOPS tabanlı yazılım güvenilirlik modelleri, pratik yazılım güvenilirlik mühendisliğinde oldukça başarılı araçlar olmuştur. Bu modeller, hata ayıklama işlemini, ortalama değer fonksiyonu ile karakterize edilen bir sayma işlemi olarak kabul ederler. Ortalama değer fonksiyonu belirlendikten sonra yazılım güvenilirliği tahmin edilebilir. Model parametreleri genellikle maksimum olabilirlik yöntemi veya regresyon kullanılarak tahmin edilir. Farklı varsayımlar üzerine farklı modeller oluşturulmuştur.

HOPS modelindeki ana sorun, beklenen sayıda başarısızlığı belirtmek için uygun bir ortalama değer fonksiyonu belirlemektir ve bu nedenle, belirli bir zamana kadar tecrübe kazanılır. Farklı varsayımlar, model ortalama değer fonksiyonunun farklı fonksiyonel formlarıyla sonuçlanacaktır. N(t) dağılımının m(t) ortalamalı bir poisson dağılımı olduğu varsayımıyla,

$$P\{N(t) = k\} = \frac{[m(t)]^k}{k!} e^{-m(t)} \quad , \quad k = 0,1,2,\cdots$$

eşitliğini elde ederiz. Kümülatif başarısızlık sayılarının ortalama fonksiyonu olan m(t), yazılımdaki başarısızlık oranı fonksiyonu ile aşağıdaki gibi ifade edilebilir.

$$m(t) = \int_0^t \lambda(x)dx$$

Yazılımın güvenilirlik fonksiyonu ise,

$$R(t) = e^{-m(t)} = e^{\int_0^t \lambda(x)dx}$$

olacaktır.

HOPS modelleri hem hata içeriği fonksiyonunun hem de hata algılama hızının zamana bağlı olmasına izin vererek, genelleştirilmiş bir yazılım güvenilirlik modeli ve ortalama değer fonksiyonu için analitik bir ifade sunmaktadır. HOPS'ne dayanan çok sayıda mevcut model de Çizelge.2.2'de özetlenmiştir. HOPS'ne dayalı yazılım sistemlerinin başarısızlıkları arasındaki ortalama süreyi (BAOS) hesaplamak için genel bir fonksiyon da sunulmaktadır. HOPS, yazılım güvenilirliğini tahmin etmek için gerçekçi bir modeldir ve yazılımın hata ayıklama ve test edilmesinde oldukça yararlıdır.

2.7. Yazılım Güvenilirlik Analizinde Yeni Yaklaşımlar

Yapay zekâ (YZ), II. Dünya Savaşı'nın hemen sonrasında bilgisayar biliminin bir alt araştırma alanı olarak ortaya çıktı ve 1970'lerde yazılım mühendisliğine doğru genişlemeye başladı. Son zamanlarda, YZ ve daha spesifik olarak makine öğrenimi, sahip olunması gereken bir teknoloji ve karmaşık etik ve

teorik soruları ele alan çok aktif bir araştırma alanı haline geldi. Hesaplama gücünün ve gelişmiş algılama ve veri iletişim altyapılarının mevcudiyetindeki sürekli artışla bu konudaki çalışmalarda yoğun bir artış yaşanmıştır.

Çizelge 2.2. Bazı HOPS yazılım güvenilirlik modelleri

Model	m(t)
Goel-Okumato Modeli (Goel ve Okumato, 1979)	$m(t) = a(1 - e^{-bt})$ $a(t) = a \qquad b(t) = b$
Gecikmeli S-şekilli (Yamada ve ark., 1984)	$m(t) = a(1 - (1 + bt)e^{-bt})$ $a(t) = a \qquad b(t) = \frac{b^2 t}{1+bt}$
Kıvrımlı S-şekilli (Ohba ve Yamada, 1984)	$m(t) = \frac{a(1 - e^{-bt})}{1 + \beta e^{-bt}}$ $a(t) = a \qquad b(t) = \frac{1}{1+\beta e^{-bt}}$
Yamada üstel modeli (Yamada ve Osaki, 1985)	$m(t) = a\left(1 - e^{-r\alpha\left(1-e^{-\beta t}\right)}\right)$ $a(t) = a \qquad b(t) = r\alpha\beta e^{-\beta t}$
Yamada-Rayleigh modeli (Yamada ve Osaki, 1985)	$m(t) = a\left(1 - e^{-r\alpha\left(1-e^{-\frac{\beta t^2}{2}}\right)}\right)$ $a(t) = a \qquad b(t) = r\alpha\beta t e^{-\frac{\beta t^2}{2}}$
Yamada kusurlu hata ayıklama modeli (1) (Yamada ve ark., 1984)	$m(t) = \frac{ab}{\alpha + b}(e^{\alpha t} - e^{-bt})$ $a(t) = ae^{\alpha t} \qquad b(t) = b$
Yamada kusurlu hata ayıklama modeli (2) (Yamada ve ark., 1984)	$m(t) = a(1 - e^{-bt})\left(1 - \frac{\alpha}{b}\right) + \alpha a t$ $a(t) = a(1 + \alpha t) \qquad b(t) = b$
Pham üstel kusurlu hata ayıklama modeli (Pham ve ark., 2000)	$m(t) = \frac{\alpha b}{b + \beta}\left(\frac{e^{(b+\beta)t} - 1}{e^{bt} + c}\right)$ $a(t) = \alpha e^{\beta t} \qquad b(t) = \frac{b}{1+ce^{-bt}}$
PNZ modeli (Pham ve ark, 1999)	$m(t) = \frac{a}{1 + \beta e^{-bt}}\left((1 - e^{-bt})\left(1 - \frac{\alpha}{b}\right) + a\alpha t\right)$ $a(t) = a(1 + \alpha t) \qquad b(t) = \frac{b}{1+\beta e^{-bt}}$
Pham-Zang Modeli (Pham ve Zang, 1997)	$m(t) = \frac{a}{1 + \beta e^{-bt}}\left((c + a)(1 - e^{-bt}) - \frac{a}{b - \alpha}(e^{-\alpha t} - e^{-bt})\right)$ $a(t) = c + a(1 - e^{-\alpha t}) \qquad b(t) = \frac{b}{1+\beta e^{-bt}}$

Makine öğrenimi, bilgisayarların yazılımın geçmiş ve şimdiki arıza verilerine dayanarak sistem davranışını geliştirmesine izin veren algoritmaların tasarımı ve geliştirilmesiyle ilgili bir yaklaşımdır. Bu nedenle, makine öğrenimi teknikleri otomatik olarak öğrenmeye, karmaşık kalıpları tanımaya ve geçmiş verilere dayanarak akıllı kararlar almaya odaklanır. Böylece bir makine, yapısını, programını veya verilerini girdilerine dayalı olarak veya dış bilgilere yanıt olarak değiştirdiğinde, gelecekteki verilerin beklenen performansının önemli ölçüde artacağı şekilde öğrenebilir (Kohavi, 1995). Bu nedenle, yazılım uygulayıcıları ve araştırmacıların, belirli bir başarısızlık veri seti için hangi belirli yöntemin iyi ve nicel olarak ne ölçüde işe yaradığını bilmesi önemlidir (Aggarwal ve ark, 2006; Goel ve Singh, 2009; Singh ve Kumar, 2010).

Kumar ve Singh (2012) yazılım güvenilirliğinin tahmini için yapay sinir ağları, destek vektör makineleri, evrişimsel sinir ağları ve bulanık mantık gibi makine öğrenimi yöntemlerini kullanarak deneysel bir çalışması yapmıştır. Zeng (2009) yapay sinir ağı, Costa ve ark. (2010) genetik programlama tekniği, Amin ve ark. (2013), zaman serisi modelleri, Tamura ve ark. (2016) derin öğrenme yöntemlerinden faydalanarak yazılım güvenilirliğini kestirecek yöntemler geliştirmişlerdir. Brahati ve Selverani (2020), mantık hataları içeren yazılım güvenilirlik tahmininde gizli Markov modelleri yaklaşımını kullanmışlardır. Suharson ve Prahba (2020), hibrit yazılım güvenilirliği için karınca kolonisi algoritması yaklaşımı önermişlerdir. Jaiswal ve Melhotra (2018), farklı makine öğrenmesi yöntemlerini karşılaştıran bir çalışma yapmışlardır. Braiek ve Khomh (2020) ise makine öğrenmesi yöntemlerini yazılım testleri aşamasında kullanımını inceleyen bir çalışma yapmışlardır.

Bu çalışmada genetik algoritma yöntemi kullanarak eğri uydurma yöntemiyle başarısızlık oranı tahmini için yöntemler geliştirilmiştir.

3. GENETİK ALGORİTMALAR

Genetik Algoritma (GA) yaklaşımının ortaya çıkışı 1970'lerin başında olmuştur. John Holland (1975), makine öğrenmesi üzerine yaptığı çalışmalarda canlılardaki evrimden ve değişimden etkilenmiş ve Genetik Algoritmaları geliştirmiştir. Genetik algoritmalar makine öğrenmesi ve optimizasyon uygulamalarında sıklıkla kullanılır.

Genetik algoritma, pek çok olası çözümün içerisinde en uygununu bulmaya çalışan algoritmalardır. Herhangi bir çözüme ilişkin tüm bilgiyi içeren dizilere kromozom denir. Her nesilde kötü çözümler yok olma, iyi çözümler ise daha iyi çözümler oluşturmak için kullanılma eğilimindedirler. Genetik algoritmalar çözüm uzayının tamamını değil yalnızca bir kısmını tararlar. Böylece etkin arama yaparak çok daha kısa sürede çözüme ulaşırlar. Genetik algoritmaların önemli bir üstünlüğü ise popülasyondaki çeşitliliği korumak suretiyle yerel çözümlerin etkisinden kurtulmalarıdır.

Problemde istenilen amacın ne kadar gerçekleştirildiğini belirlemek için bir uygunluk fonksiyonu belirlenir. Genellikle uygunluk fonksiyonunun maksimum veya minimum olması istenir. Uygunluk fonksiyonu kromozomun kalitesini belirler. Bir popülasyondaki kromozomun bir sonraki kuşağa bilgilerini aktarıp aktaramayacağında uygunluk fonksiyonunun değeri çok önemli bir rol oynar.

Algoritmanın temel operatörleri, eş seçme, üreme, çaprazlama, mutasyon gibi biyolojiden esinlenmiştir. Algoritma daha uygun sonuçlara ulaşabilmek için doğadaki en uygun olanın hayatta kalma ilkesini uygular. Her nesilde, problem alanındaki uygunluk düzeylerine göre diğer çözümlerden daha uygun olan bireylerin yaşamasına olanak vererek, doğal adaptasyonda olduğu gibi, popülasyonun daha çok gelişmesine yol açar ve böylece en uygun sonucu bulmaya çalışır.

3.1. Problemin Temsili: Kromozomların Kodlanması

Bireyler / fenotipler, genetik algoritmanın en önemli parçası olarak kabul edilebilir. Birçok yönden, doğada bulunan organizmalar gibi davranırlar. Kendi benzersiz özelliklerine sahiptirler ve diğer benzer bireylerden oluşan bir popülasyonunun parçasıdırlar. Hayatta kalmak ve üremek için savaşırlar. Bireylerin genetik algoritmada doğada olduğu gibi karmaşık yaşamları yoktur, ancak sorunlara bir çözüm olarak düşünülebilirler. Bireyler genellikle bir fonksiyon veya simülasyon için bir model veya parametre olarak kodlanırlar.

Temsil, fenotip (karar değişkenleri) uzayından genotip (kromozom, kodlanmış çözüm) uzayına yapılan bir eşlemedir. Bir kromozom kodlanarak temsil ettiği çözüm hakkında bilgi içermelidir. Tamsayı veya gerçek sayılar, ağaç, permütasyon, sekizlik, onaltılık kodlamalar şeklinde kodlamanın birçok yolu vardır. Bu çalışmada ikili kodlama ile ilgilendiğimizden, ikili genetik algoritma ve ilgili görevler anlatılmaktadır.

İkili kodlama en yaygın kodlamadır ve Çizelge 3.1'de gösterilmiştir. Bu gösterimde, her bir kromozom bit dizgisine kodlanır ve her bit, bir çözümün bazı özelliklerini kodlar. Tamsayılar veya gerçek sayılar gibi sayısal değerler de genelde ikili kodlamaya çevrilerek kullanılırlar.

Çizelge 3.1. İkili kodlama

Kromozom-1	1	0	1	1	1	1	0	0	1
Kromozom-2	0	1	1	1	0	0	0	1	1
Kromozom-3	1	1	0	0	0	0	0	1	0

3.2. Başlangıç Popülasyonu

Bir popülasyon, bireylerden oluşan bir koleksiyondur ve test edilen birkaç bireyden, bireyleri tanımlayan fenotip parametrelerinden ve arama alanıyla ilgili bazı

bilgilerden oluşur. Genetik Algoritmalarda kullanılan popülasyonun iki önemli yönü şunlardır:

- İlk nüfus oluşturma mekanizması.
- Nüfus büyüklüğü.

Dikkate alınması gereken ana sorular, ilk olarak popülasyonun büyüklüğü ve ikincisi, bireylerin seçim yöntemidir. Popülasyon büyüklüğünün seçiminde temelde yatan fikir her zaman verimlilik ve etkililik arasında bir denge kurabilmektir.

Bazı algoritmalar, sonraki popülasyonlarda çok daha azıyla çalışmadan önce olabildiğince fazla çeşitliliğe izin vermek için çok büyük bir başlangıç popülasyonu içerir. Genel olarak, daha büyük popülasyonlar, algoritmaya daha fazla çeşitlilik ve sıklıkla daha iyi sonuçlar verir. Özetlemek gerekirse, büyük bir popülasyon oldukça kullanışlıdır, ancak çok daha fazla hesaplama maliyeti, bellek ve zaman gerektirir. Çizelge 3.2, 4 kromozomdan oluşan rastgele bir popülasyon örneğini göstermektedir.

Çizelge 3.2. Her kromozomu 5 bitle kodlanan rasgele bir başlangıç popülasyonu

Popülasyon	Kromozom-1	1	1	1	0	1
	Kromozom-2	0	0	0	1	1
	Kromozom-3	1	0	0	1	0
	Kromozom-4	0	1	0	1	0

3.3. Uygunluk Fonksiyonu ve Değerlendirme

Bir bireyin bir genetik algoritmada uygunluğu, ilgili kromozomun fenotipi için bir amaç fonksiyonunun değeridir. Uygunluk veya amaç fonksiyonu, çözülecek probleme bağlı olarak, arama alanındaki iyi noktaları temsil eden kromozomların

(olası çözümlerin) yüksek uygunluk değerlerine sahip olacağı şekilde seçilir. Bu, GA'ların olası çözümleri ararken kullandığı çözümün uygunluğu için kullandığı tek bilgidir. Örneğin, Çizelge 3.2.'deki popülasyon için uygunluk fonksiyonu, S = {0, 1, 2, ..., 30, 31} arama uzayında f (x) = x^2 olarak verilsin. Bir kromozomun uygunluk değerini hesaplamak için önce kodunun çözülmesi ve uygunluk fonksiyonu ile değerlendirilmesi gerekir. Çizelge 3.2.de popülasyonu oluşturan dört kromozomun uygunluk değerleri şu şekildedir:

11101→ 29 → f(29)=841
00011 → 3 → f(3)=9
10010 → 18 → f(18)=324
01010 → 10 →f(10)=100

Başlangıç popülasyonunda, birinci kromozomun 841 uygunluk değeri ile en uygun çözüm olduğu açıktır. Uygunluk, yalnızca çözümün ne kadar iyi olduğunu göstermekle kalmaz, aynı zamanda kromozomun optimal olana ne kadar yakın olduğuna da karşılık gelir.

3.4. Genetik Operatörler

Genetik operatör, algoritmayı verilen problemin çözümüne yönlendirmek için genetik algoritmalarda kullanılan operatörlerdir. Algoritmanın başarılı olması için birbiriyle bağlantılı olarak çalışması gereken üç ana operatör türü (seçim, çaprazlama ve mutasyon) vardır. Genetik operatörler, çözümler arasında seçim yapmak (seçim operaörü), mevcut çözümleri yeni çözümlerde birleştirmek (çaprazlama operatörü) ve genetik çeşitliliği oluşturmak ve sürdürmek (mutasyon operatörü) için kullanılır.

3.4.1. Seçim Operatörü

Seçim operatörü, geçiş için bir popülasyondan ebeveynleri seçme prosedürüdür. Kromozomların yapısına karar verdikten sonra, bir sonraki adım, seçimin nasıl yapılacağına, yani popülasyonda gelecek nesil için yavrular yaratacak bireylerin nasıl seçileceğine ve her birinin kaç yavru oluşturacağına karar vermektir. Seçimin amacı, yavrularının daha yüksek uygunluk değerine sahip olması umuduyla popülasyondaki daha sağlıklı bireyleri oluşturmaktır. Kromozomlar, üreme için ebeveyn olmak üzere ilk popülasyondan seçilir. Asıl sorun, bu kromozomların nasıl seçildiğidir. Darwin'in evrim teorisine göre, en iyiler yeni yavrular yaratmak için hayatta kalır. Ebeveynlerden çapraz geçişe kromozomları seçme yöntemleri rulet tekerleği, Boltzmann, turnuva, sıralama, sabit durum, stokastik evrensel seçim gibi yöntemlerdir. En yaygın ikisi rulet çarkı seçimi ve turnuva seçimidir.

3.4.1.1. Rulet Tekerleği Seçimi

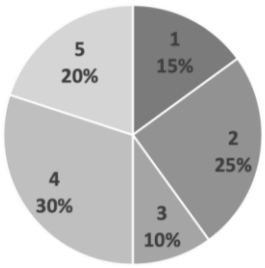
Rulet tekerleği seçiminde bireyler, uygunluk değeriyle orantılı bir prosedür kullanılarak popülasyondan seçilir; bu prosedür, bir rulet tekerleği şeklinde görselleştirilebilir, böylece her bir birey, tekerlekte uygunluk değeriyle orantılı bir dilim kaplar. Bu hayali tekerlek döndürüldüğünde ve sonunda durduğunda, bir bireyin seçilme olasılığı uygunluk değeriyle orantılı olacaktır.

Uygulamada prosedür, aşağıdaki gibi tanımlanan kümülatif bağıl uygunluk değerleri oluşturularak gerçekleştirilir:

$$\varphi_j = \frac{\sum_{i=1}^{j} F_i}{\sum_{i=1}^{N} F_i} \qquad j = 1, 2, \cdots, N$$

Burada F_i fonksiyonu i inci kromozomun uygunluk fonksiyonudur. Daha sonra, r ∈ [0, 1] sayısı rasgele olarak seçilir ve $r < \varphi_j$ değerini karşılayan en küçük j'ye sahip olan kromozom seçilir. Şekil 3.1.de bir rulet tekerleği örneği verilmiştir.

