İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

EVRİMSEL ALGORİTMALAR İÇİN UYGULAMA VE GELİŞTİRME YAZILIM ALTYAPISI

YÜKSEK LİSANS TEZİ
Müh. Fulya DEMİRKIRAN
(504011395)

Tezin Enstitüye Verildiği Tarih : 26 Nisan 2004
Tezin Savunulduğu Tarih : 20 Mayıs 2004

Tez Danışmanı : **Prof. Dr. A. Emre HARMANCI**
Diğer Jüri Üyeleri **Prof. Dr. İbrahim EKSİN (İ.T.Ü.)**
Yrd. Doç. Dr. A. Şima UYAR (İ.T.Ü.)

MAYIS 2004

ÖNSÖZ

Bu çalışma süresince değerli katkılarını esirgemeyen tez danışmanım sayın hocam Prof. Dr. A. Emre HARMANCI' ya, Yrd. Doç. Dr. A. Şima UYAR' a ve Öğr. Gör. H. Turgut UYAR' a teşekkür ederim. Ayrıca, bu çalışmada beni cesaretlendiren, çalışma azmi ve gücü veren başta ailem olmak üzere bütün dostlarıma sonsuz teşekkürlerimi sunarım.

MAYIS 2004 Fulya DEMİRKIRAN

İÇİNDEKİLER

KISALTMALAR	**vi**
TABLO LİSTESİ	**vii**
ŞEKİL LİSTESİ	**viii**
SEMBOL LİSTESİ	**x**
ÖZET	**xi**
SUMMARY	**xiii**
1. GİRİŞ	1
2. EVRİMSEL ALGORİTMALAR	3
2.1. Giriş	3
2.2. Evrimsel Algoritmaların Tanıtımı	3
2.3. Gerçeklenen Operatörler	8
2.3.1. Ebeveyn seçimi	8
2.3.1.1. Turnuva seçimi	8
2.3.1.2. Rulet çarkı seçimi	9
2.3.1.3. Stokastik evrensel örnekleme (SUS)	9
2.3.1.4. Çizgisel sıralama	10
2.3.1.5. Üstel sıralama	11
2.3.2. Çaprazlaşma	12
2.3.2.1. Tek noktadan çaprazlaşma	13
2.3.2.2. Çok noktadan çaprazlaşma	13
2.3.2.3. Homojen çaprazlaşma	13
2.3.2.4. Basit aritmetik çaprazlaşma	14
2.3.2.5. Tek aritmetik çaprazlaşma	15
2.3.2.6. Tüm aritmetik çaprazlaşma	16
2.3.3. Mutasyon	16
2.3.3.1. İkili mutasyon	17
2.3.3.2. Rasgele yeniden atama mutasyonu	17
2.3.3.3. Homojen mutasyon	17
2.3.3.4. Gauss mutasyonu	18
2.3.3.5. Takas mutasyonu	18
2.3.3.6. Araya ekleme mutasyonu	18
2.3.3.7. Karıştırma mutasyonu	19
2.3.3.8. Tersine çevirme mutasyonu	19

2.3.4. Sonlanma kriteri 20
2.3.4.1. Nesil sayısı 20
2.3.4.2. Optimum uygunluk 20
2.3.4.3. Yakınsama 20
2.3.5. Hayatta kalma seçimi 21
2.3.5.1. Nesilsel 21
2.3.5.2. En iyi bireyin seçimi 21
2.3.5.3. Turnuva seçimi 21
2.3.5.4. Rulet çarkı seçimi 22
2.3.5.5. Çizgisel sıralama seçimi 22
2.3.5.6. Üstel sıralama seçimi 23
2.4. Gerçeklenen Örnek Problemler 23
2.4.1. Bir maksimizasyonu 23
2.4.2. Altkümeler toplamı (Çanta Problemleri) 24
2.4.2.1. 0/1 çanta 24
2.4.2.2. Çok sayıda çanta 27
2.4.3. De Jong fonksiyonları 28
2.4.3.1. Gray kodlaması 28
2.4.3.2. F1 fonksiyonu 30
2.4.3.3. F2 fonksiyonu 31
2.4.3.4. F3 fonksiyonu 32
2.4.3.5. F4 fonksiyonu 32
2.4.3.6. F5 fonksiyonu 33
2.4.3.7. F7 fonksiyonu 34

3. PyLEA PAKETİ 35
3.1. Giriş 35
3.2. PyLEA Tanıtımı 35
3.3. Paket Yapısı 36
3.4. Sınıflar, Metotları ve Özellikleri 37
3.4.1. Gen sınıfı 38
3.4.2. Kromozom sınıfı 40
3.4.3. Birey sınıfı 41
3.4.4. Toplum sınıfı 42
3.4.5. İstatistik sınıfı 44
3.4.6. Grafik sınıfları 47
3.5. Bir Maksimizasyonu Problemi İçin SGA Örnek Algoritması 48
3.6. Benzer Çalışmalar 50
3.7. Avantaj ve Dezavantajlar 51

4. EA ARAYÜZÜ VE KULLANIM ÖRNEKLERİ **53**
4.1. Giriş 53
4.2. EA Arayüzü 53
4.3. Seçenek ve Özellikler 56
4.3.1. İzleme Seçenekleri 56
4.3.2. Görünüm Özelliği 57
4.3.3. İstatistik Özelliği 57
4.4. Eğitim Amaçlı Kullanımlar 58
4.5. Uygulama Amaçlı Kullanımlar 58
4.6. Geliştirme / Akademik Amaçlı Kullanımlar 59
4.7. Örnek Bir Uygulama: DomGA 60

5. SONUÇLAR **62**

6. YAPILABİLECEKLER **63**

KAYNAKLAR **64**

EK A : PYTHON **66**

EK B : REPORTLAB **68**

EK C : TK **70**

ÖZGEÇMİŞ **71**

KISALTMALAR

EA	: Evrimsel Algoritma
PyLEA	: Python Library for Evolutionary Algorithms
SUS	: Stochastic Universal Sampling
SGA	: Simple Genetic Algorithm
GENESIS	: GENEtic Search Implementation System
GENOCOP	: GEnetic algorithm for Numerical Optimisation for COnstrained Problems
EO	: Evolving Objects
BEAGLE	: Beagle Engine is an Advanced Genetic Learning Environment

TABLO LİSTESİ

		Sayfa No
Tablo 2.1.	Toplum genetiğinde bazı kavramlar	5
Tablo 2.2.	Doğadaki evrim ile problem çözme arasındaki bağlantı	5
Tablo 2.3.	Uygunluk orantılı (UO) seçim ile çizgisel sıralama (ÇS) seçimi olasılıkları	11
Tablo 2.4.	Üç bitlik Gray kodlaması	29
Tablo 3.1.	Gen tipleri ve sayısal örnekler	38
Tablo 3.2.	Gen sınıfı metotları	40
Tablo 3.3.	Kromozom sınıfı metotları	41
Tablo 3.4.	Birey sınıfı metotları	42
Tablo 3.5.	Toplum sınıfı metotları	44
Tablo 3.6.	İstatistik sınıfı metotları	47
Tablo 3.7.	Grafik sınıfı metotları	48

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1 : Evrimsel algoritmaların genel yapısı 6
Şekil 2.2 : Evrimsel algoritmaların genel akış diyagramı 8
Şekil 2.3 : Tek noktadan çaprazlaşma 13
Şekil 2.4 : İki noktadan çaprazlaşma 13
Şekil 2.5 : Homojen çaprazlaşma 14
Şekil 2.6 : Basit aritmetik çaprazlaşmanın genotip üzerinde formüllerle gösterimi 14
Şekil 2.7 : Basit aritmetik çaprazlaşmanın sayısal örnekle gösterimi 15
Şekil 2.8 : Tek aritmetik çaprazlaşmanın genotip üzerinde formüllerle gösterimi 15
Şekil 2.9 : Tek aritmetik çaprazlaşmanın sayısal örnekle gösterimi 15
Şekil 2.10 : Tüm aritmetik çaprazlaşmanın genotip üzerinde formüllerle gösterimi 16
Şekil 2.11 : Tüm aritmetik çaprazlaşmanın sayısal örnekle gösterimi 16
Şekil 2.12 : İkili mutasyon 17
Şekil 2.13 : Takas mutasyonu 18
Şekil 2.14 : Araya ekleme mutasyonu 19
Şekil 2.15 : Karıştırma mutasyonu 19
Şekil 2.16 : Tersine çevirme mutasyonu 20
Şekil 2.17 : Kısıt koşulunu sağlamayan geçersiz çözümlerin onarım akış diyagramı 26
Şekil 2.18 : Çok sayıda çanta problemine ait veriler 27
Şekil 2.19 : F1'in tanımlı olduğu aralık için fenotipler ve genotip karşılıkları 30
Şekil 2.20 : Kromozomun genotipini oluşturan yapı 31
Şekil 2.21 : F2'nin tanımlı olduğu aralık için fenotipler ve genotip karşılıkları 31
Şekil 2.22 : F4'ün tanımlı olduğu aralık için fenotipler ve genotip karşılıkları 33
Şekil 2.23 : F5'in tanımlı olduğu aralık için fenotipler ve genotip karşılıkları 34
Şekil 3.1 : PyLEA paketi ve Population alt paketi 36
Şekil 3.2 : PyLEA paketinin hiyerarşik yapısı 37
Şekil 3.3 : Gen nesnesi oluşturmak için hiyerarşik yazılım ve tamsayı tipinde oluşturulmuş bir gen nesnesinin yapısı 39
Şekil 3.4 : Kromozom nesnesi oluşturmak için hiyerarşik yazılım ve küme tipinde genlere sahip bir kromozom nesnesinin yapısı 40
Şekil 3.5 : Birey nesnesi oluşturmak için hiyerarşik yazılım ve ikili tipte genlere sahip iki kromozomun oluşturduğu birey nesnesinin yapısı 42

Şekil 3.6 : Toplum nesnesi oluşturmak için hiyerarşik yazılım ve tamsayı tipte genlere sahip iki kromozomlu bireylerin oluşturduğu toplum nesnesinin yapısı..... 43
Şekil 3.7 : PyLEA paketi kullanılarak bir maksimizasyonu probleminin basit genetik algoritma ile çözümünü aramak amacıyla geliştirilmiş kod..... 49
Şekil 4.1 : EA arayüzü ilk çalıştırıldığında oluşan pencere..... 54
Şekil 4.2 : EA arayüzünün görünüm penceresi..... 55
Şekil 4.3 : EA arayüzünün istatistik penceresi..... 56
Şekil A.1 : Pythondaki veri yapıları örnekleri..... 67
Şekil A.2 : For döngüsü örneği..... 67
Şekil B.1 : EA arayüzü ile oluşturulmuş çubuk grafiği ve çizgisel grafik örnekleri..... 68

SEMBOL LİSTESİ

p_c	: Çaprazlaşma olasılığı
p_m	: Mutasyon olasılığı
k	: Turnuva boyutu
s	: Çizgisel sıralama seçim baskısı parametresi
c	: Normalizasyon faktörü
p_i	: Sırası i olan bireyin seçilme olasılığı
q_i	: Sırası i olan bireyin kümülatif olasılığı
r	: Rasgele üretilmiş sayı
α	: Aritmetik sabiti
P_{UO}	: Uygunluk orantılı seçim olasılığı
$P_{ÇS}$	: Çizgisel sıralama seçimi olasılığı

EVRİMSEL ALGORİTMALAR İÇİN UYGULAMA VE GELİŞTİRME YAZILIM ALTYAPISI

ÖZET

Bilgisayar bilimi ve matematik için temel başlıklardan biri otomatik problem çözücülerin tasarımıdır. İnsanoğlu doğa ve doğal çözümlerden daima esinlenmiştir. En etkili doğal problem çözücüler insan beyni ve evrim işlemidir. Evrim işlemini esas alarak problem çözücüler tasarlama çalışmaları bizi evrimsel hesaplamaya yöneltir.

Bu tez PyLEA (Python Library for Evolutionary Algorithms) olarak adlandırılmış olan bir evrimsel hesaplama iskelet yapısını tanıtmaktadır. PyLEA, yüksek seviyeli, yorumlanan ve nesneye dayalı bir programlama dili olan Python kullanılarak kodlanmıştır. PyLEA'nın Python ile tasarlanması platformdan bağımsız ve esnek bir yapı oluşturulabilmesini sağlamıştır. Aynı zamanda Python tasarım sürecini kısaltmış ve tasarımın kolay bir biçimde gerçeklenebilmesini sağlamıştır. PyLEA paketinde evrimsel algoritmaya ilişkin istatistiksel verilerin hesaplanıp kaydedilebilmesini sağlayan Log adında bir alt paket bulunmaktadır. Diğer bir alt paket olan Graphics paketi ise bu kaydedilen verileri kullanarak verilerin çubuk grafikleri veya çizgisel grafikler üzerinde incelenebilmesini sağlar. Grafiklerin Adobe PDF dosyaları içerisinde çizilmesinde Reportlab kütüphanesi kullanılmıştır. Reportlab kütüphanesinin seçilmesinin sebebi kütüphane kullanılarak raporların hızlı bir şekilde oluşturulabilmesi ve verimli grafik özelliklerine sahip olmasıdır. Bunların yanısıra PyLEA paketi için bir arayüz tasarlanmıştır. Arayüzün tasarımında Tk kütüphanesi kullanılmış olması arayüzün kullanımı kolay ve platformdan bağımsız olmasını sağlamıştır.