Kromozom	Uygunluk	φ_j
1	3	0,15
2	5	0,40
3	2	0,50
4	6	0,80
5	4	1,00



Şekil 3.1. Bir rulet tekerleği örneği

3.4.1.2. Turnuva Seçimi

Açıkça görülüyor ki, rulet tekerleği seçimi biyolojik açıdan pek makul değil. Doğada çok sık görülen hayvan çiftleri arasındaki tipik yarışmalara daha çok benzeyen alternatif bir prosedür turnuva seçimidir. En basit şekliyle turnuva seçimi, popülasyondan rastgele iki bireyi seçmek ve ardından çiftin en iyi bireyini, yani daha yüksek uygunluk değerine sahip olanı, P_{tur} olasılıkla ile seçmekten oluşur. Bu nedenle, $1 - P_{tur}$ olasılığı ile daha kötü olan birey seçilir. P_{tur}, turnuva seçim parametresi olarak adlandırılır ve tipik olarak 0,7-0,8 civarında değerler alır. Pratikte, turnuva seçimi, [0, 1] aralığında rastgele bir r sayısının düzgün bir şekilde seçilmesi ve $r < P_{tur}$ ise iki bireyden daha iyi olanının seçilmesiyle gerçekleştirilir. Aksi takdirde, daha kötü olan kişi seçilir. Turnuva seçimi ikiden fazla bireyi içerecek şekilde genelleştirilebilir. Genel durumda, popülasyondan rastgele j birey seçilir ve en iyi birey, P_{tur} olasılığı ile seçilir. Bu kişi seçilmezse, sonraki adım 1 prosedürünü kalan $j - 1$ bireyler için tekrar en iyi kişiyi seçme olasılığı P_{tur} ile tekrar etmektir. j, turnuva boyutu olarak adlandırılır.

3.4.2. Elitizm

Elitizm operatörü en iyi ilk kromozomu veya en iyi birkaç kromozomu yeni popülasyona doğrudan kopyalar. Geri kalan kromozomlar klasik bir şekilde seçilerek eşleşirler. Bu tür bireyler, üremeleri için seçilmezlerse veya çaprazlama veya

mutasyon onları yok ederse kaybolabilir. Bu, GA'nın performansını önemli ölçüde artırır.

3.4.3. Çaprazlama Operatörü

Çaprazlama, GA'larda önemli bir operatördür. Ana amacı, genetik materyallerinin parçalarını yeniden birleştirerek rastgele seçilen ana kromozomlar arasında bilgi alışverişi yapmaktır. Olasılıksal olarak gerçekleştirilen bu işlem, iki ana kromozomun parçalarını birleştirerek yeni nesil için yavrular üretir. Çaprazlama yoluyla, başarılı bir birey popülasyonda çok hızlı bir şekilde yayılır, dolayısıyla çeşitliliği azaltır, bu da doğal popülasyonlarda akrabalı çiftleşmeye benzer bir süreçtir. Bu nedenle, çaprazlama aslında biraz fazla verimlidir ve popülasyonun yerel bir optimumda sıkışıp kalmasına neden olabilir. Bu nedenle, çaprazlama oranı olarak adlandırılan, yalnızca belirli bir $P_ç$ olasılığı ile geçiş yapmak yaygın olarak kullanılır. Bu olasılık bize çaprazlama işleminden geçen kromozomların beklenen sayısını $P_ç$ * popülasyon_sayısı olarak verir. Popülasyondaki her bir kromozom için:

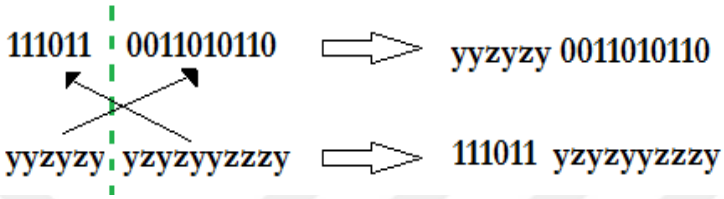
- [0,1] aralığında düzgün dağılımdan rasgele bir r sayısı üretilir.
- r <$P_ç$ ise, çaprazlama için verilen kromozomlar seçilir.

Çaprazlamanın yapılmadığı durumlarda, seçilen iki birey olduğu gibi kopyalanır. Kullanıcı tanımlı çaprazlama oranına sahip bir çaprazlama için seçilen tüm kromozomlar, çeşitli şekillerde rastgele çiftleştirilir. En yaygın olanları tek noktalı, iki noktalı ve düzgün çaprazlamadır.

3.4.3.1. Tek Noktalı Çaprazlama

Tek noktalı çaprazlama, en yaygın kullanılan çaprazlamalardan biridir. Öncelikle, çiftleşme havuzundaki seçili kromozomlar rastgele eşleştirilir. Daha sonra, bir çift üzerinde çaprazlamayı gerçekleştirmek için, çaprazlama noktası olarak

bilinen bir k tam sayısı, 1 ile u-1 arasında rastgele olarak seçilir, burada u, kromozomun uzunluğudur. (k+1) konumundan u'ya kadar tüm genler değiştirerek iki yeni kromozom oluşturulur. Şekil 3.2, rastgele bir çaprazlama noktası ile iki ebeveyn kromozomunun bir çaprazlamasından elde edilen yavruları göstermektedir.



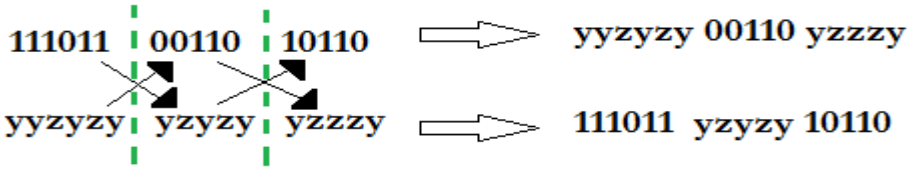
Şekil 3.2. Tek noktalı çaprazlama

Tek noktalı çaprazlamanın bazı eksiklikleri vardır. Bir kromozomun başı ve kuyruğu yavruya birlikte aktarılamaz. Bir kromozomun hem başı hem de kuyruğu iyi genetik bilgi içeriyorsa, doğrudan tek noktadan çapraz geçişle elde edilen yavruların hiçbiri iki iyi özelliği paylaşmayacaktır. İki noktalı çaprazlama kullanmak bu dezavantajı önler ve bu durumda genellikle tek noktalı çaprazlamadan daha iyi kabul edilir.

3.4.3.2. İki Noktalı Çaprazlama

Tek noktalı çaprazlamanın yanı sıra, genellikle birden fazla çaprazlama noktasını içeren birçok farklı çaprazlama algoritması geliştirilmiştir. Daha fazla çaprazlama noktası eklemenin GA'nın performansını düşürdüğüne dikkat edilmelidir. Fazladan çaprazlama noktaları eklemedeki sorun, yapı taşlarının bozulma olasılığının daha yüksek olmasıdır. Bununla birlikte, daha fazla çaprazlama noktasına sahip olmanın avantajı, sorunlu alanın daha derinlemesine araştırılabilmesidir. İki noktalı çaprazlamada, iki çaprazlama noktası seçilir ve bu noktalar arasındaki içerik iki eşleşen ebeveyn arasında değiştirilir.

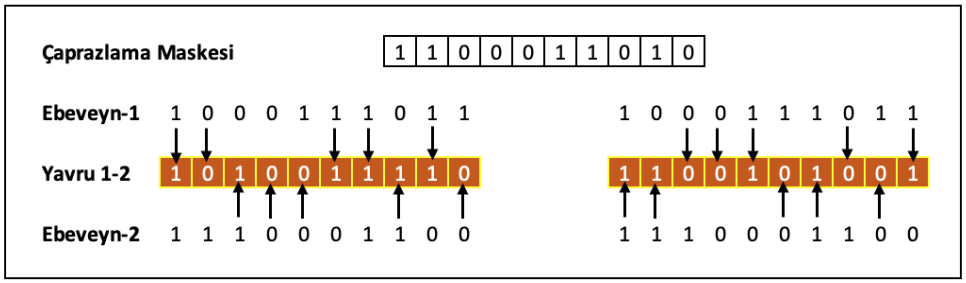
Şekil 3.3, iki noktalı çaprazlamayı göstermektedir. Böylelikle bu noktalar arasındaki içerik, ebeveynler arasında bir sonraki nesilde çiftleşmek için yeni çocuklar üretmek üzere değiş tokuş edilir.



Şekil 3.3. İki noktalı çaprazlama

3.4.3.3. Düzgün Çaprazlama

İlk olarak, ebeveynlerle aynı uzunlukta rastgele bir maske oluşturulur. Yavru-1'i yaratmak için maskedeki bit 1 olduğunda, ebeveyn-1'den karşılık gelen bit yavru-1'e kopyalanır ve maskedeki bit 0 olduğunda, ebeveyn-2'deki karşılık gelen bit, yavru-1'e geçirilir. Her ebeveyn çifti için rastgele yeni bir çaprazlama maskesi oluşturulur. Bu nedenle yavrular, her ebeveynin genlerinin bir karışımını içerir. Etkili çaprazlama noktası sayısı sabit değildir ancak ortalama olarak u/2 olacaktır (burada u, kromozom uzunluğudur). Yavru-2'yi oluşturmak için, sadece ebeveynlerin sırasını değiştirmek yetecektir. Şekil 3.4, düzgün çaprazlamayı göstermektedir.



Şekil 3.4. Düzgün Çaprazlama

Düzgün çaprazlama, segmentler yerine bitleri değiş tokuş ettiğinden, bazı özellikleri göreceli konumlarına bakılmaksızın birleştirebilir. Bu yetenek, yapı taşı çözümlerini yok etmenin dezavantajından daha ağır basabilir ve bazı problemler için düzgün çaprazlamayı üstün hale getirebilir (Syswerda, 1993).

3.4.4. Mutasyon Operatörü

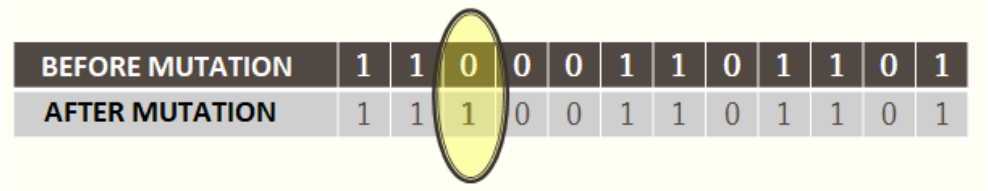
Çaprazlamadan sonra kromozomlar mutasyona tabi tutulur. Mutasyon, algoritmanın yerel minimumda kalmasını engeller. Mutasyon, genetik bilgiyi rastgele bozarak genel aramayı verimli hale getirmede önemli bir rol oynar. Geri dönüşü olmayan genetik materyal kaybına karşı koruma sağlamaktadır.

Mutasyon geleneksel olarak basit bir arama operatörü olarak kabul edilir. Bir çaprazlamanın daha iyi olan çözümleri bulmak için mevcut çözümü kullanması gerekiyorsa, mutasyonun tüm arama alanının araştırılmasına yardımcı olması beklenir. Mutasyon, popülasyondaki genetik çeşitliliği sürdürmek için bir arka plan operatörü olarak görülüyor. Bazı yapı taşlarını rastgele değiştirerek popülasyonda yeni genetik yapılar sunar. Mutasyon, yerel çözümlerin tuzağından kaçmaya yardımcı olur ve popülasyondaki çeşitliliği korur.

Farklı temsil türleri için birçok farklı mutasyon biçimi vardır. Burada, bir kromozomdaki her bitin, mutasyon olasılığı adı verilen (genellikle düşük) bir P_m olasılığı ile mutasyona tabi tutulduğunu varsayıyoruz. Her bitin (tüm popülasyondaki tüm kromozomlarda) mutasyona uğrama şansı eşittir. Çaprazlamadan sonra popülasyondaki her bir kromozom ve kromozom içindeki her bit için:

- [0,1] aralığında düzgün dağılımdan rasgele bir r sayısı üretilir
- Eğer $r<P_m$ ise, ilgili bit değiştirilir

Diğer bir deyişle, Şekil 3.5'te gösterildiği gibi, bir ebeveyn ele alınır ve bazı bitler rasgele değiştirilerek bir mutasyon kromozomu oluşturulur.



Şekil 3.5. Mutasyon opertatörü

3.4.5. Sonlandırma Kriterleri

Yerel bir optimuma ulaşıldığında sona eren basit komşuluk arama yöntemlerinin aksine, GA'lar prensipte sonsuza kadar çalışabilen stokastik arama yöntemleridir. Pratikte bir sonlandırma kriterine ihtiyaç vardır. Aşağıdaki gibi listelenen bir genetik algoritmayı durdurmak için birkaç sonlandırma kriteri vardır:

- Maksimum Nesil Sayısı: Genetik algoritma, belirtilen nesil sayısı geliştirildiğinde durur.
- Geçen Zaman: Genetik süreç, belirli bir süre geçtiğinde sona erecektir. Belirtilen süre geçmeden maksimum nesil sayısına ulaşılmışsa, işlem sona erecektir.
- Uygunluk Değerinin Sabitleşmesi: Belirli bir nesil için popülasyonun en iyi uygunluk değerinde bir değişiklik olmazsa genetik süreç sona erecektir. Maksimum nesil sayısına, herhangi bir değişiklik olmadan belirtilen nesil sayısına ulaşılmadan önce ulaşılmışsa, süreç sona erecektir.

Sonlandırma kriterine erişildiğinde arama durdurulur ve en iyi uygunluk değerine sahip kromozom çözüm için önerilir.

3.4.6. Genetik Algoritma Parametreleri

GA'larda programcı tarafından ayarlanması ve/veya düzeltilmesi gereken birkaç parametre vardır. Bunlardan bazıları popülasyon boyutu, bir kromozomun uzunluğu, çaprazlama ve mutasyon oranları, sonlandırma kriteri ve popülasyon yenileme stratejisidir. Örneğin, tüm popülasyonun yeni popülasyonla değiştirildiği nesil değiştirme stratejisinin mi yoksa sadece daha az uygun kişinin değiştirildiği sabit durum değiştirme politikasının mı kullanılacağına karar verilmesi gerekir. GA'lardaki bu tür parametreler çoğunlukla probleme bağlıdır ve literatürde bunların seçimleri için çalışmalar bulunmamaktadır.

4. KARMA MODELDE GENETİK ALGORİTMA KULLANIMI

Yapılan bu çalışmada R programlama dili kullanılarak ve genetik algoritma tekniklerinden faydalanılarak çeşitli yazılım güvenilirliği veri setleri üzerinde karma modeller oluşturulmaya çalışılmıştır.

Birden fazla dağılımın karmasından oluşan modele karma model denir. Bu çalışmada modeli oluşturan dağılımlar bir dağılım havuzundan algoritma tarafından seçilmiştir. Varsayalım ki 4.1 eşitliğinde verilen $\mathcal{F}$, her biri Λ_i parametre uzayını kullanan n adet dağılımın oluşturduğu bir vektör olsun.

$$\mathcal{F}(\Omega) = \begin{bmatrix} f_1(\Lambda_1) \\ f_2(\Lambda_2) \\ f_3(\Lambda_3) \\ \vdots \\ f_n(\Lambda_n) \end{bmatrix} \tag{4.1}$$

Karma modelin i inci bileşenini oluşturan dağılımı seçmek için 4.2 eşitliğinde verilen T_i vektörü kullanılsın.

$$T_i = [t_1 \quad t_2 \quad t_3 \quad \dots \quad t_j \quad \dots \quad t_n] = \begin{cases} t_j = 1 \ ; & i = j \\ t_j = 0 \ ; & i \neq j \end{cases} \tag{4.2}$$

Karma modelin her bir bileşeni için seçilen dağılımı 4.3 eşitliğinde verilen X vektörü ile sembolize edelim.

$$X = [x_1 \quad x_2 \quad x_3 \quad \dots \quad x_i \quad \dots \quad x_g] \ , x_i \ , g, n \in N \ ; \ 1 \leq x_i \leq n \qquad (4.3)$$

Bu eşitlikte g, karma modeldeki dağılım sayısını; f, dağılım havuzu $\mathcal{F}$'teki dağılım sayısını, x_i ise modeldeki i inci dağılımın $\mathcal{F}$'teki sıra numarasını temsil etmektedir.

Ω_i , karma modelin i inci dağılımının parametre değerleri olduğunu varsayarsak modeldeki tüm parametre değerlerinin oluşturduğu vektör $\Omega = [\Omega_1 \quad \Omega_2 \quad \dots \quad \Omega_g]$ vektörü ve ω_i , karma modeldeki i inci dağılımını karma oranı olduğunu varsayarsak modeldeki tüm karma oranlarının oluşturduğu $\omega = [\omega_1 \quad \omega_2 \quad \dots \quad \omega_g]$ vektörü olmak üzere;

$$f(X; \omega; \Omega) = \sum_{i=1}^{g} \omega_i T_{x_i} \mathcal{F}(\Omega_i)$$

eşitliği modelin genel halini vermektedir.

Modelin genel parametre vektörü $[X; \omega; \Omega]$ vektörüdür. Oluşturulan genetik algoritmada bu vektör kromozom yapısını belirlemektedir ve Şekil 4.1'de gösterilmiştir.

x_1	x_2	...	x_g	ω_1	ω_2	...	ω_g	Ω_1	Ω_2	...	Ω_g

Şekil 4.1. Genetik algoritmada kullanılan kromozom yapısı

Genetik algoritmada, 0,8 çaprazlama olasılığı ve 0,2 mutasyon olasılığı uygulanmış olup uygunluk fonksiyonu olarak gerçek veri ile tahmin arasındaki HKO değeri tercih edilmiştir.

4.1. Çalışmada Kullanılan Veri Setleri

Geliştirilen yazılım, yazılım güvenilirliği çalışmalarını içeren literatürde sıkça kullanılan dört farklı veri seti üzerinde uygulanmıştır. Bu veri setlerinin özellikleri Çizelge 4.1'de gösterilmiştir.

Çizelge 4.1. Kullanılan veri setlerinin özellikleri

Veri Seti	**Açıklama**	**Referans**
Wood	Ürünlerdeki önemli değişiklikleri içeren ve birkaç milyon satır koddan oluşan bir uygulama	(Wood, 1996)
Ohba	Yaklaşık 1.317.000 satırdan oluşan PL/I veri tabanı uygulaması	(Ohba, 1984)
Wu	Satış yazılımı	(Hsu ve Huang, 2014)
Jeske	200.000 satırdan oluşan izleme yazılımı	(Hsu ve Huang, 2014)

4.2. Model Oluşturulurken Kullanılan Dağılımlar

Uygulama sırasında oluşturulacak karma modelde kullanılmak üzere 6 farklı dağılım seçilmiştir. Bu dağılımlar yazılım güvenilirlik analizinde kullanılan dağılımlardır. Kullanılan dağılımlar Çizelge 4.2'de gösterilmiştir.