PyLEA evrimsel algoritmalar için uygulama ve geliştirme yazılım altyapısıdır. PyLEA içerisinde evrimsel operatörler ve operatörlere ait metotlar tanımlanmıştır. Ayrıca PyLEA paketinde bazı örnek problemler de tanımlanmıştır. Bir toplum ve bireylere sahip olunduğunda pakette tanımlanmış operatörler ile birlikte kullanıcı tarafından tanımlanmış operatörler istenilen sırada toplumdaki bireyler üzerinde uygulanabilmektedir. Böylece pakette tanımlanmış olan örnek problemler için ya da kullanıcı tarafından tanımlanan problemler için çözüm evrimsel algoritmalarla aranabilir. Bir kullanıcı tanımladığı problemleri, operatör ve metotları pakete dahil edebilmekte ve sonrasında bunları pakette tanımlanmış operatör ve metotlar ile birlikte toplumun bireylerine uygulayabilmektedir. Bu PyLEA yapısının esnekliği için bir örnek teşkil etmektedir. PyLEA yapısında tanımlanmış gen, kromozom, birey, toplum v.b. gibi yapılar bulunmaktadır. Kullanıcı hiçbir kısıtlama olmadan bu yapılara başka yapılar ekleyerek yapıları genişletebilir.

PyLEA'nın kullanımı kolay olan evrimsel algoritma (EA) arayüzü eğitim, uygulama ve geliştirme gibi çeşitli amaçlarda kullanılabilir. Eğitim amaçlı kullanımlarda kullanıcı pakette tanımlanmış olan örnek problemlerden birinin çözümü için bireylere uygulanacak, pakette tanımlanmış operatörler ve metotlarını,

sonrasında da bir izleme seçeneğini seçerek algoritmayı koşturur. Seçilen izleme seçeneği algoritmanın belirli aralıklarla donmasını sağlar. Algoritma donduğunda kullanıcı toplumdaki bireyleri, ebeveyn ve çocukları ayrıca varolan nesil için istatistiksel verileri inceleyebilir. Eğitim amaçlı kullanımlar herhangi bir kod yazımını gerektirmez. Tüm adımlar birkaç fare tıklaması ile gerçekleştirilir. Pakette tanımlanmış örnek problemlerden farklı olarak evrimsel algoritmalar ile çözümünün bulunmasını istediği bir probleme sahip olan kullanıcı, probleme ilişkin uygunluk fonksiyonunu yükledikten sonra paketteki operatör ve metotları seçerek algoritmayı koşturabilir. Algoritma sonlandıktan sonra, algoritma süresince hesaplanmış istatistiksel veriler ve bulunan en iyi çözüm kaydedilip incelenebilir. Bu gibi uygulama amaçlı kullanımlar çözümü aranan problemin uygunluk fonksiyonuna ilişkin bir kodun sisteme yüklenmesini gerektirir. Geliştirme amaçlı kullanımlar akademik araştırmacılara hizmet sunmaktadır. Bu kullanım şeklinde kullanıcı geliştirdiği operatörü yükler, pakette tanımlanmış bir örnek problemi seçer ve evrimsel algoritmayı koşturur. Toplumdaki değişimlerin incelenebilmesi için izleme seçeneği kullanılabilir. Evrimsel algoritma çok sayıda koşturulup her adım için istatistiksel veriler incelenebilir.

Diğer iskelet yapılar arasında PyLEA kullanımı kolay arayüzü, platformdan bağımsız ve esnek yazılımı, öğrenimi ve kullanımı kolay yapısı sayesinde üstünlük göstermektedir.

Bu tezde evrimsel algoritmalar, PyLEA paketinde tanımlanmış operatörler ve metotları ve örnek problemler hakkında bilgi verilmiştir. Ayrıca tezde PyLEA paketi tanıtılmış, PyLEA'da varolan bazı önemli yapılar hakkında bilgi verilmiş ve paket yapısı başka yapılarla karşılaştırılmıştır. Bunların yanısıra PyLEA için tasarlanmış olan EA arayüzü de tanıtılmıştır. Arayüzün bazı önemli seçenek ve özellikleri açıklanmış ve farklı kullanım amaçlarının nasıl gerçekleştirilebileceği tarif edilmiştir.

PyLEA paketi kurulduktan sonra kullanıcı kişisel bilgisayarında evrimsel algoritmaları koşturabilir. Gelecekte PyLEA için internet tabanlı bir arayüz tasarlanması amaçlanmıştır. Böylece kullanıcılar evrimsel algoritmaları çevrimiçi (online) olarak web göz atıcısı (web browser) aracılığıyla koşturabileceklerdir. Uzun vadede ise internet tabanlı bir problem çözme platformu tasarlanması amaçlanmaktadır.

Sonuç olarak bu tezde esnek, çok yönlü kullanım sağlayan, platformdan bağımsız, kullanımı kolay PyLEA evrimsel hesaplama paket yapısı ile platformdan bağımsız ve kullanımı kolay EA arayüzü tanıtılmıştır. EA arayüzü ile birlikte PyLEA farklı kullanıcı seviyelerine hizmet sunabilmektedir. Programlama bilgisi olmayan kişiler evrimsel algoritmaları EA arayüzü üzerinde birkaç fare tıklaması yaparak koşturabilirler. Diğer yandan kullanıcılar, yazıp sisteme yükledikleri kodlarla kendi problemlerini tanımlayıp problemin evrimsel algoritmalar ile çözümünü arayabilirler. Evrimsel algoritma bilgisi az olan kişiler bir izleme seçeneği seçip evrimsel algoritma her donduğunda EA arayüzünün görünüm özelliğini kullanarak toplumun bireyleri, ebeveyn ve çocukları ile o nesile ait istatistiksel verileri inceleyerek kendilerini geliştirebilirler. İstatistik ve görünüm özellikleri kendi operatörlerini geliştiren kişiler gibi evrimsel algoritmalar konusunda daha deneyimli kişilerce de kullanılabilir. EA arayüzü bu kişilerin operatörlerini yükledikten sonra evrimsel algoritmayı istedikleri kez tekrar koşturmalarını ve her adım için istatistiksel verileri inceleyip karşılaştırabilmelerini sağlamaktadır.

APPLICATION AND DEVELOPMENT FRAMEWORK FOR EVOLUTIONARY ALGORITHMS

SUMMARY

One of the central topics in computer science and mathematics is the development of automatic problem solvers. Nature and natural solutions have always been a source of inspiration for the human being. The most powerful natural problem solvers are the human brain and the evolutionary process. Trying to design problem solvers based on the evolutionary process leads us to evolutionary computing.

This thesis introduces a new evolutionary computation framework named PyLEA (Python Library for Evolutionary Algorithms). PyLEA is a package coded in Python, a high level, interpreted and, object oriented programming language. Designing PyLEA with Python provided us a flexible structure and a platform independent software. It also shortened design time and made the design simpler. The package contains a subpackage named Log which enables calculating and saving the statistical data that belongs to the evolutionary algorithm. Another subpackage, Graphics package, uses this saved data and enables users study the statistical data on bar or line charts. Reportlab library is used for creating the graphics in Adobe PDF files. Reportlab library was choosen for high report generation speed and efficient graphic features. Also an interface was designed for the PyLEA package. Using Tk library for the design provided a user friendly, platform independent interface.

PyLEA is a software package for application and development of evolutionary algorithms. Evolutionary operators and operators' methods are defined in PyLEA. Also some benchmarking problems are defined in the PyLEA package. Having a population of individuals it is possible to apply a sequence of package's operators and methods together with user defined operator and methods to the individuals in order to solve a problem defined in PyLEA or a user defined problem. A user can import problems, operators and methods he/she defined inside the package and apply them to the individuals together with the package defined ones. This is an example for PyLEA's flexibility. There are structures defined in PyLEA like gene, chromosome, individual, population etc. Users can expand these structures by adding other structures inside, without a constraint.

The user friendly evolutionary algorithm (EA) interface of PyLEA can be used for multiple purposes like education, application and development. In educational usage user selects the operators and methods defined in the package in order to solve a problem defined in the package, then user selects a trace option and runs the algorithm. Selected trace option makes the algorithm pause at some intervals. When the algorithm pauses user can study the individuals, parents, offspring of the population and also the statistical data for that generation. Educational usage does not require writing a code. All steps can be done with a few mouse clicks. Having a problem (different than the ones defined in the package) to be solved with evolutionary algorithms one can load the fitness function for the problem, select

operators and methods to be applied and run the algorithm. When the algorithm ends statistical data calculated during the algorithm and best result found can be saved and studied. Applicational usage like this requires the code for a fitness function to be loaded. Development usage serves to academic researchers. In this type of usage user loads the operator that he/she developped, chooses a problem defined in the package and runs the evolutionary algorithm. Trace options can be used to study the changes in the population. It is possible to run the evolutionary algorithm many times and study the statistical data for each run.

Among other frameworks PyLEA is better with user friendly interface, platform independent and flexible software, easy to learn and use structure.

This thesis gives information about the evolutionary algorithms, operators and their methods and also benchmarking problems defined in PyLEA. The thesis also introduces PyLEA, some important structures PyLEA has and compares PyLEA with other frameworks. The EA interface designed for PyLEA is also introduced. Some important options are explained and different usage purposes are described.

After downloading and setting up PyLEA a user can run evolutionary algorithms on his/her personal computer. As a future work a web based interface is aimed to be designed for PyLEA so that users will be able to run evolutionary algorithms online, through a web browser. In the long run designing a web based problem solving platform is aimed.

All in all, in this thesis PyLEA, a flexible, versatile, platform independent, easy to use evolutionary computation framework is introduced together with the platform independent and user friendly EA interface. PyLEA together with EA interface can serve to multiple user levels. People with no programming knowledge will be able to run evolutionary algorithms just by a few mouse clicks using EA interface. On the other hand people will also be able to write and load their codes to PyLEA in order to define and then solve problems with evolutionary algorithms. People with low evolutionary algorithm knowledge will be able to improve themselves by selecting trace options, viewing population's individuals, parents, offspring and studying statistical data when the algorithm paused, on the EA interface. Statistics and view features can also serve to people that are experienced with evolutionary algorithms, like the ones that generated their own operators. EA interface enables them to run evolutionary algorithm that contains the generated operator(s) as much as they want, study and compare the statistics for each run.

1. GİRİŞ

Yapılan çalışmada evrimsel algoritmalar için uygulama ve geliştirme yazılım altyapısı olan PyLEA paketi ve paket kullanımı için bir arayüz tasarlanmıştır. Bu çalışma daha geniş kapsamlı bir projenin parçasıdır. Bu projede internet üzerinden bir optimizasyon çözüm platformu olarak hizmet verecek bir yapı tasarlanması düşünülmüştür. Kullanıcının geliştirilecek olan XML tabanlı bir dil sayesinde optimizasyon problemini belirtmesi ve problemine ilişkin çözümü elde edebilmesi düşünülmektedir.

PyLEA paketini geliştirmede Python dili kullanılmıştır. Python yüksek seviyeli, yorumlanan ve nesneye dayalı bir programlama dilidir. Python dili projeye esneklik katmış, yazılımın platformdan bağımsız olmasını sağlamış, ayrıca projenin gerçeklenme süresini kısaltmış ve tasarımı kolaylaştırmıştır. Python dilinin öğrenimi kolaydır ve dil yapıya kullanım kolaylığı sağlamıştır. PyLEA paketinde ayrıca evrimsel algoritmaya ilişkin bazı istatistiksel verilerin kaydedilmesi ve bu verilerin grafikler üzerinde incelenebilmesi imkanı da sunulmuştur. Grafiklerin oluşturulmasında ise Reportlab kütüphanesi kullanılmıştır. PyLEA için geliştirilen arayüz Python dili ile ve Tk kütüphanesi kullanılarak tasarlanmıştır. Arayüz sayesinde programlama bilgisi olmayan kullanıcılar da kendi evrimsel algoritma yaklaşımlarını oluşturabilmektedir.

PyLEA paket yapısı oldukça esnektir. Paket yapısında bazı örnek problemler, operatörler ve pek çok metot tanımlanmıştır. Kullanıcı yapıda tanımlanmış olan örnek problemler, operatörler ve metotların yanısıra tanımlayacağı kendi problemini, geliştirdiği operatörler ve metotlarını yapıya yükleyerek kendi evrimsel algoritma yaklaşımını oluşturabilir. Ayrıca kullanıcı pakette gerçeklenmiş olan gen, kromozom, birey, toplum gibi sınıfları kullanarak oluşturacağı nesnelere başka nesneler ilave ederek yapıyı genişletebilir ve bu sınıfların metotlarına tanımlayacağı yeni metotları dahil edebilir.

PyLEA paket yapısının kullanımı için geliştirilen arayüz evrimsel algoritma (EA) arayüzü olarak adlandırılmıştır. Evrimsel algoritma arayüzü sayesinde yazılım bilgisi olmayan kişiler , hatta evrimsel algoritmalar hakkında çok az bilgi sahibi olan kişiler de kendi evrimsel algoritma yaklaşımını oluşturabilmektedir.