Çizelge 4.2. Uygulamada kullanılan dağılımlar

Dağılım	**Olasılık yoğunluk fonksiyonu**
Gumbel dağılımı	$f(x)=\frac{1}{\sigma}exp\left\{-\left(\frac{x-\mu}{\sigma}\right)\right\}exp\left[-exp\left\{-\left(\frac{x-\mu}{\sigma}\right)\right\}\right]$; $-\infty<x<\infty$, $\sigma>0$
Gompertz dağılımı	$f(x)=\theta e^{\alpha x}\,exp\left\{\frac{\theta}{\alpha}(1-e^{\alpha x})\right\}$; $x\geq 0$, $\theta>0$, $-\infty<\alpha<\infty$
Genelleştirilmiş üstel dağılım	$f(x)=\alpha\lambda x e^{-\lambda x}\{1-e^{-\lambda x}\}$; $(\alpha,\lambda)>0$, $x>0$
Genelleştirilmiş Weibull dağılımı	$f(x)=\alpha\theta x^{\alpha-1}(1+x^{\alpha})^{\theta-1}exp\{1-(1+x^{\alpha})^{\theta}\}$; $x\geq 0$, $\alpha>0,\theta>0$
Üstel lojistik dağılım	$f(x)=\frac{\alpha}{\beta}exp\left\{-\frac{x}{\beta}\right\}\left\{1+exp\left(-\frac{x}{\beta}\right)\right\}$; $x>0$, $(\alpha,\beta)>0$
Log-gama dağılımı	$f(x)=\alpha\lambda x e^{\lambda x}exp\{-\alpha e^{-\lambda x}\}$; $(\alpha,\lambda)>0$, $x>0$

4.3. Bulgular

4.3.1. Genetik Algoritma ile Dağılım Parametrelerin Belirlenmesi

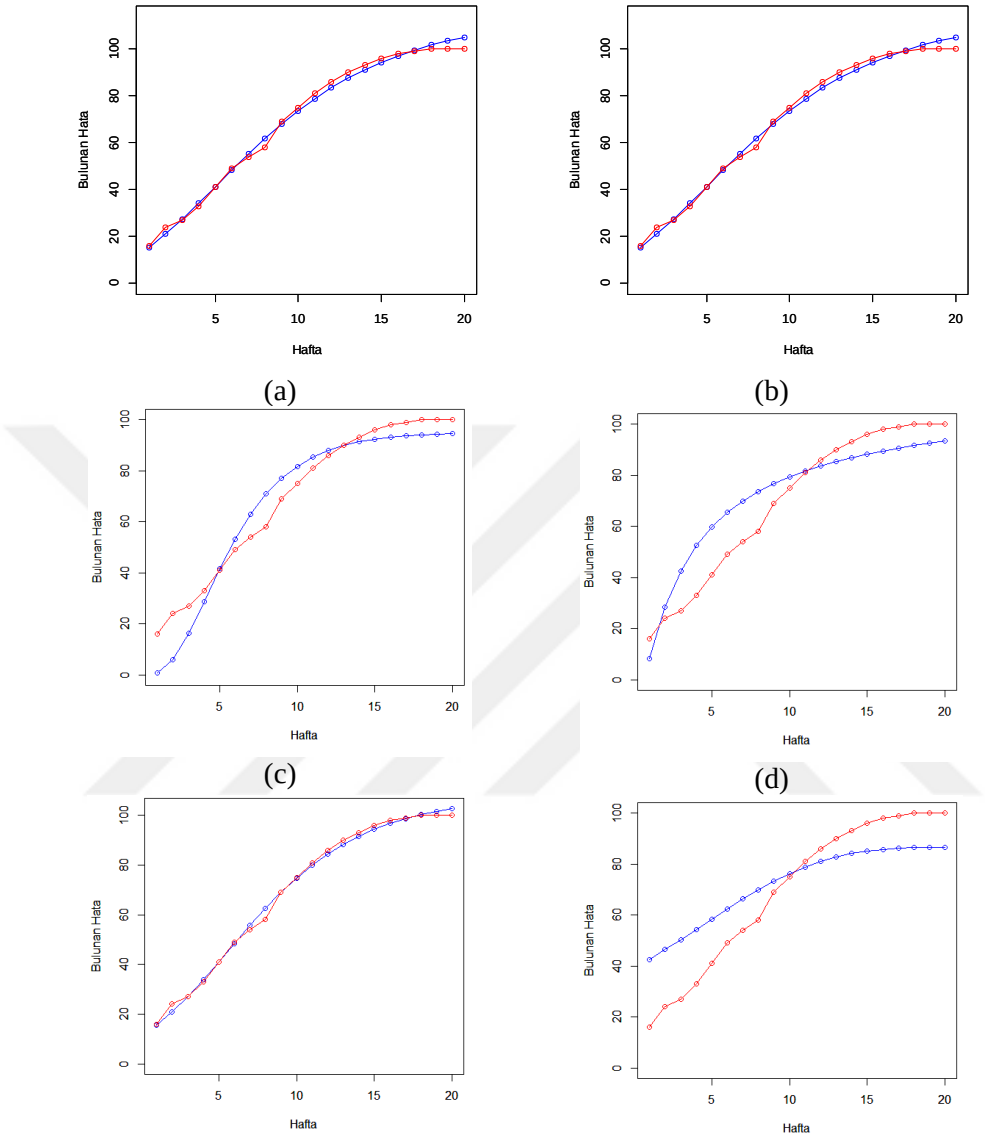
Çalışmanın bu aşamasında her bir dağılım tek başına kullanıldığında genetik algoritma ile en uygun model parametrelerinin belirlenmesi gerçekleştirildi. Bu işlem sırasında uygunluk fonksiyonu olarak HKO değerinin minimuma indirilmesi kullanıldı. Her bir model için bulunan değerler Çizelge 4.3'de gösterilmiştir. Üstel Lojistik ve Gumbel dağılımlarının 4 veri seti için de uygun modeller ürettiği gözlemlenmiştir.

Çizelge 4.3. Her bir dağılım için bulunan en düşük HKO değerleri

Model	**Veri Seti 1 HKO**	**Veri Seti 2 HKO**	**Veri Seti 3 HKO**	**Veri Seti 4 HKO**
Gumbel	4,688	89,791	**35,388**	**8,957**
Gompertz	220,304	2091,095	186,060	748,283
Genelleştirilmiş Üstel	62,313	1185.327	137,911	335,176
Genelleştirilmiş Weibull	117,235	1534,113	119,967	928,905
Üstel Lojistik	**2,836**	**85,657**	36,889	11,084
Log Gama	207,294	1231,989	203,350	1023,053

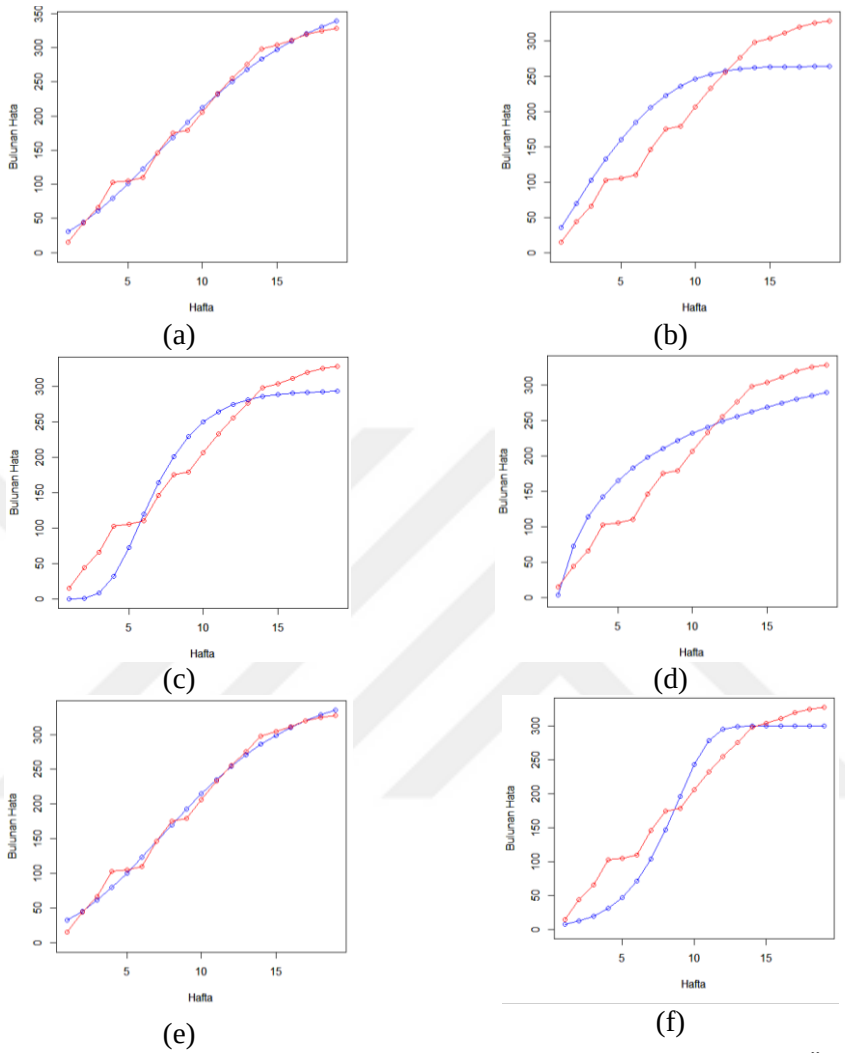
Veri seti-1 için Üstel Lojistik dağılımı ile oluşturulan modelde HKO değeri 2,836 ile diğer modellerden daha küçük çıkmıştır ve en uygun model olarak belirlenmiştir. Bunun yanı sıra Gumbel dağılımı da düşük bir HKO değeri üretmiştir.

Veri Seti-1 için her bir modelde verilerin tahmin verileri ile karşılaştırıldığı grafikler Şekil 4.2'de gösterilmiştir.



Şekil 4.2. Veri Seti-1 için (a) Gumbel, (b) Gompertz, (c) Genelleştirilmiş Üstel, (d) Genelleştirilmiş Weibull, (e) Üstel Lojistik, (f) Log Gama dağılımları kullanarak algoritma tarafından yapılan en iyi tahminler

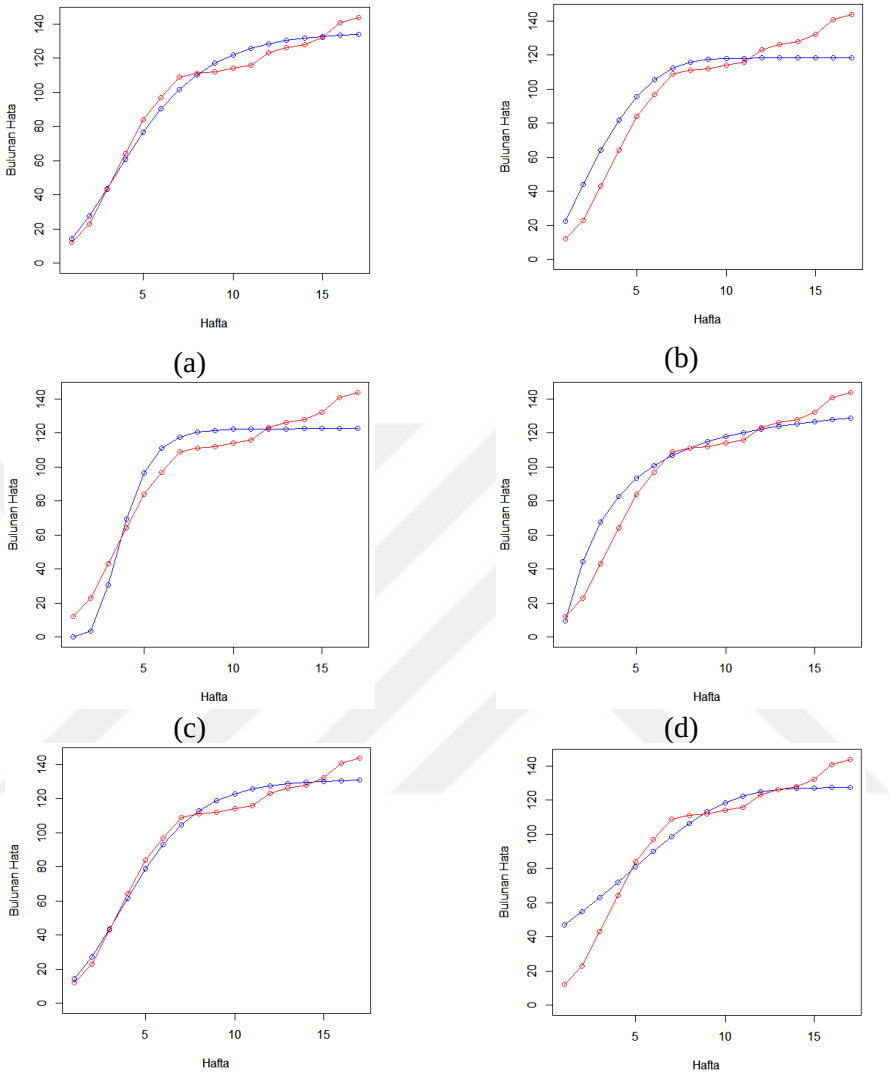
Veri Seti-2 için her bir modelde verilerin tahmin verileri ile karşılaştırıldığı grafikler Şekil 4.3'de gösterilmiştir.



Şekil 4.3. Veri Seti-2 için (a) Gumbel, (b) Gompertz, (c) Genelleştirilmiş Üstel, (d) Genelleştirilmiş Weibull, (e) Üstel Lojistik, (f) Log Gama dağılımları kullanarak algoritma tarafından yapılan en iyi tahminler

Veri seti-2 için Üstel Lojistik dağılımı ile oluşturulan model 85,637 ile en küçük HKO değerini üretmiştir ve en uygun model olarak belirlenmiştir. Yine Gumbel dağılımının kullanıldığı model de düşük bir HKO değeri üretmiştir.

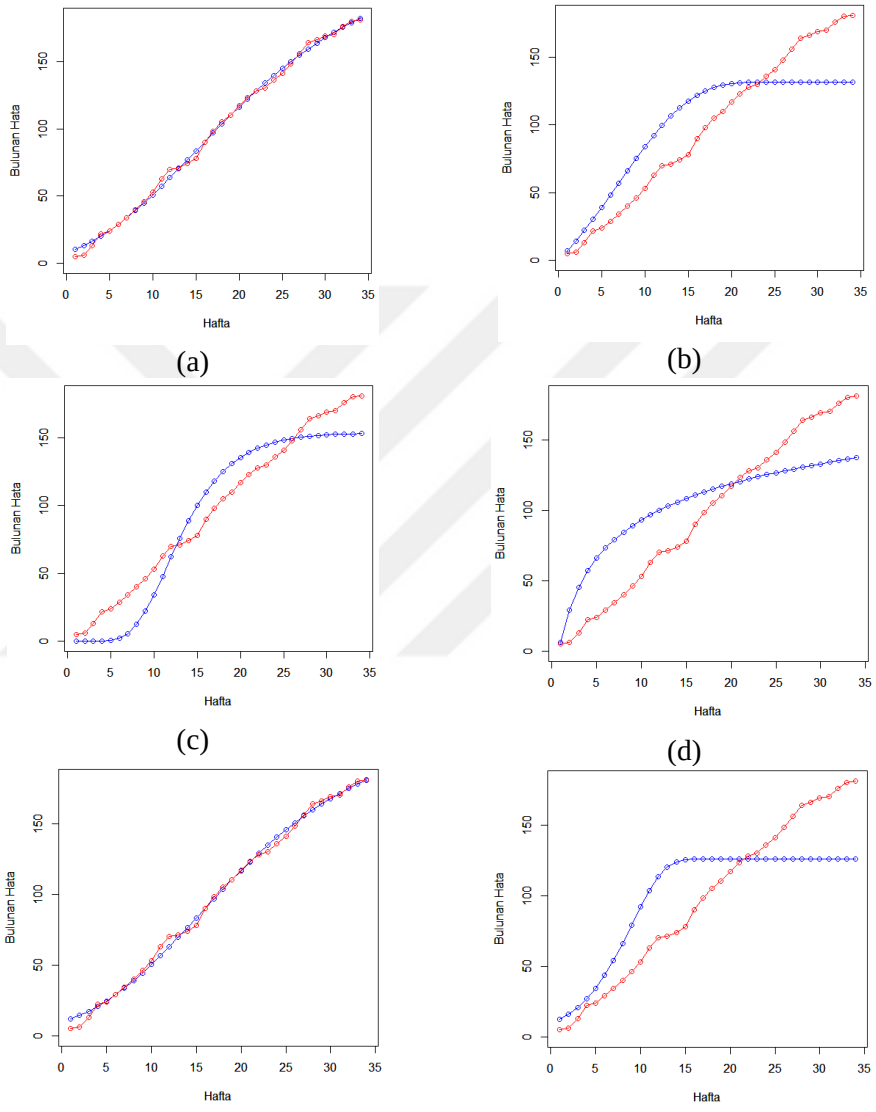
Veri Seti-3 için her bir modelde verilerin tahmin verileri ile karşılaştırıldığı grafikler Şekil 4.4'de gösterilmiştir.



Şekil 4.4. Veri Seti-3 için (a) Gumbel, (b) Gompertz, (c) Genelleştirilmiş Üstel, (d) Genelleştirilmiş Weibull, (e) Üstel Lojistik, (f) Log Gama dağılımları kullanarak algoritma tarafından yapılan en iyi tahminler

Veri Seti-3 için Gumbel dağılımı ile oluşturulan modelde HKO değeri 35,388 ile en küçük HKO değerini üretmiştir ve en uygun model olarak belirlenmiştir. Üstel Lojistik modeli de düşük bir HKO değeri üretmiştir.

Veri Seti-4 için her bir modelde verilerin tahmin verileri ile karşılaştırıldığı grafikler Şekil 4.5'de gösterilmiştir.



Şekil 4.5. Veri Seti-4 (a) Gumbel, (b) Gompertz, (c) Genelleştirilmiş Üstel, (d) Genelleştirilmiş Weibull, (e) Üstel Lojistik, (f) Log Gama dağılımları kullanarak için algoritma tarafından yapılan en iyi tahminler

Veri seti-4 için Gumbel dağılımı ile oluşturulan modelde HKO değeri 8,957 çıkmıştır ve diğer modellere göre daha küçük bir HKO değeri elde edilmiştir. Böylece Gumbel dağılımı kullanılarak elde edilen modelin en uygun model olduğu söylenebilir. Üstel Lojistik dağılımı ile elde edilen model de düşük bir HKO değeri üretmiştir.