PyLEA paket yapısı ve evrimsel algoritma arayüzü ile gerçekleştirilebilecek kullanım şekilleri temelde üç amaç başlığı altında gruplanmıştır. Her kullanım amacı için evrimsel algoritma arayüzünün nasıl kullanılabileceği aşağıda açıklanmıştır :

1. Eğitim amaçlı kullanımlar : Kullanıcı sadece birkaç fare tıklaması yaparak oluşturacağı evrimsel algoritmanın adım adım koşmasını sağlayabilir ve her adımın sonunda meydana gelen değişimleri, oluşan çözüm adaylarını ve bazı istatistiksel verileri inceleyebilir. Bu durumda herhangi bir yazılım bilgisine de gerek yoktur.
2. Uygulama amaçlı kullanımlar : Kullanıcı evrimsel algoritmalar ile çözümünün bulunmasını istediği problem için çözüm adaylarının uygunluk derecelerini belirleyecek bir uygunluk fonksiyonu kodu yazar ve yapıya yükler. Daha sonra istediği operatörler ve metotlarını seçerek algoritmayı çalıştırır. Algoritmanın sonunda probleme ilişkin sonuçları inceler ya da kaydeder.
3. Geliştirme amaçlı kullanımlar : Kullanıcı seçtiği bir örnek problem üzerinde kendi geliştirdiği operatörünü inceler. Kullanıcı kendi operatörünü diğer operatörler ile kıyaslayabilir veya operatörüne ilişkin istatistiksel verileri inceleyebilir.

Kullanıcıların kendi evrimsel algoritma yaklaşımlarını oluşturmalarına yardımcı olacak farklı dillerde yazılmış yapılar mevcut olsa da PyLEA paketi kullanım kolaylığı ve esnekliği sayesinde bu yapılardan üstünlük göstermektedir.

Tezin ikinci bölümünde evrimsel algoritmalar tanıtılmış ve paket yapısında gerçeklenmiş olan örnek problemler, operatörler ve metotları açıklanmıştır. Üçüncü bölümde PyLEA paket yapısı tanıtılmıştır. Pakette tanımlanmış sınıflar ve metotları açıklanmış, paketin kullanımına ilişkin bir örnek verilmiş ve açıklanmış, evrimsel algoritma yaklaşımlarının geliştirilmesini sağlayan benzer yapılar tanıtılmış, son olarak da PyLEA yapısının avantaj ve dezavantajları üzerinde durulmuştur. Dördüncü bölümde EA arayüzü tanıtılmış ve kullanım şekilleri açıklanmıştır. Beşinci bölümde sonuçlar tartışılmıştır. Altıncı bölümde ise gelecekte yapılabilecek çalışmalar tanıtılmış ve açıklanmıştır. Python dili, Reportlab ve Tk kütüphaneleri ile ilgili genel bilgiler eklerde sunulmuştur.

2. EVRİMSEL ALGORİTMALAR

2.1 Giriş

Bu bölümde öncelikle evrim kavramı tanıtılarak evrimsel algoritmalar ile bağlantısı açıklanmış ve evrimsel algoritmalar hakkında genel bilgi sunulmuştur. Daha sonra tez çalışmasında gerçeklenmiş evrimsel operatörler tanıtılmış ve operatörlerin metotları anlatılmıştır. Son olarak tez çalışmasında gerçeklenmiş olan örnek problemler, problemlerin evrimsel algoritmalar ile çözümünde kullanılan uygunluk fonksiyonları ve çözüm uzayındaki noktaların evrimsel araştırmanın gerçekleştiği uzaydaki noktalara dönüşümlerinin nasıl gerçekleştiği gibi problem çözümüne ilişkin bilgiler açıklanmıştır.

2.2 Evrimsel Algoritmaların Tanıtımı

Organizmaların zaman içerisinde yavaş yavaş geliştiği uzun zamandır bilinmekteyse de Charles Darwin birbirini destekleyen çok sayıda kanıtı bir araya getirip evrimin meydana gelmesi ile ilgili gerçekçi bir işleyişi tanıtan ilk kişidir. Modern evrim teorisinin kurucusu olan Darwin, Beagle adlı bir gemide beş yıl sürecek bir yolculuk yapmış, yolculuk süresince yaptığı araştırmalar ve gözlemleri sonucunda edindiği bilgilerden faydalanarak yarattığı teorisini "Türlerin Kaynağı" adlı kitabında tanıtmıştır. Kitapta evrimin işleyişi ile ilgili beş temel kavram bulunmaktadır :

1. Organizmalar birbirlerine benzer organizmaları oluştururlar. Diğer bir deyişle üreme işleminde stabilite vardır,

2. Çoğu tür için bir nesilde hayatta kalıp üreyen bireylerin sayısı başlangıçta doğmuş olan birey sayısından küçüktür,

3. Herhangi bir toplumda bireylerin arasında varyasyonlar bulunur ve bazı varyasyonlar kalıtsal olabilir,

4. Hangi bireylerin hayatta kalıp üreyeceği ve hangi bireylerin hayatta kalamayacağı varyasyonlar ile çevre arasındaki etkileşimden kaynaklanan bir derece ile belirlenir.

Bazı varyasyonlar bireylerin diğerlerinden daha uzun süre hayatta kalıp üremesini sağlar. Darwin kalıtsal olarak gelen bu varyasyonların bir nesilden diğerine geçişte daha yaygın olarak görülme eğiliminde olduğunu iddia etmiştir. Darwin tarafından bu işleme *doğal seçim* adı verilmiştir.

5. Yeterince zaman tanındığında doğal seçim bir organizma grubunu başka bir organizma grubuna çevirecek değişim birikimine yol açabilir.

Darwin'in teorisi tarafından yanıtlanamayan üç soru vardı :

1. Miras alınan karakteristik özellikler bir nesilden diğerine nasıl geçer ?

2. Miras alınan karakteristik özellikler bir nesilde yok olmuşken nasıl sonraki nesillerde tekrar ortaya çıkabilmektedir ?

3. Doğal seçimde rol oynayan varyasyonlar nasıl meydana gelmektedir ?

Bu soruların yanıtları Gregor Mendel tarafından yapılmış olan genetik çalışmaları ile bulunabildi. Mendel prensipleri ile Darwin'in evrim teorisinin birleşimi yeni bir biyoloji dalını, toplum genetiğini oluşturdu [1]. Evrim teorisi ve Mendel tarafından yapılan genetik çalışmaları sonucunda oluşan bazı kavramları ve anlamlarını Tablo 2.1'de görebilirsiniz.

Evrim, gen havuzundaki herhangi bir değişim olarak tanımlanır. Doğal evrim prosesinden esinlenilmesi sonucunda bilgisayar biliminde yeni bir araştırma alanı, evrimsel hesaplama, oluşmuştur. Problem çözmede (optimizasyon problemleri, modellemeler, simülasyonlar vb.) evrimsel hesaplamalardan faydalanılabilmektedir. Doğal evrim ile evrimsel hesaplamalar arasındaki ilişki şu şekilde açıklanabilir: Doğal evrimde bir toplum tarafından doldurulmuş bir çevre bulunmaktadır. Toplum, amaçları hayatta kalmak ve üremek olan bireylerden oluşmaktadır. Bu bireylerin uygunluğu çevre tarafından belirlenir ve bireyin amaçlarını gerçekleştirmedeki başarısı ile bağlantılıdır. Problem çözme işleminde ise çözüm adaylarından oluşan bir grup bulunur. Burada uygunluk çözüm adaylarının kalitesi yani problemi ne kadar iyi çözdükleri ile bağlantılıdır. Uygunluk, çözüm adayının tutulması ve bu adayın başkaca çözüm adayları oluşturulmasında bir tohum olarak kullanılması ihtimalini belirler [2]. Tablo 2.2'de doğal evrim ile problem çözme arasındaki bağlantı görülmektedir.

Tablo 2.1 Toplum genetiğinde bazı kavramlar

EVRİM TEORİSİ	
Toplum	: Aynı zamanda belirli bir alanı işgal eden aynı türden bireyler grubu
Uyum Sağlama	: Bir grup organizmanın kendi çevrelerinde yaşamaya ve üremeye daha uygun hale gelmesini sağlayan özelliklerin evrimi
Uygunluk	: Hayatta kalmak ve evlat bırakmak için bağıl yetenek
Doğal Seçim	: En uygunun hayatta kalışı
KLASİK MENDEL GENETİĞİ	
Kalıtım	: Genetik materyalin ebeveynden evlatlara yayılması
Gen	: Kalıtımın en küçük birimi
Gen Havuzu	: Toplumdaki tüm bireylerin genlerinin toplamı
Kromozom	: Hücre çekirdeğinde, üzerinde genlerin bulunduğu gövdelerden biri
Haploid	: Bir çift kromozoma sahip olmak
Diploid	: İki çift kromozoma sahip olmak
Genotip	: Bir hücre ya da organizmanın genetik yapısı
Fenotip	: Bir organizmanın gözlemlenebilen özellikleri
Mutasyon	: Bir genin bir formdan diğerine kalıtımsal olarak değişimi
Yeniden Birleşim	: Çiftleşme vasıtasıyla ebeveyn kromozomlarındakinden farklı bir gen birleşimi oluşturma
Çaprazlaşma	: Eşleşmiş kromozomlar arasında genetik materyalin değiş tokuşu

Tablo 2.2 Doğadaki evrim ile problem çözme arasındaki bağlantı

EVRİM		PROBLEM ÇÖZME
Çevre	↔	Problem
Birey	↔	Çözüm Adayı
Uygunluk	↔	Uygunluk

Evrimsel hesaplamalarda kullanılan evrimsel algoritmaların farklı türleri vardır. Tüm tekniklerde ortak bir düşünce bulunur. Bir toplumun bireyleri üzerinde çevresel baskı doğal seçime neden olur ve en uygun bireyler hayatta kalır. Bu da toplumun uygunluğunda bir artışa neden olur. Maksimum yapılmak istenen bir uygunluk fonksiyonu verildiğinde rasgele bir çözüm adayları kümesi yaratılır. Çözüm adayları fonksiyonun değer kümesinin elemanları olacaktır. Bu kümenin elemanlarına

uygunluk ölçüsüne karşılık gelecek biçimde uygunluk fonksiyonu uygulanır. Daha büyük uygunluğa sahip çözüm adayları daha iyidir. Uygunluğuna bağlı olarak daha iyi olan çözüm adayları bir sonraki nesil için tohum olmak üzere seçilir. Daha sonra çözüm adaylarına çaprazlaşma ve/veya mutasyon uygulanır. Çaprazlaşma ve mutasyon uygulaması yeni bir çözüm adayı kümesi (çocuklar) oluşmasına sebep olur. Oluşan çocuklar uygunluklarına bağlı olarak yeni nesilde bir yer edinebilmek için ebeveynleriyle rekabet ederler. Bu işlem yeterli bir uygunluğa (kaliteye) sahip bir çözüm adayı bulunana ya da önceden belirlenmiş bir işlemsel sınıra ulaşılıncaya kadar devam edebilir. Evrimsel algoritmaların genel yapısı Şekil 2.1'de gösterilmiştir.

```
BAŞLA
        Topluma Rasgele Çözüm Adayları Ekle
        Her Çözüm Adayı İçin Uygunluğu Hesapla
        Sonlanma Koşulu Sağlanana Kadar Tekrarla :
            1. Ebeveynleri seç
            2. Ebeveynleri çaprazla
            3. Oluşan çocukları mutasyona uğrat
            4. Yeni oluşmuş çözüm adayları için uygunluğu hesapla
            5. Sonraki nesil için bireyleri seç
BİTİR
```

Şekil 2.1 Evrimsel algoritmaların genel yapısı

Bir evrimsel algoritmanın oluşturulmasında belirtilmesi gereken bileşenler, işlemler ya da operatörler bulunur. Bunlardan Şekil 2.1'de de görülebilen en önemlileri aşağıda açıklanmıştır [2].

1. Gösterim : Problem bağlamında çözüm adayı, fenotip ve birey terimleri olası çözümler uzayındaki noktaları belirtirler. Olası çözüm uzayının kendisi ise fenotip uzayı olarak adlandırılır. Evrimsel algoritmalarda ise genotip, kromozom ve birey evrimsel araştırmanın gerçekleştiği uzaydaki noktaları belirtir. Bu uzay ise genotip uzayı adını alabilmektedir. Gösterim kelimesi fenotip uzayından genotip uzayına dönüşüm anlamında kullanılmaktadır. Fenotipleri temsil eden genotiplere dönüşüm gerçekleştirilir. Kısacası gösterim, kodlama ile aynı anlamı ifade etmektedir. Gösterim işlemi tersine çevrilebilir, ki bu da kod çözme işlemine karşılık gelir. Her genotipe karşılık gelen bir fenotip bulunur.

2. Uygunluk fonksiyonu : Uygunluk fonksiyonu, ya da değerlendirme fonksiyonu, genotiplere bir uygunluk (kalite) ölçüsü atamakta kullanılır.

3. Toplum : Toplumun görevi olası çözümlerin gösterimlerini (genotiplerini) tutmaktır. Bir toplum genotiplerden oluşan, aynı genotipin birden fazla sayıda olabileceği bir çoğul kümedir.

4. Ebeveyn seçimi : Ebeveyn seçiminin rolü bireyleri uygunluklarına dayanarak ayırt etmek, daha iyi bireylerin sonraki nesil için ebeveyn olmasına izin vermektir. Ebeveyn seçimi hayatta kalış seçimi ile birlikte uygunluk arttırımına sevketmekle sorumludur. Ebeveyn seçiminde yüksek uygunluğa sahip bireylerin ebeveyn olma ihtimali düşük uygunluğa sahip bireylere göre yüksektir. Ancak düşük uygunluğa sahip bireylerin az da olsa ebeveyn olma şansı vardır. Aksi taktirde araştırma yerel optimumda takılabilir.

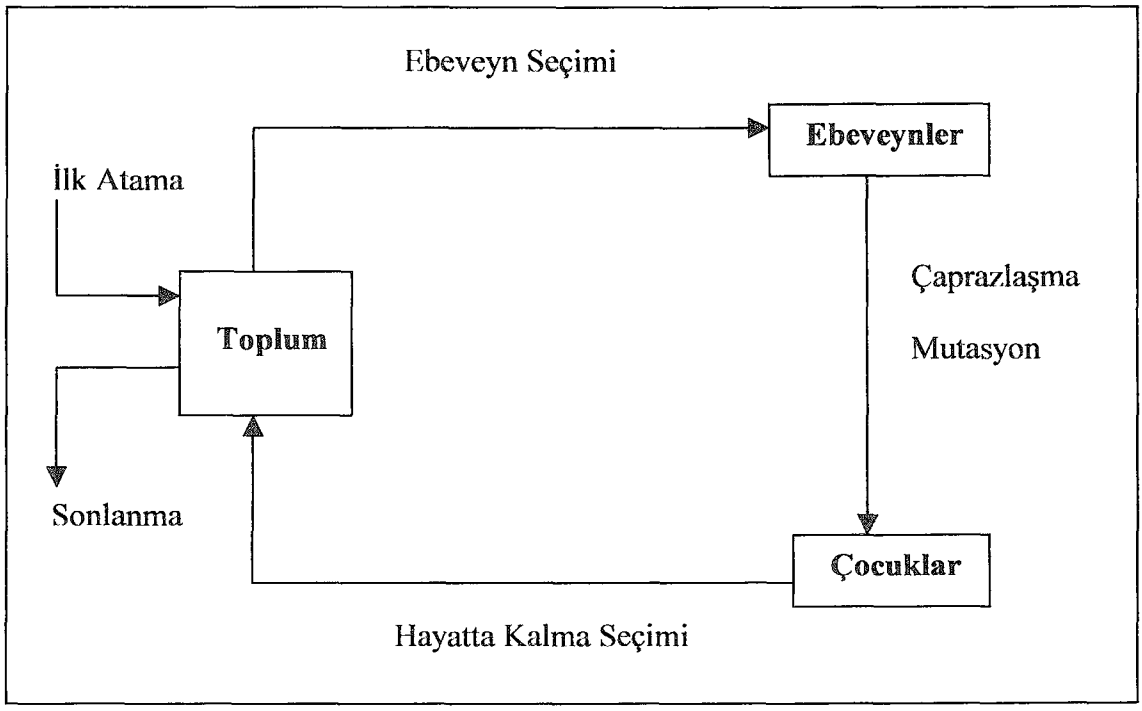
5. Varyasyon operatörleri : Varyasyon operatörleri eski bireylerden yeni bireyler meydana getirirler. Bu, fenotip uzayında yeni çözüm adayları meydana getirmeye eşdeğerdir. Varyasyon operatörleri çaprazlaşma ve mutasyondur. Çaprazlaşma operatörü ebeveyn iki bireyin genotiplerindeki bilgileri bir ya da iki çocuk genotipinde birleştirir. Mutasyon operatörü ise bir genotipe uygulanır ve değişime uğramış mutant bir çocuk oluşturur. Çocuk bir rasgele seçim dizisi sonucu meydana gelir.

6. Hayatta kalış seçimi : Hayatta kalış mekanizması seçilmiş ebeveynlerin çocukları oluştuktan sonra gerçekleştirilir. Toplumda çocuklar oluştuktan sonra hangi bireylerin sonraki nesilde bulunmasına izin verileceği kararlaştırılmalı, bir seçim yapılmalıdır. Hayatta kalış seçiminin rolü uygunluklarına göre bireyler arasından seçim yapılmasını sağlamaktır.

7. İlk atama : Topluma rasgele bireyler, çözüm adayları, atanmasıdır. İlk toplumda rasgele oluşturulmuş bireyler tohum verir.

8. Sonlanma koşulu : Evrimsel algoritmanın ne zaman sonlanacağını belirler. Örneğin optimum çözümün bilindiği durumlarda olası bir sonlanma koşulu uygunluk değeri optimuma ulaşan birey bulunduğunda algoritmanın sonlanması şeklinde olabilir.

Şekil 2.2' de bir evrimsel algoritmanın bileşenleri ve akış diyagramı görülmektedir.



Şekil 2.2 Evrimsel algoritmaların genel akış diyagramı

2.3 Gerçeklenen Operatörler

Bu bölümde PyLEA paket yapısında gerçeklenmiş olan evrimsel algoritma operatörleri tanıtılmış ve metotları açıklanmıştır.

2.3.1 Ebeveyn Seçimi

Ebeveyn seçimi metotları ile toplumdaki birey sayısı kadar birey, çocukların oluşmasını sağlamak üzere rasgele biçimde seçilir. Seçim rasgele olsa da daha uygun bireylerin seçilme şansı daha yüksektir. Ebeveynleri oluşturmak üzere seçilmiş her bireyin bir kopyası eşleşme havuzuna aktarılır. Toplumdaki bireyler sabit olacağından aynı bireyin birden fazla kez seçilmesi ve böylece bireyin birden fazla kopyasının eşleşme havuzuna aktarılması mümkündür. Diğer bir deyişle eşleşme havuzunda aynı genotipe sahip birden fazla sayıda birey bulunabilmektedir.

2.3.1.1 Turnuva Seçimi

Toplumdaki k sayıda birey rasgele biçimde seçilir. Seçilmiş olan k sayıda birey arasından en uygun, en iyi birey bulunur ve bir kopyası eşleşme havuzuna aktarılır, seçilmiş bireylerin hepsi topluma iade edilir. Seçilecek birey sayısının, k sayısının, arttırılması seçim baskısını arttırır. Bu durumda daha uygun, daha iyi bireyler

eşleşme havuzunu doldurur. Çoğu uygulama tarafından kabul gören tipik k değeri, k = 2 olmaktadır. İşlem toplumdaki birey sayısı kadar tekrar edilir. Böylece eşleşme havuzuna toplumdaki birey sayısı kadar birey ebeveynleri oluşturmak üzere kopyalanmış olur. Seçilmiş k sayıda birey topluma iade edildiğinden önceden seçilmiş olan bireylerin yeniden seçilmesi ihtimali vardır. Böylece ebeveynleri oluşturacak bireyler arasında, yani eşleşme havuzunda, birden fazla sayıda aynı genotipe sahip birey bulunabilir [2-5].

2.3.1.2 Rulet Çarkı Seçimi

Seçim metotları arasında en kolayı ve en sık kullanılanı rulet çarkı seçimidir. Bu metot dilimlerinin genişliği bireylerin uygunluğu ile doğru orantılı olan bir rulet çarkının çevirilmesi ile benzerlik göstermektedir. Metotta öncelikle bireylerin hesaplanmış olan uygunluk değerleri toplanarak toplumun toplam uygunluğu bulunur. Her birey için seçilme olasılığı bireyin uygunluk değerinin toplumun toplam uygunluk değerine bölünmesi ile hesaplanır. Daha sonra her bireyin kümülatif olasılığı $q_i = \sum_{j=1}^{i} p_j$ olacak şekilde hesaplanır. Formülde q, i numaralı bireyin kümülatif olasılığını, p ise j numaralı bireyin seçilme olasılığını ifade etmektedir. Seçim işlemi rulet çarkının toplumdaki birey sayısı kez çevirilmesi ile gerçekleşir. Her çevrimde öncelikle [0-1] aralığında rasgele bir sayı üretilir. Rasgele üretilmiş olan bu sayı ilk bireyin kümülatif olasılığından küçük ise ilk birey eşleşme havuzuna kopyalanır. Aksi taktirde $q_{i-1} < r \leq q_i$ (r, rasgele sayıyı belirtmektedir) koşulunu sağlayan i numaralı birey (i > 2) eşleşme havuzuna kopyalanır. Çark her çevirildiğinde eşleşme havuzuna kopyalanmış bir birey eklenir. İşlemin toplumdaki birey sayısı kadar tekrarlanması sonucunda, eşleşme havuzunda toplumdaki birey sayısı kadar ebeveynleri oluşturacak birey birikmiş olur. Dikkat edilmelidir ki eşleşme havuzunda aynı genotipe sahip birden fazla birey bulunabilir [2-5].

2.3.1.3 Stokastik Evrensel Örnekleme (SUS)

SUS seçimi rulet çarkı seçimi ile benzerlik göstermektedir. SUS seçimi kullanıldığında tıpkı rulet çarkı seçiminde olduğu gibi öncelikle toplumun toplam uygunluğu hesaplanır. Daha sonra her birey için seçilme olasılığı bireyin uygunluk değerinin toplam uygunluk değerine bölünmesi ile hesaplanır. Bu adımdan sonra bireylerin seçilme olasılığı kullanılarak rulet çarkı seçimindeki gibi bireylerin

kümülatif olasılıkları hesaplanır. Buraya kadar rulet çarkı seçimi ile SUS seçimi aynı adımları gerçekleştirmiştir. Rulet çarkından farklı olarak SUS seçiminde N toplumdaki birey sayısı olmak üzere adım = 1/N hesaplanır. Daha sonra [0-adım] aralığında rasgele bir sayı üretilir. Rasgele sayı üretildikten sonra eşleşme havuzu toplumdaki birey sayısı kadar bireyle dolana kadar devam edecek olan bir döngüye girilir. Toplumdaki en küçük kümülatif olasılığa sahip ilk bireyden başlanarak rasgele üretilmiş sayı bireyin kümülatif olasılığı ile karşılaştırılır. Rasgele üretilmiş olan sayı bir bireyin kümülatif olasılığından küçük ya da eşitse bireyin bir kopyası eşleşme havuzuna eklenir ve rasgele üretilmiş sayı bir adım ilerletilir (rasgele üretilmiş olan sayıya adım sayısı eklenir). Sayı halen bireyin kümülatif olasılığından küçükse bireyin bir kopyası daha tekrar eşleşme havuzuna eklenir ve sayı bir adım daha ilerletilir. Bu işlem sayı bireyin kümülatif olasılığını aşana kadar devam eder. Rasgele üretilmiş olan sayı bir bireyin kümülatif olasılığını aşmışsa sonraki bireye geçilerek aynı işlemler tekrarlanır [2-5].

2.3.1.4 Çizgisel Sıralama

Sıralamaya dayalı seçimlerde bireylerin kümülatif olasılıkları rulet çarkı seçiminde olduğu gibi seçilme olasılıkları kullanılarak hesaplanır. Yine rulet çarkı seçiminde olduğu gibi eşleşme havuzuna kopyalanacak her birey için rasgele bir sayı üretilir ve rasgele sayının bulunduğu kümülatif olasılıklar aralığına bakılarak gerekli birey seçilip kopyası eşleşme havuzuna atanır. Bu metotların rulet çarkı seçiminden farkı seçilme olasılıklarının hesaplanmasında görülmektedir. Toplumdaki bireyler uygunluk derecelerine göre sıralanırlar ve seçim olasılığı bu sıralamaya dayanarak hesaplanır. Sıralamaya dayalı seçimlerde seçim baskısı sabittir. Sıralamaya dayanarak seçim olasılığının hesaplanması çizgisel ya da üstel metotlar kullanılarak gerçekleştirilebilir.

Çizgisel sıralamada $1.0 < s \leq 2.0$ olmak üzere bir s, seçim baskısı, parametresinden yararlanılır. Öncelikle toplumdaki bireyler uygunluk derecelerine göre sıralanırlar. Toplumdaki birey sayısının m olduğunu farzedersek en iyi bireyin sırası m, en kötü bireyin sırası ise 1 olacaktır. Sırası i olan bir bireyin seçim olasılığı aşağıdaki denkleme göre hesaplanabilir :

$$p(i) = \frac{(2-s)}{m} + \frac{2i(s-1)}{m(m-1)} \qquad (2.1)$$

Bireylerin seçim olasılıkları hesaplandıktan sonra rulet çarkı seçiminde olduğu gibi $q_i = \sum_{j=1}^{i} p_j$ olacak şekilde kümülatif olasılıklar hesaplanır. Bundan sonraki işlemler rulet çarkı ile aynıdır. Eşleşme havuzu toplumdaki birey sayısına eşit sayıda birey ile dolana kadar kopyalanacak her bireyin seçimi için bir rasgele sayı üretilir. Rasgele üretilmiş sayı ilk bireyin kümülatif olasılığından küçük ise ilk bireyin, aksi taktirde rasgele üretilen sayı $q_{i-1} < r \leq q_i$ koşulunu sağlayacak şekilde i numaralı bireyin bir kopyası eşleşme havuzuna eklenir. Tablo 2.3'de A, B ve C bireyleri ve uygunlukları verilmiştir. Tabloda uygunlukla orantılı olarak hesaplanmış seçim olasılıkları ile farklı s değerleri için çizgisel sıralama metodu ile hesaplanmış seçim olasılıkları görülmektedir. Tablodan da görüleceği gibi seçim baskısının maksimum (s = 2) olduğu durumda uygunluğu en küçük olan en kötü bireyin (A) ebeveyn olarak seçilme olasılığı yoktur [2, 4, 5].