4.3.2. İki Dağılımlı Karma Modelde Genetik Algoritma ile Kullanımı

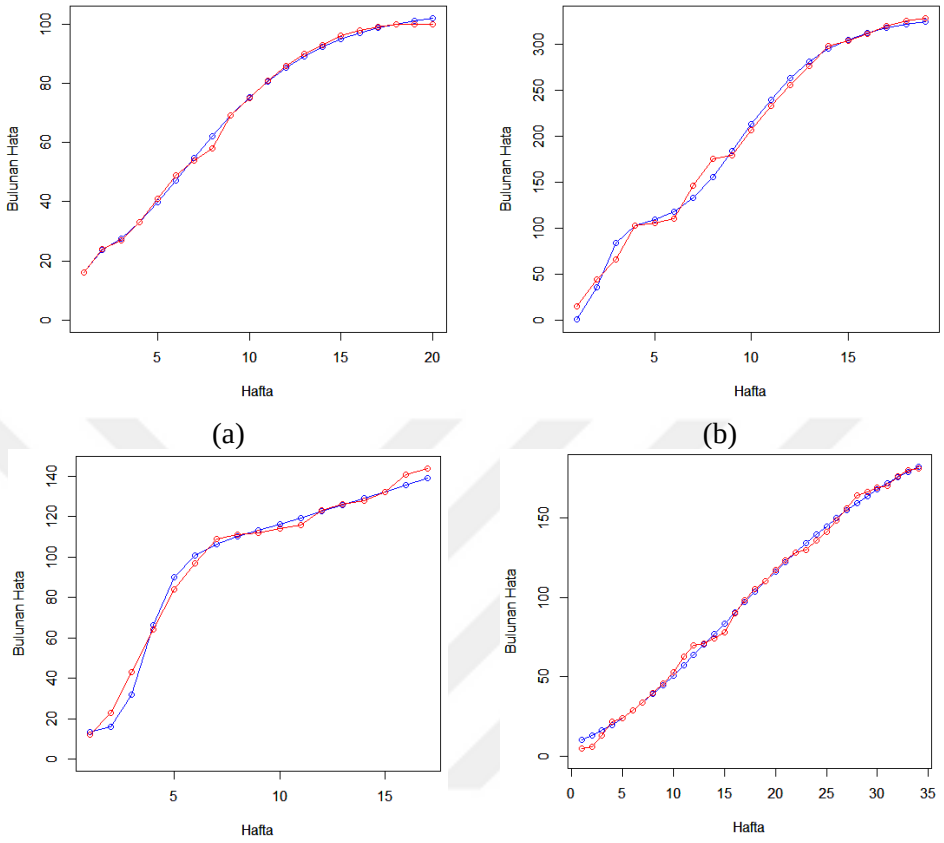
Çalışmanın bu aşamasında genetik algoritma kullanarak karma modelleme ile en iyi model oluşturulmaya çalışıldı. İki dağılımın karma modelinin oluşturulduğu sistemde hem dağılımlar hem de bu dağılımların parametreleri tamamen genetik algoritma ile belirlendi. Bu işlem sırasında uygunluk fonksiyonu olarak HKO değerinin minimuma indirilmesi kullanıldı.

Her veri seti için algoritmanın karma dağılımda kullandığı dağılımlar, bu dağılımların parametreleri ve HKO değerleri Çizelge 4.4'de gösterilmiştir.

Çizelge 4.4. Genetik algoritma ile iki dağılımlı karma model kullanarak bulunan sonuçlar

	HKO	**Dağılım-1**	**Dağılım-2**
Veri Seti-1	1,651	Gumbel	Genelleştirilmiş Üstel
		$\omega_1 = 0{,}815$	$\omega_2 = 0{,}185$
		$\mu = 6{,}508$	$\alpha = 98{,}897$
		$\sigma = 4{,}117$	$\lambda = 5{,}742$
Veri Seti-2	78,311	Üstel Lojistik	Genelleştirilmiş Üstel
		$\omega_1 = 0{,}670$	$\omega_2 = 0{,}330$
		$\alpha = 30{,}743$	$\alpha = 19{,}071$
		$\beta = 2{,}708$	$\lambda = 1{,}429$
Veri Seti-3	18,273	Gumbel	Genelleştirilmiş Üstel
		$\omega_1 = 0{,}675$	$\omega_2 = 0{,}325$
		$\mu = 17{,}780$	$\alpha = 51{,}972$
		$\sigma = 18{,}249$	$\lambda = 1{,}142$
Veri Seti-4	8,759	Gumbel	Üstel Lojistik
		$\omega_1 = 0{,}157$	$\omega_2 = 0{,}843$
		$\mu = 55{,}549$	$\alpha = 14{,}511$
		$\sigma = 32{,}422$	$\beta = 12{,}053$

Her bir veri seti, tahmin verileri ile karşılaştırıldığında elde edilen grafikler Şekil 4.6'de gösterilmiştir.



Şekil 4.6. İki dağılımlı karma modellemede, (a) veri seti-1, (b) veri seti-2, (c) veri seti-3, (d) veri seti-4 için algoritma tarafından yapılan en iyi tahminler

Tek dağılım içeren modellerde tüm veri setleri için Gumbel ve Üstel Lojistik dağılımları diğer dağılımlara göre daha iyi performans göstermesine rağmen karma model kullanıldığında Veri Seti-4 haricinde diğer veri setleri için modeli oluşturan dağılımlardan birinin kötü performans gösteren bir dağılım olması ilginçtir. Bu durumun sebebi olarak iyi performans gösteren dağılımın açıklayamadığı özelliklerin diğer bir dağılım tarafından açıklanabilmesi olduğu düşünülmektedir.

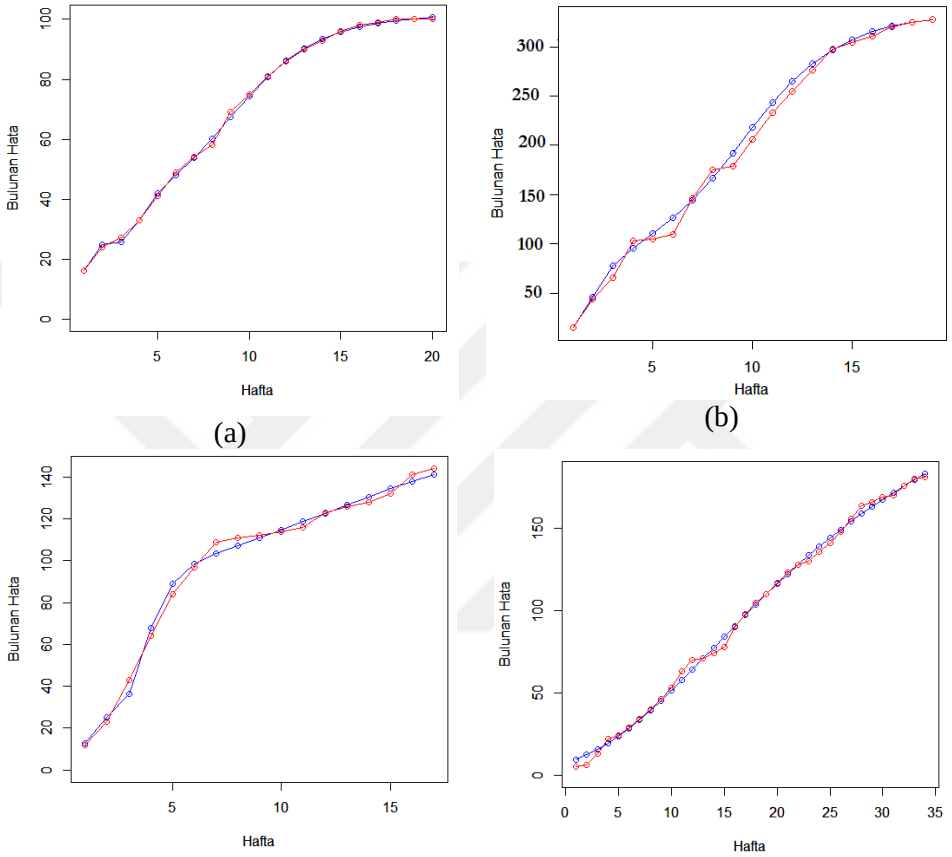
4.3.3. Üç dağılımlı Karma Modelde Genetik Algoritma Kullanımı

Çalışmanın bu aşamasında genetik algoritma kullanarak üç dağılımlı karma modelinin oluşturulduğu bir sistemde hem dağılımlar hem de bu dağılımların parametreleri tamamen genetik algoritma ile belirlendi. Bu işlem sırasında uygunluk fonksiyonu olarak HKO değerinin minimuma indirilmesi kullanıldı. Her veri seti için algoritmanın karma modelde kullandığı dağılımlar, bu dağılımların parametreleri ve HKO değerleri Çizelge 4.5'de gösterilmiştir.

Çizelge 4.5. Genetik algoritma ile üç dağılımlı karma modelde bulunan sonuçlar

	HKO	**Dağılım-1**	**Dağılım-2**	**Dağılım-3**
Veri Seti-1	0,655	Genelleştirilmiş Üstel	Üstel Lojistik	Genelleştirilmiş Üstel
		$\omega_1 = 0{,}244$	$\omega_2 = 0{,}531$	$\omega_3 = 0{,}225$
		$\alpha = 99{,}743$	$\alpha = 28{,}170$	$\alpha = 91{,}919$
		$\lambda = 5{,}440$	$\beta = 2{,}715$	$\lambda = 1{,}121$
Veri Seti-2	39,680	Genelleştirilmiş Üstel	Üstel Lojistik	Genelleştirilmiş Üstel
		$\omega_1 = 0{,}208$	$\omega_2 = 0{,}671$	$\omega_3 = 0{,}121$
		$\alpha = 99{,}661$	$\alpha = 25{,}047$	$\alpha = 73{,}211$
		$\lambda = 1{,}573$	$\beta = 2{,}893$	$\lambda = 4{,}322$
Veri Seti-3	10,174	Genelleştirilmiş Üstel	Genelleştirilmiş Üstel	Gumbel
		$\omega_1 = 0{,}409$	$\omega_2 = 0{,}140$	$\omega_3 = 0{,}451$
		$\alpha = 78{,}209$	$\alpha = 99{,}716$	$\mu = 11{,}930$
		$\lambda = 1{,}236$	$\lambda = 4{,}978$	$\sigma = 7{,}100$
Veri Seti-4	8,045	Üstel Lojistik	Gumbel	Gumbel
		$\omega_1 = 0{,}186$	$\omega_2 = 0{,}361$	$\omega_3 = 0{,}453$
		$\alpha = 92{,}923$	$\mu = 28{,}658$	$\mu = 12{,}087$
		$\beta = 90{,}486$	$\sigma = 19{,}640$	$\sigma = 10{,}214$

Her bir veri setinin en iyi üç dağılım içeren karma modeller ile karşılaştırıldığı grafikler Şekil 4.7'de gösterilmiştir.



Şekil 4.7.Üç dağılımlı karma modellemede, (a) veri seti-1, (b) veri seti-2, (c) veri seti-3, (d) veri seti-4 için algoritma tarafından yapılan en iyi tahminler

Üç dağılım içeren karma modellerde Veri Seti-4 için HKO değerindeki azalma görece düşük seviyede kalırken diğer üç veri setinde HKO değerleri ciddi bir şekilde düşmüştür.

5. SAĞDAN SANSÜRLÜ VERİLERDE GENETİK ALGORİTMA KULLANIMI

5.1. Giriş

Sistem güvenilirliğinde, sistemler farklı yapılara ve özelliklere sahip bileşenlerden oluşurlar. Sistemi oluşturan bileşenlerin güvenilirliklerinden yola çıkarak sistemin güvenilirliği elde edilebilir. Fakat birçok durumda sistemin yaşam süresi gözlemlenebilirken sistemin bileşenlerinin yaşam süreleri gözlemlenemeyebilir. Zaman ve bütçe kısıtlamaları nedeniyle yaşam testi deneylerinde sansür yaygındır (Yang ve ark., 2016). Sağdan sansürlü sistemlerde çeşitli istatistiksel dağılımlar kullanılarak oluşturulan yaşam süresi modellerinin parametrelerini tahmin etmek için en iyi doğrusal yansız tahmin (BLUE) (Balakrishnan ve ark. 2011), maksimum olabilirlik tahmini (MLE) (Balakrishnan ve ark. 2011; Ng ve ark. 2012) ve regresyona dayalı yöntem (Zhang ve ark. 2015) geliştirilmiştir. Bu yöntemler karmaşık hesaplamaları gerektirmektedirler.

Sistem güvenilirliği modellemesinde karma dağılımlar da sıklıkla kullanılır. Berchtold (2004), karma modellerde parametre tahmini için beklenti-maksimizasyon (EM), stokastik beklenti-maksimizasyon (SEM) ve genetik algoritma yöntemlerini karşılaştırmıştır. Volterman ve Balakrishnan (2013), sağdan sansürlü verilerde karma oranlarının hesaplanması için iteratif bir yöntem önermiştir. Wang (2014), Weibull dağılımının parametrelerini sansürlenmiş verilerle tahmin etmek için bare-bones parçacık sürüsü optimizasyonu (BBPSO) algoritması önermiş ve BBPSO ve EM modellerinin performansını Monte Carlo simülasyonu ile karşılaştırmıştır. Wang ve Huang (2014), iki Weibull karışımının parametrelerini tam ve çapraz sansürlü verilerle tahmin etmek için bir parçacık sürüsü optimizasyonu (PSO) algoritması kullandı. PSO ve EM modellerinin performansını bir Monte Carlo simülasyonu ile karşılaştırdılar. Yang ve ark. (2016), tam ve sansürlü sistemlerde, sistemin yaşam süresi modelinin parametrelerinin maksimum olabilirlik tahminlerini elde etmek için SEM algoritmasının kullanılmasını önerdiler. Bordes ve Chauveau

(2016), sağdan sansürlü yaşam süresi verileri için, tanımlanabilir olmaları koşuluyla, parametrik veya semiparametrik karma modellerini tahmin etmemizi sağlayan, EM ve SEM algoritmasına dayanan birkaç yinelemeli yöntem sunmuştur. Ruhi ve ark. (2015) iki katlı Weibull karma modeli için EM algoritması kullanarak modelin parametrelerini tahmin etmeye çalışmıştır.

Weibull Olasılık Kâğıdı (WPP) yöntemi, Weibull karma modellerinin parametrelerini tahmin etmek için yaygın olarak kullanılan grafiksel bir yöntemdir. Murthy ve ark. (2004)'na göre WPP yönteminde iki ciddi dezavantaj vardır. WPP yöntemi, yineleme kullanılmadığı ve görsel olarak değerlendirildiği için çok kaba tahminler üretir. Bu nedenle, daha karmaşık istatistiksel yöntemler için bir başlangıç noktası olarak kullanılabilirler. İkinci dezavantaj, tahmin edilen parametreler için istatistiksel güven limitleri uygulamamasıdır.

Weibull karma modellerinin parametrelerini tahmin etmenin başka bir yöntemi EM'dir. Ferreira ve Silva'ya (2017) göre, EM algoritmasının diğer yinelemeli algoritmalara göre bazı avantajları vardır, ancak aynı zamanda dezavantajları da vardır. EM algoritması, bazı veri kümelerinde veya çok fazla eksik bilgi içeren veri kümelerinde sonuca çok yavaş yaklaşabilir. EM algoritması, parametre tahminlerinin kovaryans matrisinin bir tahminini oluşturmak için entegre bir sürece sahip değildir. Ancak bu dezavantaj, uygun yöntemler kullanılarak giderilebilir. EM algoritmasının bir başka dezavantajı, tahminin, algoritmaya verilen ilk çözüme oldukça bağımlı olmasıdır. Bu nedenle, EM algoritması genel çözüme yakınsamayı garanti etmez. Bununla birlikte, GA, yerel çözümlerden kaçınmak için çözümü her yinelemede belirli bir oranda değiştirir. Böylelikle çözüm uzayının diğer kısımlarında aramaya devam ederek genel sonuca yaklaşır.

Bu çalışmada sağdan sansürlü güvenilirlik verisine en uygun iki katlı Weibull karma modelinin parametrelerini tahmin edebilmek için genetik algoritmaya dayanan bir yöntem sunulmuştur. Bölüm 5.2'de iki katlı Weibull modeli tanımlanmıştır. Ayrıca parametre tahmini için en çok olabilirlik tahmin yöntemi anlatılmıştır. Bölüm 5.3'de genetik algoritmaya dayanan bir yöntem önerilmiş ve ön

cam arıza verisi kullanılarak sonuçlar bilinen bazı yöntemler ile karşılaştırılmıştır. Bölüm 5.4'te farklı durumlarda yöntemin performansını değerlendirmek için Monte Carlo simülasyonu çalıştırılarak sonuçlar Bölüm 5.5'te verilmiştir.

5.2. İki Katlı Weibull Dağılımı

Weibull dağılımından alınan bir T rasgele değişkeninin olasılık yoğunluk fonksiyonu, kümülatif dağılım fonksiyonu, yaşam fonksiyonu ve risk fonksiyonu,

$$f(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} e^{-\left(\frac{t}{\alpha}\right)^{\beta}} \quad , \quad t \geq 0, \quad \alpha > 0, \quad \beta > 0$$

$$F(t) = 1 - e^{-\left(\frac{t}{\alpha}\right)^{\beta}} \quad , \; t \geq 0, \; \alpha > 0, \; \beta > 0$$

$$S(t) = e^{-\left(\frac{t}{\alpha}\right)^{\beta}} \quad , \quad t \geq 0, \quad \alpha > 0, \quad \beta > 0$$

$$h(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} \quad , \; t \geq 0, \; \alpha > 0, \; \beta > 0$$

olarak verilmiştir.

Gözlemleri D, sansürsüz gözlemlerin kümesi ($t_i \leq v_i$) ve C ise sansürlü verilerin kümesi ($t_i > v_i$) olmak üzere iki kümeye ayıralım. En çok olabilirlik fonksiyonu

$$L(\alpha, \beta) = [\prod_{i \in D} f(t_i)]\left[\prod_{i \in C}\left(1 - F(v_i)\right)\right] \tag{5.1}$$

Burada t_i sansürsüz gözlemin zamanını, v_i ise T_i için sansür zamanını göstermektedir. $u_i = min(t_i, v_i)$ olmak üzere β 'nın en çok olabilirlik tahmini,

$$\frac{\sum_{i=1}^{n} u_i^{\hat{\beta}} \ln u_i}{\sum_{i=1}^{n} u_i^{\hat{\beta}}} - \frac{1}{\hat{\beta}} - \frac{1}{k}\sum_{i\in D} \ln u_i = 0$$

eşitliğinin çözülmesi ile elde edilir (Leemis, 1995). α'nın en çok olabilirlik tahmini ise,

$$\hat{\alpha} = \left[\frac{1}{n}\left(\sum_{i=1}^{n} u_i^{\hat{\beta}}\right)\right]^{1/\hat{\beta}}$$

eşitliği ile bulunur.

k katlı bir karma modelin kümülatif dağılım fonksiyonu;

$$G(t) = \sum_{j=1}^{k} \omega_j F_j(t)$$

eşitliği ile gösterilebilir. Burada $F_j(t)$, j inci alt popülasyonun kümülatif dağılım fonksiyonu, ω_j , j inci alt popülasyonun karma dağılım oranı ve $\sum_{j=1}^{k} \omega_j = 1$ dir.

k katlı bir karma modelin olasılık yoğunluk fonksiyonu;

$$g(t) = \sum_{j=1}^{k} \omega_j f_j(t)$$

k katlı bir karma modelin güvenilirlik fonksiyonu ise

$$R_j(t) = 1 - F_j(t)$$

olarak verilmiştir.

Özel olarak iki katlı Weibull karma modelinde (k=2) kümülatif dağılım fonksiyonu;

$$G(t) = \omega F_1(t) + (1-\omega)F_2(t)$$

şeklindedir. $F_1(t)$ yerine $Weibull(\alpha_1, \beta_1)$ ve $F_2(t)$ yerine $Weibull(\alpha_2, \beta_2)$ dağılımlarını koyarsak,

$$G(t) = \left\{1 - exp\left[-\left(\frac{t}{\alpha_2}\right)^{\beta_2}\right]\right\} + \omega\left\{exp\left[-\left(\frac{t}{\alpha_2}\right)^{\beta_2}\right] - exp\left[-\left(\frac{t}{\alpha_1}\right)^{\beta_1}\right]\right\}$$

İki katlı Weibull karma modelinde olasılık yoğunluk fonksiyonu;

$$g(t) = \omega\left\{\frac{\beta_1}{\alpha_1}\left(\frac{t}{\alpha_1}\right)^{\beta_1-1} exp\left[-\left(\frac{t}{\alpha_1}\right)^{\beta_1}\right]\right\} + (1-\omega)\left\{\frac{\beta_2}{\alpha_2}\left(\frac{t}{\alpha_2}\right)^{\beta_2-1} exp\left[-\left(\frac{t}{\alpha_2}\right)^{\beta_2}\right]\right\}$$

Güvenilirlik fonksiyonu;

$$R(t) = \omega\left\{exp\left[-\left(\frac{t}{\alpha_1}\right)^{\beta_1}\right]\right\} + (1-\omega)\left\{exp\left[-\left(\frac{t}{\alpha_2}\right)^{\beta_2}\right]\right\}$$

olarak verilir.

5.1 eşitliğinde iki katlı Weibull dağılımının olasılık yoğunluk fonksiyonunu ve kümülatif dağılım fonksiyonunu yerine koyup her iki tarafın logaritmasını alırsak en çok olabilirlik fonksiyonu,

$$\ln L = \sum_{i\in D} \ln\left\{\omega\left[\frac{\beta_1}{\alpha_1}\left(\frac{u_i}{\alpha_1}\right)^{\beta_1-1} exp\left[-\left(\frac{u_i}{\alpha_1}\right)^{\beta_1}\right]\right] + (1-\omega)\left[exp\left[-\left(\frac{u_i}{\alpha_2}\right)^{\beta_2}\right]\right]\right\} +$$
$$\sum_{i\in C} \ln\left\{\omega\left[exp\left[-\left(\frac{u_i}{\alpha_1}\right)^{\beta_1}\right]\right] + (1-\omega)\left[exp\left[-\left(\frac{u_i}{\alpha_2}\right)^{\beta_2}\right]\right]\right\} \quad (5.2)$$

eşitliği ile bulunur. Parametrelerin en çok olabilirlik tahminleri 5.2 eşitliğinin sırasıyla $\beta_1, \alpha_1, \beta_2, \alpha_2 \; ve \; \omega$ parametrelerine göre kısmi türevlerinin alınıp sıfıra eşitlenmesi ile bulunur. Elde edilen eşitliklerden kapalı form çözümler elde

edilmediği için parametrelerin en çok olabilirlik tahminlerini bulmak için yinelemeli yöntemler kullanmak gerekir.

Murthy ve ark. (2004) veriler için iki katlı bir Weibull karma modeli kurmuş ve ayrıca WPP yöntemine dayalı bir grafiksel yöntem uygulayarak model parametrelerini hesaplamıştır. Model seçimi bir deneme yanılma süreci eğilimindedir. WPP, modellerden birinin belirli bir veri setini modellemek için uygun olup olmadığını belirlemek için sistematik bir prosedür sağlar.

$$y = \ln\{-\ln[1 - F(t)]\} \qquad \text{ve} \qquad x = \ln(t)$$

şeklindeki Weibull dönüşümlerine dayanır.

Bu dönüşümde x değişkenine karşı y'nin grafiğine Weibull olasılık grafiği denir. 1970'lerin başında bu dönüşüme ait verileri çizmek için özel bir kâğıt geliştirildi ve Weibull olasılık kâğıdı (WPP) denildi. Bu kâğıt yardımıyla çizilen grafikler de WPP grafiği olarak adlandırıldı. Günümüzde, birçok güvenilirlik yazılım paketi, verilen bir veri seti için otomatik olarak bu grafiği üreten programlar içermektedir.

Grafik yöntem, Weibull karma modelinin parametrelerini tahmin etmek için yaygın şekilde kullanılmıştır. Mevcut literatürün büyük kısmı, iyi ayrılmış alt popülasyon durumlarıyla ilgilenmekte, çizimlerde ve asimtotların karakterizasyonları için çeşitli yaklaşımlar kullanmaktadır. Murthy ve ark. (2004) da belirttiği gibi grafiksel yöntemde iki ciddi dezavantaj vardır. Grafiksel yöntem, tekrarlanan yineleme uygulanmadığı sürece çok kaba tahminler üretirler ve görme duyusu tarafından değerlendirilirler. Bu nedenle, daha sofistike istatistiksel yöntemlerin başlangıç noktası olarak kullanılabilirler. İkinci dezavantajı ise tahmini parametreler için istatistiksel güven sınırlamaları yapmazlar.

5.3. Önerilen Yöntem ve Uygulama

Yöntemi test etmek için ön cam arıza verisi kullanıldı. Veri seti ilk olarak Blischke ve Murthy'nin (2000) çalışmalarında kullanılmış olup belirli bir aircraft uçağı modelinin ön camında oluşan arıza sürelerini ve servis sürelerini içermektedir. Servis sürelerini içeren veriler tamamlanmamıştır, çünkü başarısızlık süreleri henüz gözlemlenmemiştir. Veri setinde başarısızlık tespit edilemeyen servis süreleri sağdan sansürlü veri olarak kabul edilmiştir.

Çizelge 5.1'de gösterilen veri setinde 88 gözlem arıza zamanını, 65 gözlem arızalanmış ön camların servis saatlerini belirtmekte olup toplam 153 veri içermektedir. Süre ölçüm birimi 1000 saattir.

Çizelge 5.1. Aircraft uçağı ön cam arıza verileri [a]

Arıza Zamanları					**Servis Zamanları**		
0.040	1.866	2.385	3.443		0.046	1.436	2.592
0.301	1.876	2.481	3.467		0.140	1.492	2.600
0.309	1.899	2.610	3.478		0.150	1.580	2.670
0.557	1.911	2.625	3.578		0.248	1.719	2.717
0.943	1.912	2.632	3.595		0.280	1.794	2.819
1.070	1.914	2.646	3.699		0.313	1.915	2.820
1.124	1.981	2.661	3.779		0.389	1.920	2.878
1.248	2.010	2.688	3.924		0.487	1.963	2.950
1.281	2.038	2.823	4.035		0.622	1.978	3.003
1.281	2.085	2.890	4.121		0.900	2.053	3.102
1.303	2.089	2.902	4.167		0.952	2.065	3.304
1.432	2.097	2.934	4.240		0.996	2.117	3.483
1.480	2.135	2.962	4.255		1.003	2.137	3.500
1.505	2.154	2.964	4.278		1.010	2.141	3.622
1.506	2.190	3.000	4.305		1.085	2.163	3.665
1.568	2.194	3.103	4.376		1.092	2.183	3.695
1.615	2.223	3.114	4.449		1.152	2.240	4.015
1.619	2.224	3.117	4.485		1.183	2.341	4.628
1.652	2.229	3.166	4.570		1.244	2.435	4.806
1.652	2.300	3.344	4.602		1.249	2.464	4.881
1.757	2.324	3.376	4.663		1.262	2.543	5.140
1.795	2.349	3.385	4.694		1.360	2.560	

[a] Arıza zamanları ve servis zamanları (arıza olmadığı durumlar) ; zaman birimi 1000 saattir.

Non-parametrik bir yöntem olan Kaplan-Meier yöntemi ile (Zhou ve Yang, 2015) gözlemlerden güvenilirlik fonksiyonu tahmin edildi. Bu tahmini $\hat{R}_{KM}$ olarak isimlendirelim. $\hat{R}_{KM}$ ile en uyumlu iki katlı Weibull karma modeline erişmek için modelin parametreleri genetik algoritma kullanılarak optimize edildi ve $\hat{R}_{GA}$ tahminleri bulundu.

En uygun modele erişmek için, aşağıdaki üç mesafe metriği olan 5.3 – 5.5 eşitlikleri uygunluk fonksiyonu olarak kullanılır. Hata kareler ortalaması (HKO), ortalama mutlak hata (MHO) ve ortalama mutlak yüzde hatası (MYHO) sırasıyla,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{R}_{KM}^{(i)} - \hat{R}_{GA}^{(i)}\right)^2 \tag{5.3}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{R}_{KM}^{(i)} - \hat{R}_{GA}^{(i)}\right| \tag{5.4}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{R}_{KM}^{(i)} - \hat{R}_{GA}^{(i)}}{\hat{R}_{KM}^{(i)}}\right| \tag{5.5}$$

eşitlikleri ile tanımlanır. Uygunluk fonksiyonlarının minimum olması amaçlanmaktadır. Üç farklı uygunluk fonksiyonu kullanılarak oluşturulan yöntemler sırasıyla GA_{MSE}, GA_{MAE} ve GA_{MAPE} olarak isimlendirilsin.

Veri ile bulunan modelin uyumluluğunu test etmek için Anderson-Darling testi kullanıldı. Fischer ve Kamps (2013) sağdan sansürlü verilerde uyumun iyiliği için kullanılan farklı testlerin gücünü karşılaştırmış ve Anderson-Darling testi çalışmada birçok durumda en iyiler arasında yer almıştır. Anderson-Darling testi, sıfır hipotezinde belirtilen kümülatif dağılım fonksiyonu ile ampirik dağılım fonksiyonu arasındaki farka dayanır (Razali ve Yap, 2011). Test istatistiği,

$$A^2 = -\frac{1}{n}\sum_{i=1}^{n}(2i-1)\{\ln[p_{(i)}] + \ln[1-p_{(n-i+1)}]\} - n$$

eşitliği ile verilir. Burada $p_{(i)} = F([x_i - \bar{x}]/s)$ olup F, hipotezde verilen kümülatif dağılım fonksiyonu ve $\bar{x}$ ile s, sırasıyla verinin ortalaması ve standart dağılımıdır. Test sonucu elde edilen Anderson-Darling değerinin (A^2) küçük olması veri ile modelin daha uyumlu olduğunu gösterir.

R programlama dili (R Core Team, 2018) ve GA paketi (Scrucca, 2013) kullanılarak oluşturulan yazılımda GA'nın hiper parametreleri Çizelge 5.2'de verilmiştir.

Çizelge 5.2. Genetik Algoritma Parametreleri

Kromozom yapısı	$[\omega, \alpha_1, \beta_1, \alpha_2, \beta_2]$
Popülasyon sayısı	100
İterasyon sayısı	200
Seçme Yöntemi	Rulet Tekerleği
Elitizm	5
Mutasyon Oranı	0,1
Uygunluk Fonksiyonu	min(HKO) min(MHO) min(MYHO)

200 iterasyon sonucunda genetik algoritma ile elde edilen tahmin, WPP yöntemi ve EM yöntemi ile karşılaştırılmış ve sonuçlar Çizelge 5.3'de gösterilmiştir. WPP yönteminde Murthy ve ark.nın (2004) çalışmalarından elde edilen sonuçlar kullanılmıştır. EM yönteminde ise Ruhi ve ark. (2015)'nın çalışmalarında kullandıkları yazılım ile sonuçlar elde edilmiştir. Başlangıç değerleri hem GA hem de EM yöntemlerinde rastgele alınmıştır.

Çizelge 5.3. GA yöntemi ile literatürdeki diğer yöntemlerin karşılaştırılması

		WPP	EM	Önerilen Yöntemler		
				GA_{HKO}	GA_{MHO}	GA_{MYHO}
Eğitim	A^2	1,1910	**0,4093**	0,7112	0,6177	0,7731
	p-value	0,2705	0,8415	0,5711	0,6392	0,5260
Test	A^2	-	1,3596	**0,9050**	0,9758	0,9286
	p-value	-	0,2801	0,5179	0,5098	0,5297
Model	$\hat{\omega}$	0,136	0,0188	0,0004	0,1704	0,1926
	$\hat{\alpha}_1$	8,230	0,2568	4,1248	4,2835	4,0261
	$\hat{\beta}_1$	0,429	1,2709	3,9644	3,2830	5,2992
	$\hat{\alpha}_2$	3,210	3,5138	3,3261	3,3204	3,3131
	$\hat{\beta}_2$	2,990	2,8405	2,3786	2,5441	2,2846
Modelin Performansı	HKO	0,00133	0,00123	0,00108	0,00086	**0,00084**
	MHO	0,03074	0,02704	0,02896	**0,02207**	0,02374
	MYHO	0,11441	**0,09688**	0,10674	0,10828	0,09949

EM yöntemi uygulandığında eğitim aşaması için A^2 değeri 0,4093 olarak elde edilmektedir. Bu değer diğer yöntemlerde bulunan A^2 değerlerinden daha düşük olduğu için EM yöntemi eğitim aşamasında en iyi performansı göstermiştir. Öte yandan, test aşaması için GA-HKO yönteminde en düşük A^2 değeri 0,9050 olarak bulunmuştur. EM yönteminin performansının test aşamasını önemli ölçüde düşürdüğü gözlemlenmiştir. Bu da GA yöntemleriyle elde edilen modellerin yeni verilerle daha uygun olduğunu göstermektedir.

Ayrıca, bu çalışmada elde edilen modeller HKO, MHO ve MYHO ölçümleri kullanılarak karşılaştırılmıştır. GA-MYHO yöntemi ile elde edilen model en düşük HKO değerini, GA-MHO yöntemi ile elde edilen model ise en düşük MHO değerini üretmiştir. Buna karşılık, en düşük MYHO değeri, EM yöntemi kullanılan model ile elde edilmiştir.

5.4. Simulasyon Çalışması ve Sonuçları

Bu bölümde genetik algoritma kullanılarak önerilen yöntemlerin performanslarını değerlendirmek için simülasyon çalışması yapılmıştır. Simülasyonlar farklı örneklem büyüklükleri, farklı sansür oranları ve farklı karma oranları için gerçekleştirilmiştir. Simülasyonda iki katlı Weibull dağılımına uygun veri aşağıdaki tanımlamalara göre üretilmiştir.

$$F_{ij} \sim \text{Weibull}(\alpha_j, \beta_j) \qquad i = 1, 2, \cdots, n \;\; ; \;\; j = 1, 2$$

$$x_i = \omega f_{i1} + (1 - \omega) f_{i2} \qquad i = 1, 2, \cdots, n$$

Burada f_{ij} Weibull dağılımından elde edilen bağımsız pseudo-rasgele sayılardır. ω ise iki katlı Weibull dağılımında karma oranıdır.

Üretilen rasgele örneklemlerden farklı oranlarda veri rasgele seçilerek sağdan sansürlü veri olarak kabul edildi. Verilere uyumlu modeller, EM algoritması yöntemiyle ve GA yöntemleri ile oluşturuldu. Her simülasyon tekrarı için A^2 ve p değerleri hesaplandı.

Simülasyon çalışmasında iki katlı Weibull dağılımı için α_1=7 , β_1=3 , α_2=2 ve β_2=4 olarak sabit tutuldu. Beş farklı karma oranı (ω=0,1 ; 0,3 ; 0,5 ; 0,7 ve 0,9), dört farklı örneklem büyüklüğü (n = 25, 100, 250 ve 500) ve dört farklı sansür oranı (%10, %20, %30 ve %40) için toplam 80 durum incelenmiştir. Her durum 100 defa tekrar edildi.

Her durum için Tahmini HKO (THKO) ve Tahmini Yanlılık (TYAN) aşağıdaki eşitlikler yardımıyla hesaplandı.

$$THKO = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{\theta}_i - \theta\right)^2$$

$$TYAN = \frac{1}{m}\sum_{i=1}^{m} \hat{\theta}_i - \theta$$

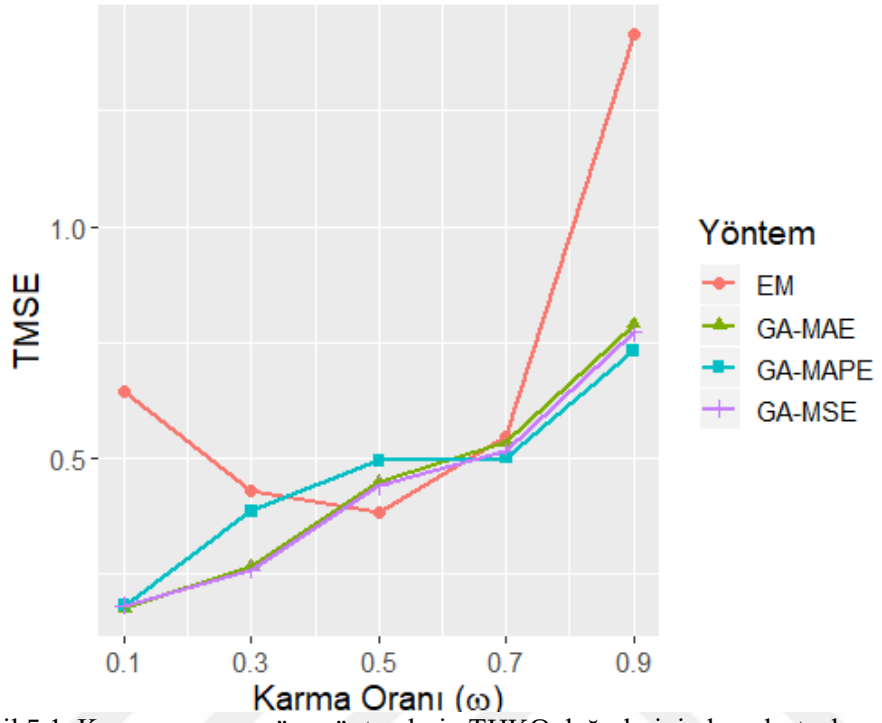
Burada $\theta = (\alpha_1, \beta_1, \alpha_2, \beta_2, \omega)$ ve m ise simülasyondaki deneme sayısıdır.