Tablo 2.3 Uygunluk orantılı (UO) seçim ile çizgisel sıralama (ÇS) seçimi olasılıkları

	Uygunluk	Sıra	P_{UO}	$P_{ÇS}$ (s=2)	$P_{ÇS}$ (s=1.5)
A	1	1	0.1	0	0.167
B	5	3	0.5	0.67	0.5
C	4	2	0.4	0.33	0.33
Toplam	10	3	1.0	1.0	1.0

2.3.1.5 Üstel sıralama

Çizgisel sıralama seçiminde uygulanabilinecek seçim baskısı sınırlıdır (maksimum s = 2). Daha yüksek seçim baskısı gerektiren durumlarda üstel sıralama seçimi kullanılabilir. Toplumdaki bireyler uygunluklarına göre çizgisel sıralamada olduğu gibi sıralandıktan sonra sırası i olan bireyin seçim olasılığı 2.2 denklemi kullanılarak hesaplanabilir.

$$p(i) = \frac{1 - e^{i}}{c} \qquad (2.2)$$

Formüldeki c normalizasyon faktörüdür ve bireylerin seçim olasılıkları bulunmadan önce hesaplanmış olmalıdır. Bireylerin seçim olasılıklarının toplamı 1.0 değerini vermelidir. Buna göre toplam birey sayısı m olan bir toplum için c faktörü 2.3 eşitliğinden yararlanılarak hesaplanabilir.

$$\sum_{i=1}^{m} \frac{1-e^{i}}{c} = 1.0 \qquad (2.3)$$

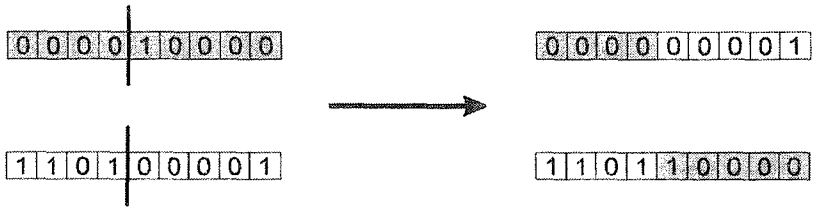
Bu seçim metodunda seçim olasılıklarının hesabından sonraki adımlar çizgisel sıralama seçimindeki ile aynıdır [2, 4, 5].

2.3.2 Çaprazlaşma

Ebeveyn seçimi işlemi sonucunda eşleşme havuzunu doldurmuş olan ebeveynler çaprazlaşma işlemi sonucunda çocuklar oluştururlar. Çaprazlaşma olasılığı (p_c) bize toplumdaki bireylerin kaçta kaçının çaprazlaşmaya uğramasının beklenildiğini gösterir. Örneğin p_c = 0.25 ise bireylerin %25'inin çaprazlaşmaya uğrayacağı beklenilir. P_c çaprazlaşma olasılığı [0-1] aralığında değer alabilir. Çaprazlaşma işlemi çaprazlaşmaya uğrayacak ebeveyn seçimi ve onların eşleştirilmesi olmak üzere iki adım ile başlar. Çaprazlaşmaya uğrayacak ebeveyn seçiminde öncelikle eşleşme havuzundaki her birey için [0-1] aralığında rasgele bir sayı üretilir. Her birey için rasgele üretilmiş olan sayı çaprazlaşma olasılığından (p_c) küçük ise birey çaprazlaşma için seçilir. Çaprazlaşmaya uğramayacak olan bireyler ise kendileri ile aynı özelliklere sahip çocuklar oluşturur. Daha sonra eşleştirme adımına geçilir. Çaprazlaşmaya uğramak için seçilmiş bireylerin sayısına bakılır. Eğer sayı çift ise hiçbir işleme gerek kalmadan eşleştirmeye başlanabilir. Sayının tek olduğu durumlarda ise 0 veya 1 sayılarından birisi rasgele seçilir. 0 seçilmiş olması durumunda ebeveyn seçimi aşamasında doldurulmuş olan eşleşme havuzundan rasgele bir birey çaprazlaşma işlemi için seçilir. 1 seçilmiş olması durumunda ise çaprazlaşmaya uğrayacak olan bireylerden eşleştirme adımına geçebilmiş olan bir birey rasgele seçilip çıkarılır. Bu birey sanki eşleştirme işlemi için hiç seçilmemiş gibi kendisi ile aynı özelliklere sahip bir çocuk oluşturur. Eşleştirme işleminin sonunda çaprazlaşmaya uğrayacak çift sayıda birey kalmıştır. Bu bireyler rasgele ikişer ikişer seçilerek ebeveynler oluşturulur [3]. Eşleştirme işlemi sonucunda oluşan ebeveynler çaprazlaşmanın sonraki adımlarında kullanılacaktır. Her ebeveyn iki çocuk oluşturacaktır. Çaprazlaşmanın sonraki adımları metottan metoda farklılık göstermektedir. Her metot için gerçekleştirilen aşamalar anlatılmıştır.

2.3.2.1 Tek noktadan çaprazlaşma

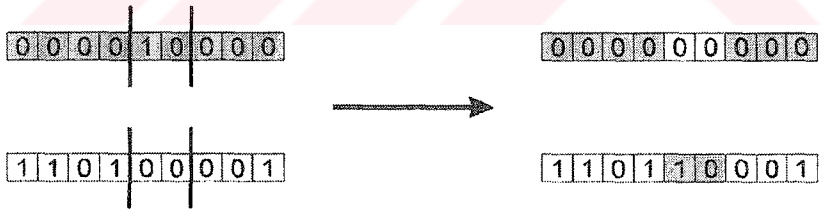
Ebeveyn kromozomları üzerinde rasgele bir nokta seçilir. Oluşan çocukların kromozomları ebeveyn kromozomlarının rasgele seçilmiş noktadan sonraki genlerinin yer değiştirilmiş şekli ile aynıdır. Şekil 2.3'te solda bir ebeveyn yani çaprazlaşma için seçilmiş iki kromozom ve sağda bu kromozomların tek noktadan çaprazlaşması sonucu oluşan iki çocuk görülmektedir [2-4].



Şekil 2.3 Tek noktadan çaprazlaşma

2.3.2.2 Çok noktadan çaprazlaşma

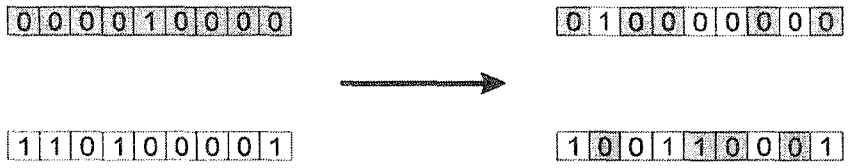
Çok noktadan çaprazlaşma tek noktadan çaprazlaşma ile aynı mantıkla işler. Tek fark çok noktadan çaprazlaşmada kromozom üzerinde önceden belirlenmiş sayıda nokta(lar) rasgele bir biçimde seçilir. Daha sonra ebeveyn kromozomlarının bu noktalar arasında kalan genlerinin yer değiştirmiş şekline sahip kromozomları taşıyan çocuklar oluşturulur. Şekil 2.4 rasgele seçilmiş iki noktadan çaprazlaştırılan ebeveyn (solda) ile çaprazlaşma sonucunda oluşan çocukları (sağda) göstermektedir [2-4].



Şekil 2.4 İki noktadan çaprazlaşma

2.3.2.3 Homojen çaprazlaşma

Homojen (uniform) çaprazlaşmada çocukların kromozomları başlangıçta ebeveyn kromozomları ile aynıdır. Kromozomlardan birinin üzerindeki her gen için rasgele bir sayı üretilir. Üretilen rasgele sayı genin çaprazlaşma olasılığından küçük ise bu noktada kromozomlar üzerindeki genler yer değiştirir. Şekil 2.5'de ebeveyn (solda) ile homojen çaprazlaşma sonucunda oluşan iki çocuk (sağda) görülmektedir [2-4].

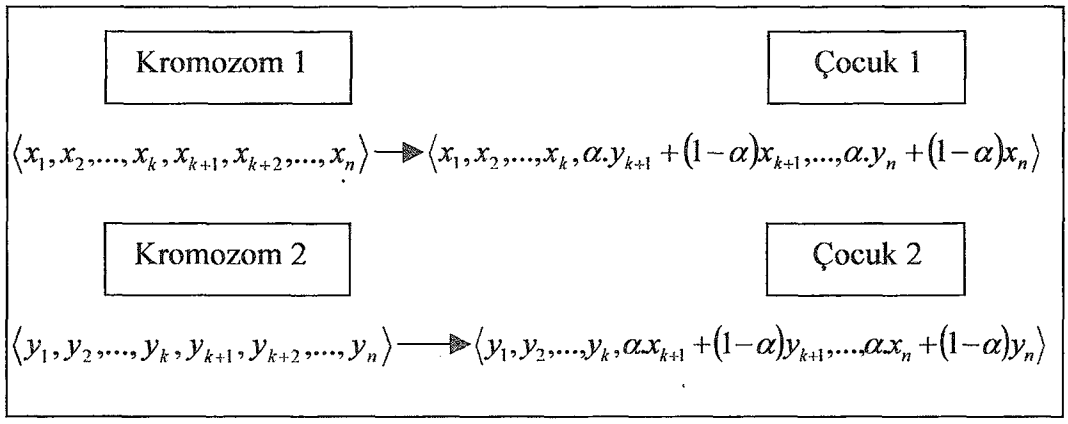


Şekil 2.5 Homojen çaprazlaşma

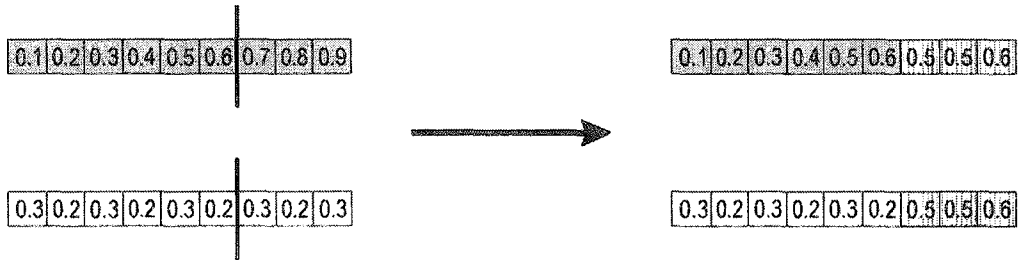
2.3.2.4 Basit aritmetik çaprazlaşma

Aritmetik çaprazlaşmalar kayan noktalı gen yapısına sahip kromozomlar üzerinde kullanılmaktadır. Yeni gen değerleri oluşabilir. Ebeveynde bir bireyin sırası i olan gen değerinin x_i ile, diğerinin aynı sıradaki gen değerinin y_i ile gösterildiği farzedildiğinde çocukların aynı sıradaki geni bireylerin genlerinin değerleri arasında bir değere sahip olur. Çocuklardan birinin geninin taşıdığı değer $z_i = \alpha.x_i + (1-\alpha).y_i$ formülü ile, diğerinin geninin taşıdığı değer ise $z_i = \alpha.y_i + (1-\alpha).x_i$ formülü ile hesaplanabilir. Burada α , [0-1] aralığında bir değer alabilir ve aritmetik sabiti olarak adlandırılır. α için genellikle kullanılan değer 0.5'tir.

Basit aritmetik çaprazlaşmada ebeveyn kromozomları üzerinde rasgele bir nokta seçilir. Seçilen noktadan sonraki genlere aritmetik çaprazlaşma uygulanarak çocukların kromozomları oluşturulur. Şekil 2.6'da uzunluğu n olan kromozomun k noktası rasgele seçilmiştir. Şeklin solunda ebeveyn kromozomlar, sağında ise bu kromozomların k noktasından basit aritmetik çaprazlaşmaya uğraması ile oluşmuş çocuklar görülmektedir. Şekil 2.7'de ise k = 6 ve $\alpha = 0.5$ için basit aritmetik çaprazlaşma işlemi sayısal bir örnek üzerinde gösterilmiştir [2, 4].