Simülasyon sonucunda farklı karma oranları, farklı örneklem büyüklükleri ve farklı sansür oranları için bulunan değerler Ek 1-4'te gösterilmiştir.

Simülasyon çalışmasındaki çoğu durumda, GA yöntemleri EM yönteminden daha uygun modelleri bulmuştur. GA yöntemlerinde, uygunluk fonksiyonu olarak veri ile model arasındaki mesafeyi ölçen metriklerin kullanılmasının sadece uygulamayı kolaylaştırmakla kalmayıp daha iyi sonuçlar sağladığı görülmüştür.

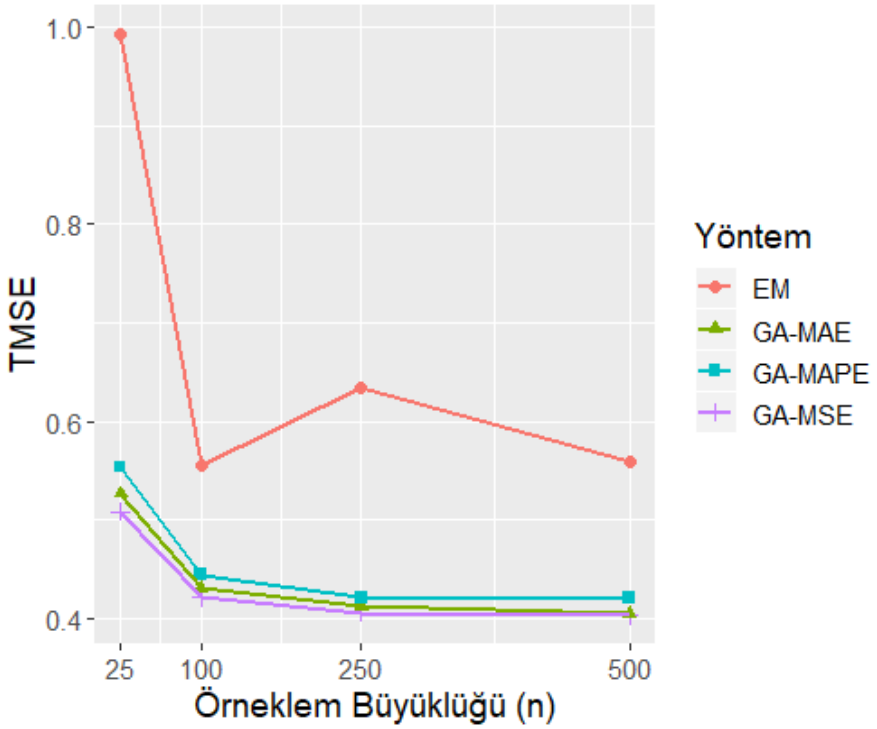
80 vaka arasında önerilen GA-MYHO yöntemi 30 vakada en düşük THKO değerini, 18 vakada GA-HKO yöntemini ve 15 vakada GA-MHO yöntemini üretmiştir. Ancak, EM yöntemi yalnızca 17 durumda en düşük THKO değerlerini üretmiştir. EM yöntemi, karma oranının 0,5 veya 0,7'ye eşit olduğu bazı durumlarda başarılıdır.

Öte yandan, β_2 parametresinin TYAN'ı için, EM algoritması 80 durumdan 54'ünde en iyi değerlere sahipken, $\alpha_1, \beta_1 \, ve \, \alpha_2$ parametrelerinin TYAN'ı için önerilen GA yöntemleri çoğunlukla en iyi değerlere sahiptir. ω parametresi için, EM ve GA algoritmaları benzer sayıda durumda en iyi TYAN değerine sahiptir.



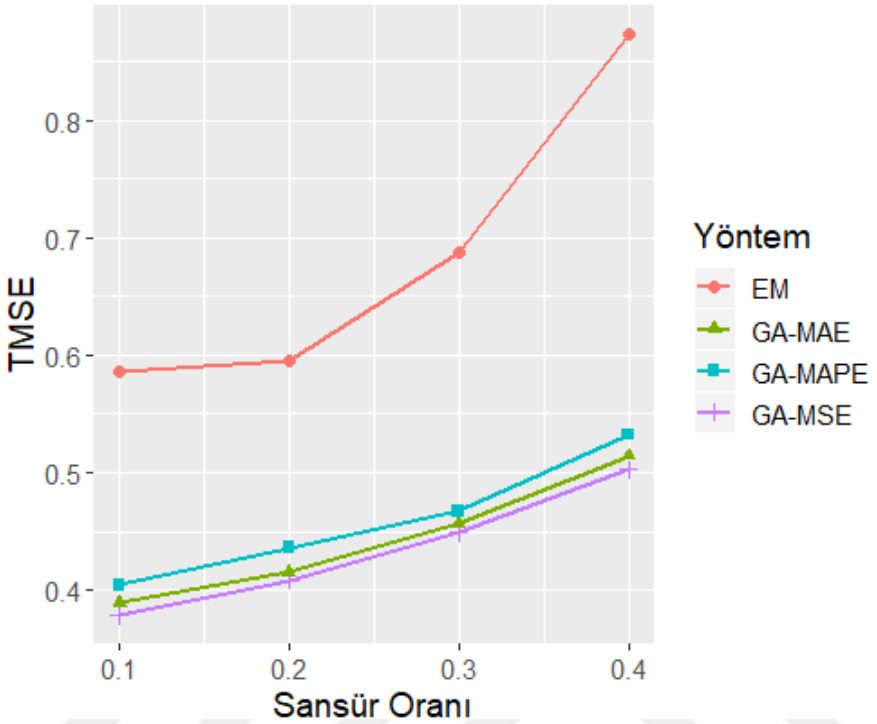
Şekil 5.1. Karma oranına göre yöntemlerin THKO değerlerinin karşılaştırılması

Şekil 5.1'deki grafikte karma oranına göre tüm durumlarda THKO değerinin değişimi gözlemlenebilir. GA yöntemlerinde karma oranı arttıkça THKO değeri de artmakta yani modelin verilere uyumunun azaldığı görülmektedir. EM yönteminde ise en iyi THKO değeri karma oranı 0,5'e eşit olduğunda elde edilmiştir. Karma oranı 0,5'e eşitken EM yöntemi önerilen yöntemlere göre daha başarılıdır. Aksine karma oranının 0,5'ten farklı olduğu durumlarda GA yöntemleri daha başarılıdır. EM yönteminin karma oranının küçük veya büyük olduğu durumlarda daha kötü sonuçlar vereceği düşünülmektedir, çünkü EM algoritması başlangıç değerlerinden etkilenmekte ve yerel çözümlere yakınsamaktadır.



Şekil 5.2. Örneklem büyüklüğüne göre yöntemlerin THKO değerlerinin karşılaştırılması

Şekil 5.2'deki grafikte, simülasyondaki her vaka için THKO değerinin örneklem büyüklüğüne göre değişimi gözlemlenebilir. Örneklem büyüklüğü arttıkça THKO değeri düşmekte, yani modellerin performansları artmaktadır. GA-HKO yöntemi diğer yöntemlere göre daha iyi performansa sahiptir.



Şekil 5.3. Sansür oranına göre yöntemlerin THKO değerlerinin karşılaştırılması

Şekil 5.3'teki grafikte simülasyondaki tüm durumlarda THKO değerinin sansür oranına göre değişimi gözlemlenebilmektedir. Sansür oranı arttıkça THKO değeri artmakta yani modellerin performansları düşmektedir. GA-HKO yöntemi diğer yöntemlere göre daha iyi performansa sahip olduğu gözlemlenmiştir.

Sonuç olarak, önerilen yöntem ile literatürde kullanılan bir veri seti için iki katlı Weibull modeli oluşturulmuş ve modelin parametreleri tahmin edilmiştir. Bulunan sonuçlar aynı veri seti üzerinde Ruhi ve ark. (2015) yaptığı çalışmalar ile karşılaştırıldığında, GA_{MYHO} yöntemiyle oluşturulan modelde Anderson-Darling uyumluluk testi sonucunun en küçük değerde olduğu tespit edilmiştir. Bu da GA_{MYHO} yöntemiyle bulunan modelin diğer yöntemlere göre veriye daha uyumlu olduğunu göstermektedir.

Gerçek veriden elde edilen sonucu daha iyi inceleyebilmek için Monte Carlo simülasyon çalışması yapılmıştır. Yapılan simülasyon çalışması ile çeşitli örneklem

büyüklükleri, sansür oranları ve karma oranları için modelin uyumluluğu test edilmiştir. Simülasyon çalışması, önerilen GA yöntemleriyle bulunan modellerin, EM algoritmasının kullanıldığı yönteme kıyasla çoğu durumda verilere daha iyi uyum sağladığını göstermiştir. Özellikle GA-MYHO yöntemi, simülasyonda daha fazla vakada en iyi performansı göstermiştir. EM yöntemi, yalnızca karma oranının 0,5'e eşit olduğu durumlarda daha iyi performans göstermiştir. GA'nın başlangıç değerlerinden ve yerel çözümlerden EM algoritmasına göre daha az etkilenmesinin bu farkı yarattığı düşünülebilir.

6. SONUÇLAR VE ÖNERİLER

Yazılım güvenilirliği bir yazılımın kalitesini gösteren en önemli metriklerden bir tanesidir. Günümüzde birçok uygulamada yazılımın giderek daha kritik görevler üstlendiği düşünülürse bir yazılımın güvenilirliğini tahmin etmenin önemi daha çok anlaşılacaktır. Bu tezin ana amacı yazılım güvenilirliğini istatistiksel açıdan incelemek ve yazılım güvenilirliği tahmin yöntemlerinde Genetik Algoritma tekniği kullanarak kestirim performansı yüksek modeller elde edebilmektir.

Yapılan ilk uygulamada karma modeller kullanılarak yazılımdaki hata sayıları modellenmeye çalışılmıştır. Farklı dağılımlar, bu dağılımların parametreleri ve karma oranlarının genetik algoritma kullanılarak optimize edilmesine dayanan bir yöntem önerilmiştir. Literatürde sıkça kullanılan ve çeşitli yazılımlardan alınan başarısızlık verileri kullanılarak önerilen yöntemle farklı modeller elde edilmiştir. Örneğin veri seti-1 için tek bir dağılımın kullanıldığı durumda Üstel Lojistik dağılımı en iyi performansı gösterirken (HKO=2,836) iki dağılımlı karma modeller için Gumbel ve Genelleştirilmiş Üstel dağılımdan elde edilen karma model en iyi performansı göstermiştir (HKO=1,651). Karma modeldeki dağılım sayısı üçe çıkarıldığında ise farklı parametrelere sahip iki adet Genelleştirilmiş Üstel dağılım ile Üstel Lojistik dağılımlarının karma modeli başarılı olmuştur (HKO=0,655).

Yapılan ikinci uygulamada ise sağdan sansürlü güvenilirlik verisini iki katlı Weibull karma modeli kullanarak tahmin edebilmek için modelin parametreleri genetik algoritma ile optimize edilmiştir. Veri seti 10-katlı çapraz doğrulama yöntemi kullanılarak 10 parçaya ayrılmış, her seferinde 1 parça test verisi olarak alınarak 10 defa eğitilmiştir. Eğitim verisi ile model parametreleri optimize edilirken test verisi ile elde edilen verinin performansı test edilmiştir. Önerilen GA-HKO, GA-MHO ve GA-MYHO yöntemleri EM ve WPP yöntemleri ile karşılaştırılmıştır. Eğitim verisinde EM algoritması en iyi performansı gösterirken (A^2=0,4093) test verisinde GA-HKO yöntemi en başarılı model olmuştur (A^2=0,9050).

İkinci uygulamada yapılan simülasyon çalışmasında ise farklı senaryolar için iki katlı Weibull karma dağılımından alınan sağdan sansürlü yapay güvenilirlik veri setleri üretilerek denemeler yapılmıştır. Simülasyon çalışmasında önerilen GA yöntemlerinin EM yöntemine göre genelde daha iyi performans gösterdiği söylenebilir. Sadece karma oranının 0,5 değerine eşit olduğu senaryolarda EM yönteminin performansının daha başarılı olduğu gözlemlenmiştir.

Bu tez çalışmasında yazılım güvenilirlik analizinde eğri uydurma modelleri için genetik algoritma tekniği uygulanarak kestirim performansı daha fazla olan modeller elde edilmiştir. Genetik algoritma parametre optimizasyonu yapmakta oldukça başarılı bir tekniktir. Farklı güvenilirlik modelleri için de genetik algoritma kullanılarak yeni tahmin yöntemleri geliştirilebilir.

Dahası makine öğrenmesi ve yapay zekâ günümüzde giderek gelişen konulardır. Genetik algoritmanın dışında yapay sinir ağları, destek vektör makineleri, derin öğrenme gibi birçok makine öğrenmesi tekniği vardır ve gelişmektedir. Farklı makine öğrenmesi teknikleri kullanılarak yeni yazılım güvenilirlik analizi yöntemleri geliştirmek mümkündür.

Son olarak bilişim alanı sürekli ve hızlı gelişmekte olan bir alan olduğu herkes tarafından bilinmektedir. Sürekli artan hesaplama gücü talebine bağlı olarak paralel hesaplamalar yapan sistemler, kuantum bilgisayar sistemleri, optik bilgisayar sistemleri gibi yeni gelişen sistemler için farklı yazılım güvenilirliği modelleri geliştirilmesine ihtiyaç olacaktır. Bu tezin, istatistikçiler için yazılım güvenilirliği alanında bir rehber görevi göreceğini umuyoruz.

KAYNAKÇA

Aggarwal K.K., Singh Y., Kaur A., Malhotra R. (2006) Investigating the effect of coupling metrics on fault proneness in object-oriented systems. Softw Qual Prof 8(4):4-16

Amin A., Grunske L., Colman A. (2013) An approach to software reliability prediction based on time series modeling, The Journal of Systems and Software, 86, s. 1923–1932

Arnold D.N. (2020) Two disasters caused by computer arithmetic errors, http://www-users.math.umn.edu/~arnold/455.f96/disasters.html, (Erişim Tarihi: 29.08.2020)

Balakrishnan N., Ng H.K.T., Navarro J. (2011) "Linear inference for type-II censored lifetime data of reliability systems with known signatures" IEEE Transaction on Reliability 60:426–440

Bazovsky I. (2004) Reliability Theory and Practice, Prentice-Hall, Englewood Cliffs, NJ

Beizer, B. (1995) Black-box Testing: techniques for functional testing of software and systems. Wiley, New York

Belady L.A., Lehman M.M. (1976) "A model of large program development" IBM System Journal, vol 3

Berchtold A. (2004) "Optimization of mixture models: Comparison of Different Strategies" Computational Statistics 19:385-406

Blischke W.R. ve Murthy D.N.P. (2000) "Reliability modeling, prediction, and optimization" Wiley, New York

Bordes L. ve Chauveau D. (2016) "Stochastic EM algorithms for parametric and semiparametric mixture models for right-censored lifetime data" Computational Statistics 31:1513-1538

Box J., Reinsel J. (1994) Time-series Model, Wiley, New York

Brahati R. ve Selverani R. (2020) Hidden Markov Model Approach for Software Reliability Estimation with Logic Error, International Journal of Automation and Computing, 17(2), s. 305-320

Braiek H.B. ve Khomh F. (2020) On testing machine learning programs, Journal of Systems and Software, vol. 164, s. 1-18

Cai K.Y. (1998) "On estimating the number of defects remaining in software," Journal of Systems and Software, Vol. 40(l)

Costa E.O., Pozo A.T.R. ve Vergilio S.R. (2010) A Genetic Programming Approach for Software Reliability Modeling, IEEE Transactions on Reliability, (59) 1 s. 222.230

Coutinho J.S. (1973) "Software reliability growth" Proceedings International Conference on Reliable Software, IEEE Computer Society Press, Los Angeles

Felderer M., Büchler M., Johns M., Brucker A.D., Breu R., Pretschner A. (2016) Securty Testing: A Survey, Advances in Computers, 101, s. 1-54

Ferreira L.A. ve Silva J.L. (2017) "Parameter estimation for Weibull distribution with right censored data using EM algorithm" Eksploatacja i Niezawodnosc – Maintenance and Reliability 19(2):310–315

Fischer T. ve Kamps U. (2013) "Power maps in goodness-of-fit testing" Computational Statistics 28:1365-1382

Goel A.L. ve Okumoto K. (1979) "Time-dependent error-detection rate model for software and other performance measures," IEEE Transaction on Reliability, 28:206-211

Goel A.L. ve Okumoto K. (1985) "When to stop testing and start using software?" ACM/Sigmetrics, 10:131-135

Goel B. ve Singh Y. (2009) An empirical analysis of metrics. Softw Qual Prof 11(3):35–45

Halstead M.H. (1977) Elements of Software Science, Elsevier, New York

Hamlet D. (1994) Foundations of software testing: dependability theory, Proceedings of the second ACM SIGSOFT symposium on Foundations of software engineering, s. 128 - 139

Hetzel, W.C. (1988) The Complete Guide to Software Testing, Wellesley, Mass.

Howden W.E. (1987) Functional program Testing and Analysis. McGraw-Hill

Hsu C.J. ve Huang C.Y. (2014) "Optimal Weighted Combinational Models for Software Reliability Estimation and Analysis", IEEE Transactions on Reliability, vol. 63, no. 3, s. 731-749

Huang X.X. (1984) "The hypergeometric distribution model for predicting the reliability of software," Microelectronics and Reliability, Vol. 24(l)

IEEE (1990) IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990), IEEE Computer Soc.

Ireson W.G., Coombs C.F., Moss R.Y., (1995), "Handbook of Reliability Engineering and Management", McGraw-Hill Comp.

Jaiswal A. ve Melhotra R. (2018) Software reliability prediction using machine learning techniques, Int J Syst Assur Eng Manag, 9(1): s. 230–244

Jelinski Z., Moranda P.B. (1972) "Software reliability research" Statistical Computer Performance Evaluation, W. Freiberger (ed), Academic Press, New York

Keiller, P.A. ve Miller, D.R. (1991) "On the Use and the Performance of Software Reliability Growth Models", Software Reliability and Safety, Elsevier, s. 95-117

Knight C.R., Jervis E.R. ve Herd G.R. (1955) Terms of Interest in the Study of Reliability, ARINC Monograph No:2, Aeronautical Radio Inc, Washington

Kohavi R. (1995) The power of decision tables. In: The eighth european conference on machine learning (ECML-95), Heraklion, Yunanistan, s. 174-189

Koopman P. ve DeVale J. (1999) Comparing the Robustness of POSIX Operating Systems. Proceedings of FTCS'99, Madison, Wisconsin.