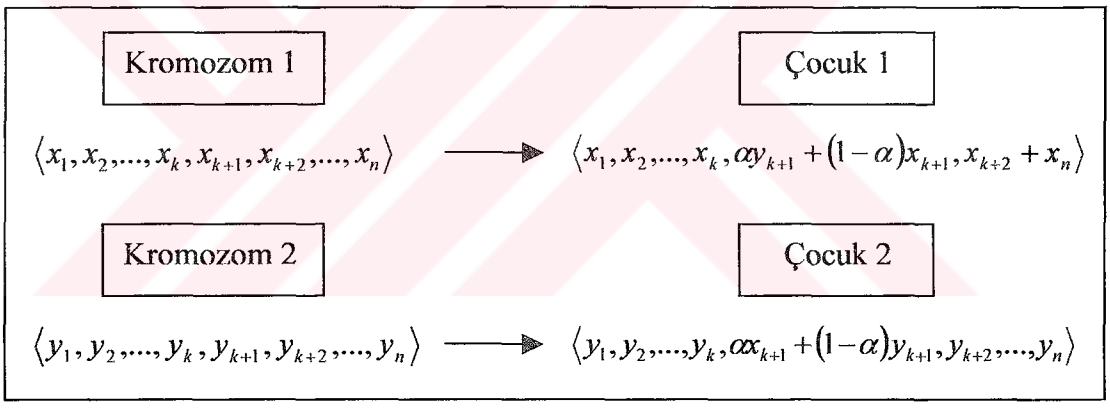
Şekil 2.6 Basit aritmetik çaprazlaşmanın genotip üzerinde formüllerle gösterimi



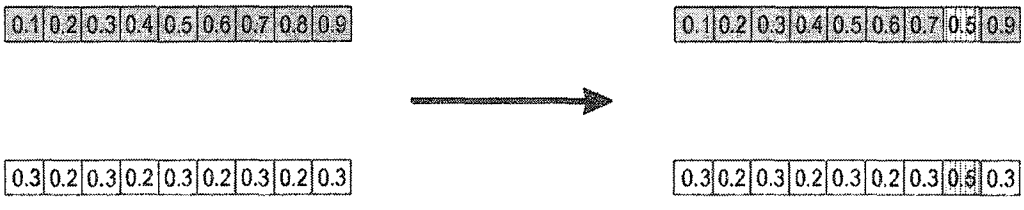
Şekil 2.7 Basit aritmetik çaprazlaşmanın sayısal örnekle gösterimi

2.3.2.5 Tek aritmetik çaprazlaşma

Tek aritmetik çaprazlaşmada ebeveyn kromozomlardan birinin üzerindeki genlerden birisi rasgele seçilir ve diğer kromozomdaki aynı sırada bulunan gen ile aritmetik çaprazlaşma işlemine uğratılıp çocuklar oluşturulur. Aritmetik çaprazlaşma işlemi tek bir gen çifti üzerinde gerçekleşmiş olur. Şekil 2.8 ebeveyn kromozomlar tek aritmetik çaprazlaşmaya uğradıktan sonra oluşan çocukları göstermektedir. Şekil 2.9'da ise k = 8 ve $\alpha = 0.5$ değerleri için tek aritmetik çaprazlaşma işlemi sayısal bir örnek üzerinde gösterilmiştir [2, 4].



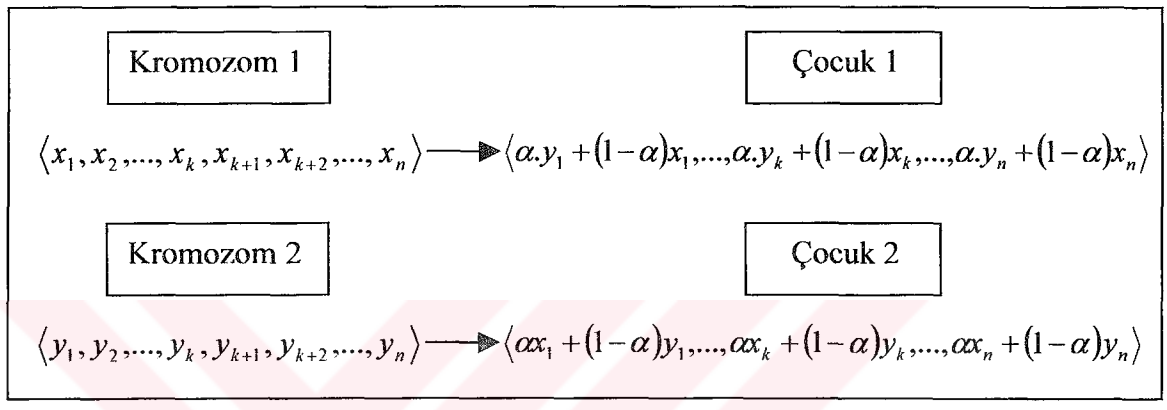
Şekil 2.8 Tek aritmetik çaprazlaşmanın genotip üzerinde formüllerle gösterimi



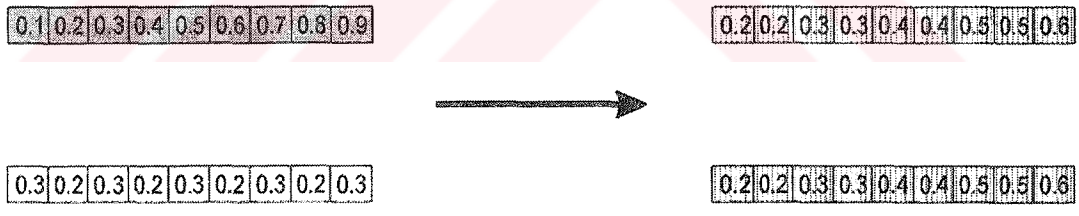
Şekil 2.9 Tek aritmetik çaprazlaşmanın sayısal örnekle gösterimi

2.3.2.6 Tüm aritmetik çaprazlaşma

Aritmetik çaprazlaşma metotları içinde en sık kullanılan metot tüm aritmetik çaprazlaşmadır. Bu çaprazlaşma metodunda aritmetik çaprazlaşma işlemi ebeveyn kromozomlar üzerindeki tüm genlere uygulanarak çocuklar oluşturulmaktadır. Şekil 2.10 ebeveyn kromozomların yapılarını ve bu kromozomlar tüm aritmetik çaprazlaşmaya uğradıktan sonra oluşan çocukların kromozom yapılarını göstermektedir. Şekil 2.11'de $\alpha = 0.5$ değeri için tüm aritmetik çaprazlaşma işlemi sayısal bir örnek üzerinde gösterilmiştir. $\alpha = 0.5$ olması durumunda her iki çocuk aynı gen değerlerinden oluşan özdeş kromozomlara sahip olur [2, 4].



Şekil 2.10 Tüm aritmetik çaprazlaşmanın genotip üzerinde formüllerle gösterimi



Şekil 2.11 Tüm aritmetik çaprazlaşmanın sayısal örnekle gösterimi

2.3.3 Mutasyon

Çaprazlaşmadan sonra ikinci varyasyon operatörü mutasyondur. Mutasyon operatörleri sadece bir bireyi kullanır ve bireyden bir çocuk oluşmasını sağlar. Çaprazlaşma işlemindeki p_c çaprazlaşma olasılığı gibi mutasyon işleminde de p_m mutasyon olasılığından bahsedilir. P_m olasılığı [0-1] aralığında bir değer alabilir. Mutasyon olasılığı toplumun gen havuzundaki genlerin kaçta kaçının mutasyona

uğramasının beklenildiğini gösterir. Kromozom uzunlukları m, ve toplam birey sayısı ise n olan bir toplum için p_m.m.n adet genin mutasyona uğraması beklenilir. Çaprazlaşma operatörlerinde olduğu gibi mutasyon operatörlerinde de bir rasgelelik mevcuttur. Mutasyon operatörleri genotip üzerine etki ederler.

2.3.3.1 İkili mutasyon

İkili mutasyonda bireyin kromozomu üzerindeki her gen için [0-1] aralığında rasgele bir sayı üretilir. Rasgele üretilmiş olan sayının p_m olasılığından küçük olması durumunda genin değeri 0 ise 1'e, aksi taktirde değer 1 ise 0'a dönüştürülür. Şekil 2.12 solda mutasyona uğrayacak birey görülmektedir. Rasgele sayı üretimi sonucunda mutasyona uğrayacak genleri koyu renk ile gösterilmiştir. Şeklin sağında ise genler mutasyona uğradıktan sonra oluşan çocuğun kromozomu görülmektedir [2-4].

Şekil 2.12 İkili mutasyon

2.3.3.2 Rasgele yeniden atama mutasyonu

Tamsayı tipinden veri taşıyan genler üzerinde kullanılan bir mutasyondur. Bireyin kromozomu üzerindeki her gen için [0-1] aralığında rasgele bir sayı üretilir. Rasgele üretilen sayı p_m olasılığından küçük ise gen mutasyona uğratılır. Genin mutasyona uğratılması genin değer alabileceği tamsayı aralığından rasgele bir tamsayı seçilerek gene değer olarak atanması anlamına gelmektedir [2, 4].

2.3.3.3 Homojen mutasyon

Homojen mutasyon tamsayı tipten genler için tanımlanmış rasgele yeniden atama mutasyonuna benzer şekilde çalışır. Ancak bu mutasyon kayan noktalı değerler taşıyan genlere uygulanabilmektedir. Kromozom üzerindeki her gen için [0-1] aralığında rasgele bir sayı üretilerek p_m değeri ile karşılaştırılır. Rasgele sayının mutasyon olasılığından küçük olması durumunda gen mutasyona uğratılır. Genin değer alabileceği aralıkta rasgele bir sayı üretilir ve gene değer olarak atanır [2, 4].

2.3.3.4 Gauss mutasyonu

Gauss mutasyonu kayan noktalı sayı değeri taşıyan genlere uygulanabilen diğer bir mutasyondur. Bu mutasyonda standart sapması istenilen değer olan, sıfır ortalamalı bir Gauss dağılımı kullanılır. Kromozomdaki her gen için Gauss dağılımından rasgele bir değer seçilerek genin değerine eklenir. Daha sonra genin değerinin, alabileceği değer aralığını aşıp aşmadığı kontrol edilir. Değer aralığının aşılması durumunda değer alt sınırdan daha küçük ise alt sınıra, üst sınırdan daha büyük ise üst sınıra taşınarak kırpılmış olur [2, 4].

2.3.3.5 Takas mutasyonu

Takas mutasyonu ve sonraki mutasyonlar permütasyon gösterimlerinde kullanılmaktadır. Bu tip gösterimlerde genler bağımsız olarak düşünülemez. Geçerli mutasyonların oluşması farklı gen değerlerinin genotip içinde hareket etmesi ile sağlanır. Bu durumda p_m mutasyon olasılığı, genin mutasyona uğrama olasılığı yerine gen katarının mutasyona uğrama olasılığı olarak yorumlanmalıdır.

Takas mutasyonunda kromozom için [0-1] aralığında rasgele bir sayı üretilir. Rasgele üretilmiş bu sayının p_m olasılığından küçük olması durumunda kromozom mutasyona uğratılır. Kromozomun mutasyona uğratılması kromozom üzerinde rasgele iki farklı genin seçilmesi ile başlar. Daha sonra rasgele seçilmiş bu iki genin birbirleri ile yer değiştirmesi sağlanır. Takas mutasyonunun nasıl gerçeklendiği Şekil 2.13'te sayısal bir örnek üzerinde gösterilmiştir [2, 4].

Şekil 2.13 Takas mutasyonu

2.3.3.6 Araya ekleme mutasyonu

Araya ekleme mutasyonunda takas mutasyonunda olduğu gibi kromozomun mutasyona uğrayıp uğramayacağının belirlenmesi için [0-1] aralığında rasgele bir sayı üretilir. Bu sayının p_m olasılığından küçük olması durumunda kromozom mutasyona uğratılır. Kromozom üzerinde iki gen rasgele seçilir. Seçilmiş olan ikinci gen birinci genin yanına taşınır. Mutasyonun nasıl gerçekleştiği sayısal bir örnek

üzerinde Şekil 2.14'te gösterilmiştir. Şekilde ikinci sırada seçilmiş değeri beş olan gen ilk sırada seçilen değeri iki olan genin yanına taşınmıştır [2, 4].

Şekil 2.14 Araya ekleme mutasyonu

2.3.3.7 Karıştırma mutasyonu

Karıştırma mutasyonunda [0-1] aralığında rasgele üretilmiş olan sayı kromozomun mutasyona uğrayıp uğramayacağını belirler. Rasgele üretilmiş sayının p_m değerinden küçük olması durumunda kromozom mutasyona uğratılır. Mutasyona uğrayacak kromozomlarda kromozom üzerinde rasgele iki gen seçilir. Kromozomun bu genler de dahil olmak üzere iki gen arasında kalan bölümünün sırası rasgele değiştirilir. Şekil 2.15'te ikinci sıradaki ve beşinci sıradaki genler rasgele olarak seçilmiştir. Sağda kromozomun mutasyona uğramasından sonra oluşan kromozom ve genlerin rasgele karıştırılarak dizilmiş şekli görülmektedir [2, 4].

Şekil 2.15 Karıştırma mutasyonu

2.3.3.8 Tersine çevirme mutasyonu

Tersine çevirme mutasyonunda önceki permütasyon mutasyonlarında olduğu gibi [0-1] aralığında rasgele bir sayı üretilir. Rasgele üretilen sayının p_m olasılığından küçük olması durumunda kromozom mutasyona uğratılır. Mutasyon işleminde öncelikle kromozom üzerinde iki gen rasgele seçilir. Daha sonra seçilmiş olan iki gen ve bu genler arasında kalan genlerin sırası ters çevrilir. Şekil 2.16'da ikinci ve beşinci genler rasgele seçilmiştir. Kromozomun mutasyona uğramadan önceki şekli (solda) ve tersine çevirme mutasyonuna uğradıktan sonra oluşan yeni kromozom ve genlerin ters bir biçimde dizilişi (sağda) görülmektedir [2, 4].