Kumar P. ve Singh Y. (2012) An empirical study of software reliability prediction using machine learning techniques, Int J Syst Assur Eng Manag, 3(3):194-208

Leemis L.M. (1995) “Reliability: Probabilistic Models and Statistical Methods” Prentice-Hall, Englewood Cliffs, NJ.

Lin H. (1985) Scientific American, vol. 253, no. 6, s.48

Lions, J. (1996) Ariane 5 Flight 501 Failure, European Space Agency, Paris, http://sunnyday.mit.edu/nasa-class/Ariane5-report.html, (Erişim Tarihi: 23.09.2020)

Littlewood B. (1979) "Software reliability model for modular program structure" IEEE Trans. Reliability, Vol. R-28(3)

Lyu M.R. (1996) Handbook of Software Reliability Engineering, McGraw-Hill, New York

Malaiya Y.K., Srimani P.K. (1990) Software Reliability Models: Theoretical Developments, Evaluation and Applications, IEEE Computer Society Press, Los Angeles

Myers, G.J. (1979) The art of software testing, Wiley, New York

McCabe T.J. (1976) “A complexity measure” IEEE Trans. Software Engineering, Vol. SE-2(4)

Miller D.R., Sofer A. (1985) “Completely monotone regression estimation of software failure rate” Proc. International Conference on Software Engineering, IEEE Computer Society Press, Los Angeles

Mills H.D. (1970) "On the statistical validation of computer programs," IBM FSD

Moranda P.B. (1979) “An error detection model for application during software development” IEEE Trans. Reliability, Vol. R-28(5)

Murthy D.N.P., Xie M., Jiang R. (2004) “Weibull models” Wiley, New York

Ng H.K.T., Navarro J., Balakrishnan N. (2012) “Parametric inference from system lifetime data under a proportional hazard rate model” Metrika 75:367–388

Ohba M. (1984) “Software reliability analysis models,” IBM J. Res. Develop., vol. 28, no. 4, s. 428–443

Ohba M. ve Yamada S. (1984) "S-shaped software reliability growth models," Proc. 4th Int.Conf. Reliability and Maintainability:430-436

Pan J. (1999) Lecture Notes of Sotware Testing, Carnegie Melon University

Parrington N. ve Roper M. (1989) Understanding Software Testing, John Willey & Sons

Perry, W.E. (1990) A standard for testing application software

Pham H. ve Zhang X. (1997) "An NHPP software reliability model and its comparison," International Journal of Reliability, Quality and Safety Engineering, vol. 4, no. 3:269-282

Pham H., Nordmann L., Zhang X. (1999) "A general imperfect software debugging model with s-shaped fault detection rate," IEEE Transactions on Reliability, vol. 48, no. 2, June:169-175

Pham H. (2000) Software Reliability, Springer-Verlag

Pham H. (2005) "A new generalized systemability model", *International Journal of Performability Engineering*, vol 1, no. 2:145-155

Pham H. (2006) "System Software Reliability", Springer Series in Reliability Engineering

R Core Team (2018) “R: a language and environment for statistical computing” R foundation for statistical computing, Vienna, Austria

RAC (1996) Introduction to Software Reliability: A state of the Art Review, Reliability Analysis Center (RAC)

Rausand M. ve Hoyland A., 2004, System Reliability Theory; Models, Statistical Methods and Applications. John Wiley & Sons Inc

Razali N.M. ve Yap B.W. (2010) “Power Comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling Tests” Proceedings of the Regional Conferences on Statistical Sciences (RCSS’10) at Kota Bharu, Kelentan, Malaysia

Ruhi S., Sarker S., Karim M.R. (2015) “Mixture models for analyzing product reliability data: a case study” SpringerPlus 4: 634

Schick G.J. ve Wolverton R.W. (1978) “An analysis of competing software reliability Models” IEEE Trans. Software Engineering, vol. SE- 4(2)

Scrucca L. (2013) “GA: A Package for Genetic Algorithms in R” Journal of Statistical Software 53(4): 1-37.

Singh Y. ve Kumar P. (2010) A software reliability growth model for three-tier client–server system. Int J Comp Appl 1(13):9–16

Smith, C.U. (1990) Performance Engineering of Software Systems. Addison-Wesley.

Suharson D. ve Prahba Dr. (2020), "Hybrid software reliability model with Pareto distribution and ant colony optimization (PD–ACO)", International Journal of Intelligent Unmanned Systems, Vol. 8 No. 2, s. 129-140.

Sukert A.N. (1977) “An investigation of software reliability models” Proc. Annual Reliability & Maintainability Symposium, IEEE Reliability Society, Piscataway

Syswerda, G. (1993) “Simulated Crossover in Genetic Algorithms” In Proceedings of the 2nd Workshop on Foundations of Genetic Algorithms, L. Whitley, Ed. Morgan-Kaufmann Publishers, San Mateo, Calif., 239–255.

Tamura Y., Matsumoto M. ve Yamada S. (2016) "Software Reliability Model Selection Based on Deep Learning," International Conference on Industrial Engineering, Management Science and Application (ICIMSA), Jeju, s. 1-5.

Tohma Y., Yamano H., Ohba M., Jacoby R. (1991) "The estimation of parameters of the hypergeometric distribution and its application to the software reliability growth model” IEEE Trans. Software Engineering, vol. SE-17(5)

Trivedi, K. S. (2001) “Probability and Statistics with Reliability Queuing and Computer Science Applications.” Wiley

Vokolos F.I. ve Weyuker E.J. (1998) Performance testing of software systems, Proceedings of the first international workshop on Software and performance, s. 80 - 87

Volterman W. ve Balakrishnan N. (2013) “Efficient iterative computation of mixture weights for pooled order statistics for meta-analysis of multiple type-II right censored data” Computational Statistics 28: 2231-2239

Wall J.K., Ferguson P.A. (1977) “Pragmatic software reliability prediction” Proc. Annual Reliability & Maintainability Symposium, IEEE Reliability Society, Piscataway

Wang F.K. (2014) “Using BBPSO Algorithm to Estimate the Weibull Parameters with Censored Data” Communications in Statistics - Simulation and Computation, 43:10, 2614-2627

Wang F.K. ve Huang P.R. (2014) “Implementing particle swarm optimization algorithm to estimate the mixture of two Weibull parameters with censored data” Journal of Statistical Computation and Simulation, 84:9, 1975-1989

Weibull W. (1951) “A statistical distribution function of wide applicability," J Applied Mech., Vol. 18:293-297

Wood A. (1996) “Predicting software reliability,” IEEE Comput., vol. 29, no. 11, s. 69–77.

Xie M. (1991) Software Reliability Modelling, World Scientific, Singapore

Yamada S., Ohba M., Osaki S. (1984) "S-shaped software reliability growth models and their applications," IEEE Trans. Reliability, R-33

Yamada, S. ve Osaki, S. (1985) “Software reliability growth modeling: models and applications,” IEEE Transactions on Software Engineering, SE-11(12):1431-1437

Yamada S., Tokuno K., Kasano Y. (1998) “Quantitative assessment models for software safety/reliability,” Electronics and Communications in Japan, Part 2, vol 81, no. 5

Yang Y., Ng H.K.T., Balakrishnan N. (2016) “A stochastic exceptation-maximization algorithm for the analysis of system lifetime data with known signature” Computational Statistics 31: 609-641

Zheng J. (2009) Predicting software reliability with neural network ensembles, Expert Systems with Applications, (36) 2:1, s. 2116-2122

Zhang J., Ng H.K.T., Balakrishnan N. (2015) “Statistical inference of component lifetimes with location-scale distributions from censored system failure data with known signature” IEEE Transaction on Reliability 64: 613-626

Zhou M. ve Yang Y. (2015) “A recursive formula for the Kaplan-Meier estimator with mean constraints and its application to emprical likelihood” Computational Statistics 30:1097-1109

ÖZGEÇMİŞ

1969 yılında Adana'da doğdu. İlköğrenimini Adana Ziyapaşa İlkokulu'nda, Ortaöğrenimini Muğla Merkez Ortaokulu'nda ve Muğla Turgut Reis Lisesi'nde tamamladı. 1993 yılında İstanbul Teknik Üniversitesi Elektrik-Elektronik Fakültesi, Kontrol ve Bilgisayar Mühendisliği Bölümü'nden mezun oldu. Yüksek Lisans eğitimin 2006 yılında Çukurova Üniversitesi Fen Bilimleri Enstitüsü İstatistik Anabilim dalında tamamlayıp 2009 yılında aynı ana bilim dalında doktora eğitimine başladı.

1995-1996 yıllarında Donanma Komutanlığında askeri görevini tamamladı. 1993-2011 yılları arasında özel ve kamu sektörlerindeki çeşitli kurumlarda Bilgi Sistemleri Uzmanı olarak çalıştı.

Evli ve iki çocuk babası olan Erkut Tekeli, 2011 yılından bu yana Çukurova Üniversitesi Kozan Meslek Yüksekokulu Bilgisayar Programcılığı bölümünde Öğretim Görevlisi olarak çalışma hayatına devam etmektedir.

EKLER

Ek 1. Simülasyon sonuçlarına göre n=25 için TYAN ve THKO sonuçları

Vakalar		EM					GA-HKO					GA-MHO					GA-MYHO				
ω	SO	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$
0,1	0,1	-0,752	0,214	0,081	0,012	0,092	0,012	0,019	0,072	0,061	0,005	0,018	-0,009	0,069	0,064	0,008	0,024	0,126	0,074	0,098	0,001
		THKO= 0,6489					THKO= 0,2204					THKO= 0,2305					**THKO= 0,1701**				
	0,2	-0,737	0,220	0,096	0,016	0,095	0,011	0,010	0,085	0,068	0,009	-0,014	-0,001	0,079	0,079	0,014	-0,023	0,058	0,088	0,093	0,002
		THKO= 0,6387					THKO= 0,2251					THKO= 0,2053					**THKO= 0,1888**				
	0,3	-0,726	0,230	0,107	0,025	0,094	0,017	0,044	0,096	0,090	0,012	0,010	-0,086	0,091	0,112	0,018	0,017	0,046	0,096	0,135	0,006
		THKO= 0,6375					**THKO= 0,1732**					THKO= 0,2026					THKO= 0,2216				
	0,4	-0,705	0,237	0,128	0,031	0,096	0,012	0,000	0,114	0,126	0,018	0,023	0,000	0,108	0,134	0,023	-0,008	0,042	0,112	0,159	0,016
		THKO= 0,6271					THKO= 0,2594					THKO= 0,2521					**THKO= 0,2420**				
0,3	0,1	-0,587	0,144	0,247	0,047	0,065	0,106	0,011	0,239	0,000	0,027	-0,130	0,032	0,234	-0,001	0,024	-0,139	0,045	0,245	0,016	0,038
		THKO= 0,4460					**THKO= 0,2384**					THKO= 0,2387					THKO= 0,2412				
	0,2	-0,564	0,148	0,270	0,044	0,061	0,111	0,052	0,257	0,025	0,021	-0,123	0,028	0,253	0,033	0,017	-0,134	0,045	0,263	0,032	0,032
		THKO= 0,4358					THKO= 0,2868					THKO= 0,2775					**THKO= 0,2711**				
	0,3	-0,546	0,156	0,287	0,036	0,059	0,100	0,006	0,275	0,064	0,016	-0,114	0,017	0,268	0,065	0,012	-0,131	0,050	0,276	0,075	0,023
		THKO= 0,4336					**THKO= 0,2737**					THKO= 0,3003					THKO= 0,3222				
	0,4	-0,514	0,162	0,319	0,031	0,058	0,109	0,059	0,298	0,088	0,006	-0,116	0,032	0,292	0,106	0,001	-0,148	0,121	0,297	0,073	0,013
		THKO= 0,4305					**THKO= 0,3338**					THKO= 0,3413					THKO= 0,3418				
0,5	0,1	-0,421	0,109	0,412	0,076	0,041	0,212	0,325	0,351	0,017	0,032	-0,206	0,351	0,343	-0,012	0,031	-0,224	0,584	0,336	0,021	0,035
		THKO= 0,3831					THKO= 0,4719					THKO= 0,5062					THKO= 0,6743				
	0,2	-0,390	0,112	0,443	0,073	0,037	0,185	0,286	0,371	0,010	0,024	-0,186	0,353	0,369	0,021	0,022	-0,205	0,561	0,348	0,020	0,025
		THKO= 0,3872					THKO= 0,4512					THKO= 0,5613					THKO= 0,6597				
	0,3	-0,366	0,120	0,467	0,066	0,029	0,161	0,307	0,396	0,067	0,019	-0,165	0,288	0,397	0,066	0,021	-0,184	0,507	0,367	0,041	0,020
		THKO= 0,3965					THKO= 0,5413					THKO= 0,5401					THKO= 0,6508				
	0,4	-0,323	0,125	0,511	0,062	0,025	0,117	0,324	0,420	0,071	0,007	-0,114	0,295	0,430	0,101	0,009	-0,147	0,446	0,401	0,114	0,010
		THKO= 0,4194					THKO= 0,6032					THKO= 0,6050					THKO= 0,7154				

Ek 1. Devamı

Vakalar		EM					GA-HKO					GA-MHO					GA-MYHO				
ω	SO	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$
0,7	0,1	-0,255	0,088	0,578	-0,094	-0,025	-0,090	0,358	0,456	-0,007	-0,046	-0,106	0,353	0,436	0,007	-0,041	-0,089	0,460	0,433	0,025	-0,041
		THKO= 0,4383					THKO= 0,5881					THKO= 0,5700					THKO= 0,6059				
	0,2	-0,217	0,091	0,617	-0,091	-0,018	-0,042	0,333	0,460	0,055	-0,037	-0,058	0,357	0,488	0,056	-0,036	-0,046	0,433	0,448	0,056	-0,036
		THKO= 0,4683					THKO= 0,5774					THKO= 0,6647					THKO= 0,6296				
	0,3	-0,186	0,098	0,647	-0,085	-0,011	-0,022	0,351	0,516	0,069	-0,036	-0,014	0,303	0,541	0,084	-0,037	-0,030	0,442	0,495	0,039	-0,035
		THKO= 0,4993					THKO= 0,6844					THKO= 0,7306					THKO= 0,7134				
	0,4	-0,131	0,103	0,702	-0,081	-0,008	0,067	0,308	0,571	0,123	-0,035	0,034	0,279	0,596	0,113	-0,029	0,017	0,439	0,561	0,117	-0,029
		THKO= 0,5661					THKO= 0,7931					THKO= 0,7721					THKO= 0,8717				
0,9	0,1	-0,147	0,073	0,791	0,365	-0,054	0,075	0,321	0,552	0,045	-0,068	0,057	0,366	0,591	0,007	-0,069	0,065	0,390	0,558	0,046	-0,067
		THKO= 1,7628					**THKO= 0,6604**					THKO= 0,7564					THKO= 0,6939				
	0,2	-0,081	0,080	0,799	0,347	-0,049	0,100	0,323	0,633	0,075	-0,065	0,107	0,317	0,633	0,077	-0,065	0,089	0,362	0,626	0,082	-0,066
		THKO= 1,6300					THKO= 0,8326					**THKO= 0,8266**					THKO= 0,8574				
	0,3	-0,014	0,101	0,807	0,484	-0,047	0,151	0,348	0,661	0,108	-0,063	0,154	0,340	0,640	0,051	-0,065	0,132	0,385	0,636	0,098	-0,061
		THKO= 2,5637					THKO= 0,9177					**THKO= 0,8521**					THKO= 0,9072				
	0,4	0,075	0,146	0,916	0,732	-0,045	0,194	0,299	0,742	0,125	-0,060	0,178	0,371	0,745	0,079	-0,064	0,195	0,354	0,724	0,148	-0,059
		THKO= 6,0531					**THKO= 1,0425**					THKO= 1,1017					THKO= 1,0990				

EK 2. Simülasyon sonuçlarına göre n=100 için TYAN ve THKO

Vakalar		EM					GA-HKO					GA-MHO					GA-MYHO				
ω	SO	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$
0,1	0,1	-0,742	0,157	0,091	-0,036	0,097	0,017	0,006	0,080	-0,028	0,002	0,003	0,039	0,077	-0,019	0,004	0,048	-0,038	0,087	-0,052	0,007
		THKO= 0,5975					THKO= 0,1823					THKO= 0,1615					**THKO= 0,1496**				
	0,2	-0,729	0,162	0,105	-0,032	0,094	0,014	0,094	0,094	-0,025	0,000	-0,044	0,015	0,088	-0,044	0,009	0,022	-0,047	0,099	-0,055	0,004
		THKO= 0,5830					THKO= 0,1789					**THKO= 0,1614**					THKO= 0,1881				
	0,3	-0,713	0,163	0,121	-0,031	0,094	0,028	0,092	0,110	-0,023	0,003	-0,029	0,051	0,103	-0,052	0,013	0,039	-0,024	0,112	-0,054	0,001
		THKO= 0,5645					THKO= 0,1773					**THKO= 0,1579**					THKO= 0,1640				
	0,4	-0,694	0,161	0,139	-0,032	0,092	0,027	-0,006	0,127	-0,030	0,008	-0,003	0,004	0,118	-0,049	0,017	0,048	0,009	0,128	-0,057	0,006
		THKO= 0,5436					**THKO= 0,1555**					THKO= 0,1797					THKO= 0,1921				
0,3	0,1	-0,571	0,091	0,262	-0,091	0,062	0,132	0,026	0,254	-0,037	0,031	-0,142	0,072	0,247	-0,032	0,027	0,177	0,204	0,260	-0,015	0,040
		THKO= 0,4173					**THKO= 0,2050**					THKO= 0,2217					THKO= 0,3642				
	0,2	-0,550	0,095	0,283	-0,087	0,060	0,151	0,188	0,269	-0,047	0,029	-0,176	0,060	0,262	-0,029	0,021	0,207	0,215	0,277	-0,014	0,036
		THKO= 0,4064					THKO= 0,2533					**THKO= 0,2201**					THKO= 0,3288				
	0,3	-0,526	0,097	0,308	-0,086	0,059	0,134	0,084	0,293	-0,020	0,022	-0,185	0,123	0,283	-0,030	0,016	0,219	0,276	0,287	-0,009	0,026
		THKO= 0,3949					**THKO= 0,2596**					THKO= 0,2743					THKO= 0,3745				
	0,4	-0,496	0,095	0,337	-0,088	0,058	0,149	0,078	0,314	-0,027	0,014	-0,169	0,148	0,304	-0,019	0,009	0,204	0,242	0,315	-0,009	0,020
		THKO= 0,3849					**THKO= 0,2627**					THKO= 0,2951					THKO= 0,3848				
0,5	0,1	-0,400	0,059	0,433	-0,118	0,025	0,210	0,376	0,373	-0,086	0,038	-0,219	0,357	0,367	-0,086	0,033	0,228	0,501	0,346	-0,041	0,029
		THKO= 0,3691					THKO= 0,4264					THKO= 0,4235					THKO= 0,4871				
	0,2	-0,372	0,063	0,462	-0,115	0,028	0,196	0,342	0,390	-0,084	0,029	-0,208	0,378	0,381	-0,083	0,023	0,203	0,485	0,368	-0,055	0,024
		THKO= 0,3730					THKO= 0,4157					THKO= 0,4325					THKO= 0,4928				
	0,3	-0,339	0,064	0,495	-0,113	0,024	0,177	0,339	0,420	-0,077	0,020	-0,164	0,367	0,422	-0,066	0,025	0,187	0,448	0,391	-0,053	0,015
		THKO= 0,3813					THKO= 0,4656					THKO= 0,4789					THKO= 0,4818				
	0,4	-0,299	0,062	0,534	-0,115	0,025	0,138	0,296	0,458	-0,070	0,015	-0,135	0,329	0,440	-0,048	0,010	0,143	0,469	0,428	-0,072	0,012
		THKO= 0,3981					THKO= 0,4420					THKO= 0,4640					THKO= 0,5748				