Şekil 2.16 Tersine çevirme mutasyonu

2.3.4 Sonlanma Kriteri

Evrimsel algoritmanın ne zaman sonlanacağı sonlanma kriteri ile belirlenir. Çağırılan metot geriye True değerini gönderdiği sürece evrimsel işlemler sürer. PyLEA paketinde gerçeklenmiş olan sonlanma kriteri metotları aşağıda açıklanmıştır. Evrimsel algoritma çalıştığı sürece oluşan çözüm adayları arasında en iyi olanı paketin istatistik bölümü tarafından saklanır. Sonlanma kriteri olarak optimum uygunluk metodu seçilmedikçe bulunmuş olan en iyi çözüm adayının optimum olacağı garanti edilemez.

2.3.4.1 Nesil sayısı

Sonlanma kriteri metotlarından birisi nesil sayısı metodudur. Algoritma belirlenmiş nesil sayısına ulaşılıncaya kadar çalışır. Şekil 2.1'de gösterilmiş olan beş adımdan oluşan döngü belirlenmiş nesil sayısı kez çalışır [2, 4].

2.3.4.2 Optimum uygunluk

Problemin optimum çözümünün bilindiği durumlarda bu metot kullanılabilir. Algoritma, uygunluğu optimum uygunluk derecesine eşit olan bir birey, bir çözüm adayı bulununcaya dek devam eder. Ancak evrimsel algoritmalarda optimum çözümün bulunacağı kesin değildir. Algoritma yerel bir optimumda da takılabilir. Yani sonlanma kriteri olarak optimum uygunluk metodunun kullanılması durumunda algoritma hiç sonlanmayabilir [2, 4].

2.3.4.3 Yakınsama

Yakınsama kavramı temelde aynılığa doğru ilerleyiş olarak düşünülebilir. Bir toplumda bireylerin belirli bir yüzdesinin (örneğin %95'inin) kromozomları üzerinde aynı sıradaki genlerinin (örneğin ilk, birinci sıradaki, genlerinin) aynı olması, yani aynı değeri taşıması durumunda o gen için gen yakınsaması söz konusudur. Topumdaki bireylerin kromozomları üzerinde tüm genleri yakınsadıysa toplum yakınsamıştır denilir. Sonlanma kriteri olarak yakınsama metodu kullanılması

durumunda bir yakınsama yüzdesi seçilir. Toplumdaki yakınsama yüzdesi oranında bireyin tüm genleri yakınsadıysa toplum yakınsamış demektir ve algoritma sonlanır [2, 4].

2.3.5 Hayatta Kalma

Darwin'in evrim teorisindeki doğal seçim, aynı kaynaklar için rekabet eden bireyler arasından en uygun bireyin hayatta kalacağını diğer bir deyişle çevresel koşullara en iyi adaptasyon sağlayan bireyin hayatta kalacağını belirtmekteydi. Evrimsel algoritmalarda çocuklar oluştuktan sonra yapılan hayatta kalış seçimi ise hangi bireylerin yani hangi çözüm adaylarının sonraki nesile geçirileceğini belirler. Hayatta kalma seçimi sonucunda oluşan yeni nesilde toplumdaki birey sayısı sabit kalmaktadır.

2.3.5.1 Nesilsel

Nesilsel seçimlerde toplumdaki birey sayısı kadar oluşturulmuş olan çocuk toplumdaki bireylerin yerini alır. Burada bireylerin uygunlukları göz önüne alınmaz. Her birey sadece bir nesil içinde varolabilir. Sonraki nesile sadece o nesilde oluşturulmuş olan çocuklar aktarılır [2, 4].

2.3.5.2 En iyi bireyin seçimi

En iyi bireyin seçiminde toplumdaki bireylerin ve çocukların uygunluk dereceleri göz önüne alınır. Toplumdaki bireyler arasında uygunluk derecesi en kötü olan birey ve çocuklar arasında uygunluk derecesi en iyi olan çocuk bulunur. Daha sonra bulunmuş olan toplumdaki en kötü birey çıkarılıp onun yerine bulunmuş, en iyi uygunluğa sahip çocuk bireylerin arasına eklenir ve bireyler sonraki nesile aktarılır. Bu metotla en iyi birey hep toplumda tutulmuş olur [2, 4].

2.3.5.3 Turnuva seçimi

Hayatta kalma seçimlerinden turnuva seçimi metodu ebeveyn seçimi metotlarından olan turnuva seçimi metodundan biraz farklılık gösterir. Hayatta kalma seçimi metodu olan turnuva seçiminde toplumdaki birey ve çocuklar bir araya getirilir. Belirlenmiş k kadar yani seçilecek birey sayısı kadar birey toplam çözüm adayları arasından rasgele seçilir. Daha sonra seçilmiş olan çözüm adayları arasından uygunluğu en iyi olan çözüm adayı bulunur ve bir sonraki nesile aktarılır.

Toplumdaki birey sayısının sabit kalması gerektiğinden bu işlem toplumdaki birey sayısı kez tekrarlanır. Hatırlanacağı gibi ebeveyn seçimi metodu olan turnuva seçiminde aynı bireyin, yani aynı çözüm adayının, birden fazla kez seçilmesi ihtimali vardı. Hayatta kalma seçiminde ise seçilen her çözüm adayı (birey ya da çocuk) toplam çözüm adayları arasından çıkarılır. Böylece aynı çözüm adayının birden fazla kez seçilebilmesi engellenmiş olur [4].

Hayatta kalma seçim metotları arasında bulunan turnuva seçimi, rulet çarkı seçimi, çizgisel ve üstel sıralama seçimlerinin ebeveyn seçim metotlarından farkı yapılan seçimlerin birey ve çocukların toplamı arasından yapılıyor olması ve seçilmiş birey ya da çocukların toplamdan çıkarılarak tekrar seçilme ihtimallerinin ortadan kaldırılmasıdır.

2.3.5.4 Rulet çarkı seçimi

Rulet çarkı seçiminde öncelikle oluşturulan toplam çözüm adayları için seçilme olasılıkları hesaplanır. Seçilme olasılıklarının hesaplanmasında kullanılan formül aynıdır. Burada toplam uygunluk her bireyin uygunluğu ile her çocuğun uygunluğunun toplamı olacaktır. Sonra hesaplanan seçilme olasılıkları kullanılarak ebeveyn seçimi metodu olan rulet çarkı seçiminde olduğu gibi her birey ve çocuğun kümülatif olasılığı hesaplanır. Rulet çarkı toplumdaki birey sayısı kadar çevirilir. Her çevrimde [0-1] aralığında rasgele bir sayı üretilir ve ebeveyn seçimi metodundaki gibi, kümülatif olasılıklar göz önünde bulundurularak, bir çözüm adayı seçilir (B.k. 2.2.1.2. Rulet çarkı seçimi). Seçilen çözüm adayı toplam çözüm adayları arasından çıkarılıp bir sonraki nesile aktarılır. Kalan çözüm adayları toplamı için aynı şekilde yeniden seçilme olasılıkları ve kümülatif olasılıklar hesaplanır ve çark yeniden çevirilir [4].

2.3.5.5 Çizgisel sıralama seçimi

Çizgisel sıralama seçiminde bireylerin ve çocukların toplamı olan tüm çözüm adayları uygunluk derecelerine göre sıralanırlar. Her çözüm adayı için seçilme olasılığı ve kümülatif olasılık ebeveyn seçimi metodunda olduğu gibi hesaplanır. Sonra ebeveyn seçimi metodundaki gibi [0-1] aralığında rasgele bir sayı üretilir, kümülatif olasılıklardan da faydalanılarak bir çözüm adayı seçilmiş olur (B.k. 2.2.1.4. Çizgisel sıralama). Seçilen çözüm adayı toplam çözüm adayları listesinden çıkarılır ve sonraki nesile aktarılır. Kalan çözüm adayları için seçilme

olasılıkları ve kümülatif olasılıklar tekrar hesaplanır, rasgele bir sayı üretilir, başka bir çözüm adayı daha bir sonraki nesile aktarılmak için seçilir. Seçilen bireylerin tekrar seçilmesi mümkün olmaz. Toplumdaki birey sayısı sabit kalacağından işlemler toplumdaki birey sayısı kadar tekrarlanır [4].

2.3.5.6 Üstel sıralama seçimi

Çizgisel sıralama seçiminde olduğu gibi bireylerin ve çocukların toplamı olan tüm çözüm adayları uygunluk derecelerine göre sıralanırlar. Ebeveyn seçimi metodunda olduğu gibi *c* normalizasyon faktörü hesaplanır. Normalizasyon faktörünün formülde yerine konulmasıyla her bir çözüm adayı için seçilme olasılığı hesaplanır. Seçilme olasılıkları kullanılarak her bir çözüm adayının kümülatif olasılığı bulunur. Daha sonra rasgele üretilen sayı ve kümülatif olasılıklar kullanılarak tüm çözüm adayları arasından bir çözüm adayı seçilir (B.k. 2.2.1.5. Üstel sıralama). Ebeveyn seçimi metodundan farklı olarak seçilen çözüm adayı tüm adaylar toplamından çıkarılır ve bir sonraki nesile aktarılır. Böylece seçilmiş olan aday tekrar seçilemeyecektir. Geride kalan çözüm adayları için sırasıyla normalizasyon faktörü, seçilme olasılıkları, kümülatif olasılıklar tekrar hesaplanır. Rasgele bir sayı üretilerek, aynı şekilde başka bir çözüm adayı daha seçilip adaylar toplamından çıkarılır ve bir sonraki nesile aktarılır. Seçim işlemi yeni nesildeki birey sayısı önceki neslin birey sayısına eşit oluncaya dek sürer [4].

2.4 Gerçeklenen Örnek Problemler

Önceki bölümlerde PyLEA paketinde gerçeklenmiş olan operatörler ve metotları anlatılmıştır. Bu bölüm evrimsel algoritmalarda bazı metotların ya da parametrelerin testi v.b. amaçlarla kullanılabilen problemleri anlatmaktadır. PyLEA paketinde tanımlanmış olan bu problemler optimizasyon (en iyileme) problemleridir ve evrimsel hesaplamalarda yaygın olarak kullanılırlar.

2.4.1 Bir maksimizasyonu

Bir maksimizasyonu (One Max) probleminde belirlenmiş bir uzunlukta 0 ya da 1'lerden oluşmuş bir dizi vardır. Problemin amacı dizide varolan toplam 1 sayısının maksimum olmasını sağlamaktır. Dizi uzunluğu n olan bir maksimizasyonu problemi için bulunabilecek en iyi çözüm dizinin tüm elemanlarının 1 olması durumundaki

toplam, yani n değeri olacaktır. Bir maksimizasyonu probleminde uygunluk derecesi $f(x) = \sum_{i=1}^{n} x_i$ formülü ile hesaplanır. Burada f uygunluk fonksiyonu olarak adlandırılır ve x_i dizinin i. sıradaki elemanının değeridir. Problemin evrimsel algoritmalar kullanılarak çözülmeye çalışılması durumunda her çözüm adayı 0 ya da 1 değerine sahip n adet genden oluşacaktır [2, 4].

2.4.2 Alt Kümeler Toplamı (Çanta Problemleri)

Bir maksimizasyonu probleminden farklı olarak alt kümeler toplamı ya da çanta (Knapsack) problemleri kısıt koşullarına sahiptir. Evrimsel algoritma devam ederken oluşan yeni çözüm adaylarından kısıt koşullarını sağlayanlar geçerli, koşulları sağlamayanlar ise geçersizdir. Amaç kısıt koşullarını sağlayan optimum çözümü bulmak olduğundan toplumda varolan geçersiz çözüm adayları ile ne yapılacağına karar verilmelidir. Bu amaçla kullanılan yöntemlerden birisi geçersiz çözümlerin bir ceza fonksiyonu kullanarak uygunluk derecelerinin düşürülmesidir. Kullanılan diğer bir yöntem geçersiz çözümlerin geçerli çözümler haline getirilmesi, diğer bir deyişle onarılmasıdır.

Alt kümeler toplamı problemlerine ait çanta taşıma kapasitesi ya da kapasiteleri, nesnelerin karları ve ağırlıkları gibi parametreler EA arayüzünü kullanarak PyLEA paket yapısına bir dat dosyası içinde yüklenebilir. Bu durumda dosyadaki parametrelerin dosyadan hatasız okunabilmeleri için parametreler dosyaya uygun formatta kaydedilmiş olmalıdırlar [8]. Alt kümeler toplamı problemleri ve geçersiz çözümlerin ne şekilde ele alındığı sonraki bölümlerde daha detaylı bir biçimde anlatılmaktadır.