Ek 2. Devamı

Vakalar		EM					GA-HKO					GA-MHO					GA-MYHO				
ω	SO	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$
0,7	0,1	-0,229	0,039	0,604	-0,134	-0,011	-0,089	0,267	0,496	-0,127	-0,042	-0,093	0,241	0,499	-0,085	-0,042	-0,076	0,310	0,461	-0,086	-0,039
				THKO= 0,4418					THKO= 0,4232					THKO= 0,4302					**THKO= 0,3860**		
	0,2	-0,193	0,043	0,640	-0,131	-0,012	-0,039	0,278	0,538	-0,109	-0,046	-0,062	0,263	0,533	-0,083	-0,038	-0,048	0,324	0,502	-0,084	-0,038
				THKO= 0,4713					THKO= 0,4738					THKO= 0,4652					**THKO= 0,4399**		
	0,3	-0,152	0,044	0,682	-0,130	-0,009	-0,019	0,241	0,586	-0,053	-0,039	-0,010	0,270	0,577	-0,089	-0,039	-0,007	0,309	0,536	-0,059	-0,034
				THKO= 0,5120					THKO= 0,5404					THKO= 0,5480					**THKO= 0,5115**		
	0,4	-0,101	0,043	0,732	-0,131	-0,008	0,039	0,227	0,621	-0,060	-0,034	0,043	0,270	0,611	-0,059	-0,035	0,026	0,266	0,577	-0,060	-0,023
				THKO= 0,5718					THKO= 0,5714					THKO= 0,6223					**THKO= 0,5403**		
0,9	0,1	-0,120	-0,020	0,837	0,050	-0,061	0,077	0,204	0,655	-0,117	-0,071	0,061	0,171	0,670	-0,128	-0,068	0,090	0,234	0,604	-0,106	-0,066
				THKO= 0,8098					THKO= 0,6039					THKO= 0,5989					**THKO= 0,5256**		
	0,2	-0,062	-0,012	0,860	0,099	-0,059	0,123	0,195	0,694	-0,100	-0,071	0,115	0,200	0,704	-0,094	-0,070	0,111	0,231	0,694	-0,115	-0,068
				THKO= 0,9748					**THKO= 0,6642**					THKO= 0,6965					THKO= 0,6725		
	0,3	-0,003	-0,001	0,893	0,081	-0,059	0,163	0,179	0,759	-0,071	-0,065	0,164	0,198	0,758	-0,075	-0,068	0,169	0,216	0,698	-0,061	-0,061
				THKO= 0,9383					THKO= 0,8026					THKO= 0,8044					**THKO= 0,7237**		
	0,4	0,089	0,002	0,917	0,052	-0,058	0,225	0,215	0,801	-0,052	-0,057	0,219	0,213	0,812	-0,038	-0,061	0,215	0,208	0,779	-0,017	-0,059
				THKO= 0,9822					THKO= 0,9194					THKO= 0,9852					**THKO= 0,9068**		

Ek 3. Simülasyon sonuçlarına göre n=250 için TYAN ve THKO

Vakalar		EM					GA-HKO					GA-MHO					GA-MYHO				
ω	SO	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$
0,1	0,1	-0,760	0,119	0,132	0,134	0,097	0,007	0,016	0,082	0,021	0,000	-0,036	-0,026	0,075	0,035	0,008	-0,035	0,025	0,090	0,025	-0,010
		THKO= 0,6779					THKO= 0,1897					**THKO= 0,1485**					THKO= 0,1615				
	0,2	-0,747	0,115	0,144	0,126	0,095	-0,020	0,041	0,094	0,023	0,002	-0,014	-0,018	0,088	0,029	0,009	-0,003	0,000	0,102	0,033	-0,006
		THKO= 0,6587					THKO= 0,1485					**THKO= 0,1385**					THKO= 0,1922				
	0,3	-0,734	0,120	0,163	0,226	0,093	0,006	0,087	0,110	0,017	0,003	-0,036	0,059	0,103	0,032	0,011	-0,040	0,045	0,114	0,046	-0,001
		THKO= 1,1044					THKO= 0,1651					**THKO= 0,1548**					THKO= 0,1717				
	0,4	-0,715	0,120	0,181	0,187	0,091	0,029	0,085	0,127	0,029	0,007	-0,026	0,075	0,120	0,040	0,016	0,000	0,042	0,131	0,041	0,002
		THKO= 0,8108					THKO= 0,1718					**THKO= 0,1712**					THKO= 0,1754				
0,3	0,1	-0,601	0,045	0,314	0,024	0,052	-0,134	0,029	0,253	0,034	-0,031	-0,130	0,169	0,247	-0,041	0,028	-0,221	0,338	0,261	-0,018	-0,042
		THKO= 0,4824					**THKO= 0,2035**					THKO= 0,2615					THKO= 0,4303				
	0,2	-0,583	0,045	0,336	0,057	0,051	-0,160	0,115	0,271	0,040	-0,028	-0,146	0,042	0,266	-0,038	0,023	-0,223	0,269	0,276	-0,023	-0,034
		THKO= 0,4955					THKO= 0,2351					**THKO= 0,2048**					THKO= 0,3575				
	0,3	-0,554	0,043	0,356	0,064	0,051	-0,131	0,094	0,295	0,047	-0,025	-0,185	0,173	0,282	-0,034	0,017	-0,245	0,356	0,293	-0,028	-0,027
		THKO= 0,5062					**THKO= 0,2351**					THKO= 0,2952					THKO= 0,4339				
	0,4	-0,525	0,047	0,385	0,066	0,050	-0,157	0,186	0,319	0,043	-0,020	-0,174	0,152	0,309	-0,039	0,010	-0,202	0,346	0,319	-0,036	-0,023
		THKO= 0,5089					THKO= 0,2934					**THKO= 0,2837**					THKO= 0,4179				
0,5	0,1	-0,395	0,048	0,438	-0,126	0,027	-0,213	0,327	0,379	0,098	-0,037	-0,224	0,338	0,373	-0,095	0,034	-0,217	0,451	0,357	-0,056	-0,033
		THKO= 0,3685					THKO= 0,3758					**THKO= 0,3683**					THKO= 0,4146				
	0,2	-0,367	0,049	0,467	-0,126	0,022	-0,194	0,376	0,397	0,093	-0,030	-0,205	0,334	0,396	-0,093	0,027	-0,203	0,411	0,369	-0,057	-0,022
		THKO= 0,3729					THKO= 0,4410					THKO= 0,3931					THKO= 0,3935				
	0,3	-0,333	0,050	0,500	-0,125	0,024	-0,190	0,305	0,427	0,092	-0,017	-0,180	0,329	0,419	-0,104	0,020	-0,185	0,398	0,396	-0,082	-0,014
		THKO= 0,3822					THKO= 0,4126					THKO= 0,4095					THKO= 0,4001				
	0,4	-0,292	0,051	0,541	-0,124	0,025	-0,142	0,268	0,464	0,084	-0,013	-0,134	0,308	0,454	-0,087	0,015	-0,160	0,374	0,419	-0,066	-0,003
		THKO= 0,3996					THKO= 0,4105					THKO= 0,4184					THKO= 0,4246				

(Devamı Var)

Ek 3. Devamı

Vakalar		EM					GA-HKO					GA-MHO					GA-MYHO				
ω	SO	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$
0,7	0,1	-0,274	-0,020	0,666	-0,017	-0,026	-0,092	0,209	0,522	-0,123	-0,043	-0,095	0,221	0,526	-0,133	-0,045	-0,069	0,283	0,481	-0,089	-0,043
		THKO= 0,5586					THKO= 0,3977					THKO= 0,4163					**THKO= 0,3909**				
	0,2	-0,229	-0,020	0,692	-0,001	-0,025	-0,055	0,230	0,562	-0,147	-0,047	-0,061	0,226	0,549	-0,130	-0,043	-0,042	0,246	0,507	-0,114	-0,035
		THKO= 0,5837					THKO= 0,4439					THKO= 0,4493					**THKO= 0,3754**				
	0,3	-0,177	-0,019	0,722	0,020	-0,024	-0,008	0,192	0,602	-0,121	-0,039	-0,016	0,192	0,613	-0,126	-0,042	-0,006	0,240	0,555	-0,120	-0,033
		THKO= 0,6689					THKO= 0,4813					THKO= 0,5060					**THKO= 0,4320**				
	0,4	-0,104	-0,020	0,750	0,001	-0,023	0,043	0,182	0,622	-0,086	-0,028	0,029	0,172	0,652	-0,087	-0,029	0,047	0,229	0,623	-0,100	-0,034
		THKO= 0,6538					**THKO= 0,5185**					THKO= 0,5565					THKO= 0,5395				
0,9	0,1	-0,095	-0,032	0,829	-0,033	-0,060	0,062	0,156	0,687	-0,144	-0,072	0,063	0,150	0,700	-0,143	-0,070	0,083	0,172	0,625	-0,101	-0,063
		THKO= 0,7501					THKO= 0,5930					THKO= 0,6134					**THKO= 0,4974**				
	0,2	-0,037	-0,027	0,856	-0,018	-0,058	0,104	0,133	0,746	-0,137	-0,073	0,093	0,129	0,749	-0,125	-0,065	0,117	0,160	0,686	-0,126	-0,059
		THKO= 0,8017					THKO= 0,6900					THKO= 0,6982					**THKO= 0,5989**				
	0,3	0,024	-0,023	0,893	0,007	-0,057	0,147	0,108	0,809	-0,110	-0,066	0,160	0,129	0,792	-0,115	-0,063	0,169	0,183	0,743	-0,132	-0,065
		THKO= 0,9326					THKO= 0,8025					THKO= 0,7844					**THKO= 0,7270**				
	0,4	0,117	-0,021	0,921	-0,019	-0,056	0,208	0,107	0,857	-0,099	-0,060	0,180	0,109	0,883	-0,105	-0,064	0,208	0,141	0,833	-0,096	-0,060
		THKO= 0,9578					THKO= 0,9124					THKO= 0,9673					**THKO= 0,8888**				

Ek 4. Simülasyon sonuçlarına göre n=500 için TYAN ve THKO

Vakalar		EM					GA-HKO					GA-MHO					GA-MYHO				
ω	SO	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$
0,1	0,1	-0,738	0,146	0,095	-0,045	0,092	0,008	0,045	0,082	0,037	0,001	-0,025	0,052	0,078	0,030	0,006	-0,003	-0,090	0,093	0,036	-0,010
		THKO= 0,5865					**THKO= 0,1693**					THKO= 0,1747					THKO= 0,1899				
	0,2	-0,724	0,147	0,109	-0,044	0,096	-0,001	0,040	0,096	0,035	0,002	-0,039	0,024	0,090	0,045	0,010	-0,042	0,006	0,105	0,033	-0,008
		THKO= 0,5696					**THKO= 0,1687**					THKO= 0,1818					THKO= 0,1893				
	0,3	-0,708	0,148	0,125	-0,043	0,093	0,001	0,084	0,113	0,020	0,001	-0,026	0,032	0,105	0,036	0,011	-0,040	-0,025	0,117	0,044	-0,003
		THKO= 0,5507					THKO= 0,1798					**THKO= 0,1607**					THKO= 0,1698				
	0,4	-0,689	0,148	0,144	-0,043	0,098	-0,006	0,079	0,131	0,029	0,005	-0,035	0,068	0,122	0,040	0,014	-0,069	-0,061	0,134	0,046	0,004
		THKO= 0,5291					THKO= 0,1635					THKO= 0,1628					**THKO= 0,1626**				
0,3	0,1	-0,566	0,078	0,268	-0,102	0,071	-0,168	0,154	0,253	-0,035	-0,032	-0,177	0,126	0,246	-0,028	-0,024	-0,243	0,395	0,262	-0,022	-0,041
		THKO= 0,4120					**THKO= 0,2478**					THKO= 0,2680					THKO= 0,4607				
	0,2	-0,544	0,079	0,289	-0,101	0,064	-0,172	0,169	0,270	-0,041	-0,028	-0,183	0,154	0,263	-0,033	-0,021	-0,242	0,400	0,277	-0,019	-0,035
		THKO= 0,4004					THKO= 0,2534					**THKO= 0,2474**					THKO= 0,5067				
	0,3	-0,519	0,080	0,314	-0,100	0,060	-0,195	0,185	0,291	-0,038	-0,023	-0,173	0,154	0,287	-0,032	-0,018	-0,244	0,363	0,297	-0,027	-0,030
		THKO= 0,3890					**THKO= 0,2851**					THKO= 0,2978					THKO= 0,4724				
	0,4	-0,489	0,080	0,344	-0,100	0,058	-0,178	0,197	0,320	-0,043	-0,019	-0,162	0,139	0,312	-0,032	-0,013	-0,268	0,396	0,313	-0,038	-0,018
		THKO= 0,3788					THKO= 0,3026					**THKO= 0,2640**					THKO= 0,4707				
0,5	0,1	-0,393	0,045	0,440	-0,129	0,022	-0,220	0,359	0,382	-0,098	-0,037	-0,212	0,394	0,368	-0,086	-0,034	-0,221	0,451	0,360	-0,051	-0,033
		THKO= 0,3683					THKO= 0,4101					THKO= 0,4025					THKO= 0,4179				
	0,2	-0,365	0,044	0,469	-0,130	0,024	-0,194	0,338	0,404	-0,097	-0,031	-0,181	0,343	0,400	-0,094	-0,032	-0,195	0,428	0,377	-0,069	-0,026
		THKO= 0,3729					THKO= 0,3970					THKO= 0,4018					THKO= 0,4050				
	0,3	-0,331	0,044	0,503	-0,130	0,025	-0,167	0,296	0,432	-0,104	-0,026	-0,157	0,299	0,425	-0,103	-0,025	-0,183	0,357	0,396	-0,078	-0,013
		THKO= 0,3825					THKO= 0,3749					THKO= 0,3699					**THKO= 0,3688**				
	0,4	-0,290	0,044	0,544	-0,130	0,027	-0,140	0,285	0,473	-0,078	-0,014	-0,129	0,266	0,467	-0,089	-0,018	-0,149	0,334	0,431	-0,094	-0,008
		THKO= 0,4003					THKO= 0,4373					THKO= 0,4008					**THKO= 0,3702**				

(Devamı Var)

Ek 4. Devamı

Vakalar		EM					GA-HKO					GA-MHO					GA-MYHO				
ω	SO	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\beta}_1$	$\hat{\alpha}_2$	$\hat{\beta}_2$	$\hat{\omega}$
0,7	0,1	-0,258	-0,018	0,652	-0,069	-0,026	-0,081	0,239	0,511	-0,128	-0,047	-0,087	0,208	0,531	-0,129	-0,047	-0,081	0,236	0,481	-0,101	-0,037
		THKO= 0,5080					THKO= 0,3930					THKO= 0,4059					**THKO= 0,3413**				
	0,2	-0,212	-0,019	0,678	-0,065	-0,025	-0,046	0,204	0,548	-0,141	-0,043	-0,057	0,202	0,557	-0,117	-0,044	-0,051	0,231	0,524	-0,118	-0,033
		THKO= 0,5224					THKO= 0,4077					THKO= 0,4261					**THKO= 0,3894**				
	0,3	-0,164	-0,020	0,714	-0,020	-0,024	-0,006	0,201	0,581	-0,128	-0,040	-0,016	0,166	0,598	-0,139	-0,039	-0,010	0,207	0,549	-0,130	-0,030
		THKO= 0,6312					THKO= 0,4458					THKO= 0,4495					**THKO= 0,3949**				
	0,4	-0,099	-0,022	0,751	-0,017	-0,023	0,042	0,182	0,644	-0,114	-0,034	0,032	0,165	0,660	-0,130	-0,034	0,034	0,180	0,615	-0,137	-0,027
		THKO= 0,6747					THKO= 0,5359					THKO= 0,5432					**THKO= 0,4764**				
0,9	0,1	-0,075	-0,028	0,810	-0,087	-0,061	0,082	0,156	0,683	-0,155	-0,075	0,064	0,132	0,696	-0,143	-0,068	0,072	0,143	0,631	-0,139	-0,059
		THKO= 0,6862					THKO= 0,5788					THKO= 0,5826					**THKO= 0,4880**				
	0,2	-0,022	-0,028	0,845	-0,071	-0,059	0,115	0,131	0,715	-0,128	-0,065	0,105	0,105	0,744	-0,146	-0,066	0,129	0,158	0,689	-0,143	-0,067
		THKO= 0,7526					THKO= 0,6242					THKO= 0,6503					**THKO= 0,5899**				
	0,3	0,043	-0,028	0,880	-0,044	-0,057	0,141	0,085	0,808	-0,115	-0,064	0,151	0,084	0,825	-0,114	-0,064	0,162	0,134	0,754	-0,130	-0,062
		THKO= 0,8862					THKO= 0,7743					THKO= 0,8198					**THKO= 0,7051**				
	0,4	0,131	-0,028	1,009	-0,033	-0,055	0,213	0,119	0,863	-0,100	-0,064	0,211	0,082	0,857	-0,105	-0,058	0,211	0,112	0,832	-0,132	-0,059
		THKO= 1,1772					THKO= 0,9256					THKO= 0,8950					**THKO= 0,8459**				