2.4.2.1 0/1 Çanta

0/1 çanta probleminde bir çanta ve belirlenmiş sayıda nesne bulunmaktadır. Her nesnenin bir ağırlığı ve çantada bulunduğu taktirde sağlayacağı kazancı vardır. Ayrıca çantanın taşıma kapasitesi de belirlidir. Problemin amacı çantanın taşıma kapasitesini aşmayacak şekilde çantaya toplamda en fazla kazancı sağlayacak nesneleri koymaktır. Problemin evrimsel algoritmalarla çözümünde çözüm adayları nesne sayısı kadar gene sahip olan kromozomlardan oluşacaktır. Kromozom üzerindeki her gen belirli bir nesneyi temsil eder. Genler ikili tiptendir yani 0 ya da 1

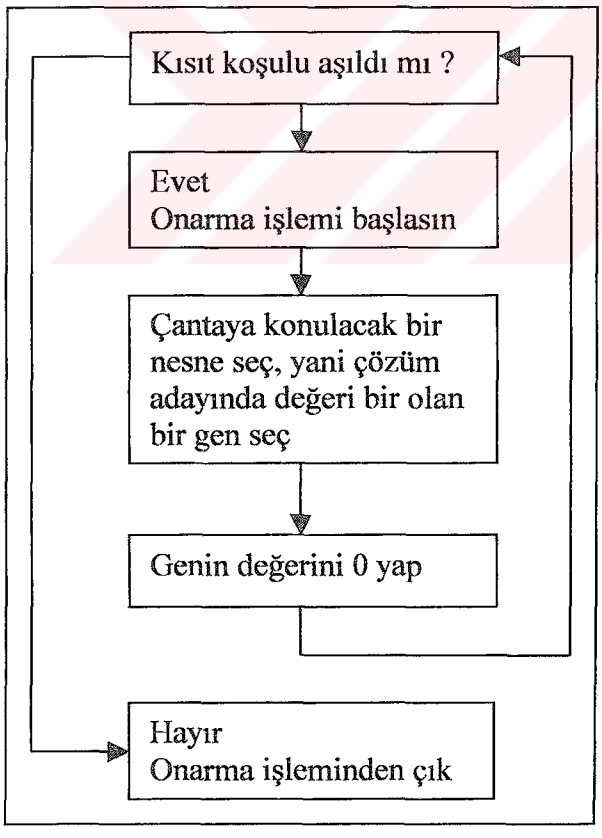
değerlerini alabilirler [2-4]. Bir nesneye karşılık gelen genin değerinin 1 olması o nesnenin çantada bulunması anlamına gelmektedir. Nesne sayısı n olan bir 0/1 çanta problemi için her kromozom $\mathbf{k} = [g_1,g_2,g_3,...,g_n]$ dizi yapısında olur. Problemde amaç $P(k) = \sum_{i=1}^{n} g_i.P(i)$ toplamının en büyük olmasıdır. Eşitlikte $P(k)$ çantadaki tüm nesnelerin toplam kazancı, g_i i. nesnenin çantada varolup olmadığını belirten i. sıradaki genin değeri, $P(i)$ ise i. nesnenin çantada bulunması halinde sağlayacağı kardır. Problemde her kromozom için uyulması gereken kısıt koşulu ise $\sum_{i=1}^{n} g_i.W_i \le C$ şeklindedir. W_i, i. nesnenin ağırlığı, C ise çantanın taşıyabileceği maksimum ağırlık yani çantanın taşıma kapasitesidir. Çantada bulunacak nesnelerin toplam ağırlığı çantanın taşıma kapasitesinden fazla olmamalıdır. Ancak rasgele oluşturulan ilk toplumda, veya bazı operatörlerden sonra kısıt koşulunu sağlamayan çözüm adayları oluşması mümkündür. Bu durumda çözüm adaylarına uygulanmak üzere yukarıda bahsedilmiş ceza fonksiyonu uygulaması ve çözümlerin onarılması yöntemleri paket yapısında gerçeklenmiştir. Yöntemlerin nasıl gerçeklendiği aşağıda detaylı biçimde açıklanmıştır.

Ceza fonksiyonu yönteminde kısıt koşulunu sağlamayan çözüm adayının uygunluk derecesi azaltılmaktadır. Problemin amacı en büyük $P(k) = \sum_{i=1}^{n} g_i.P(i)$ toplamını veren çözüm adayını bulmaktı. O halde kısıt koşulunu sağlayan her çözüm adayının uygunluk derecesi $f(k) = \sum_{i=1}^{n} g_i.P(i)$ formülü ile hesaplanabilir. Formüldeki $f(k)$ uygunluk fonksiyonu çözüm adayına göre çantada bulunan nesnelerin karlarının toplamını hesaplamaktadır. Çözüm adayının kısıt koşulunu sağlamadığı durumlarda ise uygunluk fonksiyonu tarafından hesaplanan uygunluk derecesi bir ceza miktarınca azaltılmalıdır. PyLEA paket yapısında ceza miktarı $Pen(k) = \log_2\left(1 + \rho.\left(\sum_{i=1}^{n} g_i.W_i - C\right)\right)$ formülüne göre hesaplanmıştır. Formüldeki $\rho = \max_{i=1...n}\{P_i/W_i\}$ olmaktadır. Diğer bir deyişle uygunluk fonksiyonu, $Pen(k)$ 2.4 denklemi ile gösterilmiş ceza fonksiyonu olmak üzere 2.5 denkleminde belirtildiği biçimde hesaplanabilir [3].

$$Pen(k) = \begin{cases} 0 & , \sum_{i=1}^{n} g_i.W_i \le C \\ \log_2\left(1 + \rho.\left(\sum_{i=1}^{n} g_i.W_i - C\right)\right) & , \sum_{i=1}^{n} g_i.W_i > C \end{cases} \quad (2.4)$$

$$f(k) = \sum_{i=1}^{n} g_i.\mathrm{P}(i) - Pen(k) \quad (2.5)$$

Çözüm adaylarının onarılması yönteminde uygunluk derecesi hesaplanmadan önce çözüm adayının kısıt koşulunu aşıp aşmadığı kontrol edilir. Çözüm adayı kısıt koşulunu aşmış ise çözüm adayının çantaya konulmak üzere temsil ettiği nesnelere karşılık gelen genlerden biri seçilip seçilen genin değerinin 0 yapılması ile çözüm adayı kısıt koşulunu sağlar hale getirilmeye çalışılır. Kısıt koşulu sağlanmadığı sürece aynı işlem tekrarlanır. Kısacası çantanın taşıma kapasitesinin aşılmaması için nesnelerden birinin ya da birkaçının çantaya konulması engellenir. Bu yöntemde eski çözüm adayı değiştirilir, yeni bir çözüm adayı elde edilmiş olur. Ancak sonuçta tüm çözüm adayları kısıt koşulunu sağlamış olurlar. Onarma işleminin nasıl gerçekleştiği Şekil 2.17'de görülmektedir.



Şekil 2.17 Kısıt koşulunu sağlamayan geçersiz çözümlerin onarım akış diyagramı

Onarım işlemi başladıktan sonraki adımda nesnenin seçilmesi iki yöntemle yapılabilir. İlk yöntemde kromozom üzerinde değeri bir olan genler arasından birisi rasgele seçilir. PyLEA paket yapısında gerçeklenmiş olan ikinci yöntemde ise çantaya konulacak nesneler karlarının ağırlıklarına olan oranlarına göre sıralanırlar ve kar-ağırlık oranı en küçük olan nesne seçilir [3].

2.4.2.2 Çok sayıda çanta

Çok sayıda çanta probleminde 0/1 çanta probleminden farklı olarak tek çanta yerine belirlenmiş sayıda çanta bulunmaktadır. Çanta sayısı m, ve nesne sayısı n olan bir çok sayıda çanta problemi için Şekil 2.18'de koyu renk yazı ile gösterilmiş olan veriler bilinmelidir.

Çanta Sayısı	:	1	2	...	m	
Kapasiteler	:	c_1	c_2	...	c_m	
Nesne Sayısı	:	1	2	...	n	
Karlar	:	p_1	p_2	...	p_n	
Ağırlıklar	:	w_{1j}	w_{2j}	...	w_{nj}	, j = 1,2,...,m

Şekil 2.18 Çok sayıda çanta problemine ait veriler

Şekilden de görüleceği gibi 0/1 çanta probleminin aksine çok sayıda çanta probleminde çanta sayısı birden fazla olmaktadır ve bu problemde her nesnenin ağırlığı her çanta içinde farklılık gösterebilmektedir. Problemin evrimsel algoritmalarla çözümünde 0/1 çanta probleminde olduğu gibi nesne sayısı kadar (n adet) genden oluşan kromozomlar kullanılır. Her gen bir nesneye karşılık gelir ve i. sıradaki genin değeri 1 ise i. nesne tüm çantalara konulacak, değer 0 ise hiçbir çantaya konulmayacak anlamına gelir. Amaç kısıt koşullarını sağlayan ve $P(k)=\sum_{i=1}^{n} g_i.p_i$ olacak şekilde en büyük toplamı veren çözüm adayının bulunmasıdır. Çok sayıda çanta probleminde kısıt koşulu tek değil, çanta sayısı kadardır. Kısıt koşulları her bir çanta için çantaya konulmuş olan nesnelerin o çanta içindeki ağırlıklarının çantanın taşıma kapasitesini aşmamasıdır, diğer bir deyişle $j=1,2,...,m$ olmak üzere koşul $\sum_{i=1}^{n} w_{ij}.g_i \le c_j$ eşitsizliği ile ifade edilebilir.

Kısıt koşulunu sağlamayan çözüm adayları için ceza fonksiyonu uygulanması yöntemi seçilmiştir. $Pen(k)$ ceza fonksiyonu ve $f(k)$ uygunluk fonksiyonu sırasıyla 2.6 ve 2.7 denklemlerinde gösterilmiştir.

$$Pen(k) = s.maksimum\{p_i\} \tag{2.6}$$

$$f(k) = \left(\sum_{i=1}^{n} p_i.g_i\right) - Pen(k) \tag{2.7}$$

Ceza fonksiyonundaki s, taşıma kapasitesi aşılmış çanta sayısı, $maksimum\{p_i\}$ ise çantaya konulacak nesnelerin karlarının en büyüğüdür. Tüm kısıt koşulları sağlandığı taktirde taşıma kapasitesi aşılmış çanta sayısı, yani s, 0 olacağından uygunluk derecesinden çıkarılacak ceza da 0 olur [6].

2.4.3 De Jong Fonksiyonları

Problemlerde amaç belirli bir değer aralığı içinde De Jong fonksiyonlarının alabileceği optimum değerin bulunmasıdır. Yani problemler optimizasyon problemleridir. PyLEA paket yapısında f1, f2, f3, f4, f5 ve f7 fonksiyonları için evrimsel algoritmalarla optimum çözümün aranmasına imkan tanınmaktadır.

Problemlerde kısıt koşulu mevcuttur, çünkü çözümler belirli bir değer aralığında aranmaktadır. Bu değer aralığının dışındaki çözümler geçersiz olacaktır. Çözümün aranmasında gen değerleri ikili düzende olan yani 0 ya da 1 değerlerini alabilen genlerden oluşan kromozomlar kullanılmıştır.

Sonraki bölümlerde çözüm adaylarının oluşturulması, uygunluk derecelerinin hesaplanması gibi konular detaylı bir biçimde anlatılmıştır.

2.4.3.1 Gray kodlaması

Tamsayı gösterimlerinde fenotip uzayında yan yana olan iki noktanın genotip uzayında ikili gösterimlerde bir bitlik bir farka sahip olması gerekir. Diğer bir deyişle fenotip uzayında parametre değerindeki bir adımlık bir artış genotip uzayında bir bitlik bir değişime karşılık gelmelidir. Sayısal bir örnek üzerinde açıklamak gerekirse 7 sayısının değişerek 6 ya da 8 olma olasılığı eşit olmalıdır. İkilik düzendeki 7 sayısı 0111 şeklinde ifade edilir. Bu sayının altıya dönüşmesi için (0111 → 0110) bir bitlik bir değişim gerek iken sayının sekize dönüşmesi için (0111 → 1000) dört

bitlik bir değişim gerekmektedir. Örneğin mutasyon işlemini ele alalım. Yedi sayısının mutasyona uğrayarak altıya dönüşmesi ihtimali ile sekize dönüşmesi ihtimali aynı değildir. Bu sorunun ortadan kaldırılması amacıyla Gray kodlaması kullanılmaktadır. İkili düzende üç bitlik sayılar için tamsayı karşılığı ve Gray kodu Tablo 2.4'de gösterilmiştir.

Tablo 2.4 Üç bitlik Gray kodlaması

Tamsayı	0	1	2	3	4	5	6	7
Standart	000	001	010	011	100	101	110	111
Gray Kodu	000	001	011	010	110	111	101	100

İkili düzenden Gray koduna geçişte en anlamlı bit daima sabit kalır. Ondan sonra Gray kodundaki her bit, kendisi ile aynı sırada olan bit ile onun hemen solundaki bitin XOR işlemine tabi tutulması ile bulunur. Örneğin tablodaki 2 sayısı için Gray koduna bakılacak olursa en anlamlı bit olan 0 sabit kalmıştır. Ondan sonraki bit, standart gösterimdeki ikinci bit ile hemen solundaki en anlamlı bit olan ilk bitin XOR işlemine tabi tutulması ile (1 XOR 0 = 1) bulunmuştur. Gray kodundaki üçüncü sıradaki son bitin bulunması standart gösterimdeki üçüncü ve ikinci bitlerin XOR işlemine uğratılması sonucu (0 XOR 1 = 1) bir olarak bulunmuştur.

Gray kodundan ikili düzene geçişte en anlamlı bit sabit kalır ve saklanır. Sonraki bitler için bir döngüye girilir. Eğer Gray kodundaki bitin değeri 0 ise saklanmış olan değer, aksi taktirde, Gray kodundaki bitin değeri 1 ise, saklanmış olan değerin tersi (0 ise 1, 1 ise 0) standart gösterimdeki bite atanır ve bu kez Gray kodundaki bitin değeri saklanır. Yine tablodaki iki sayısı için Gray kodundan standart gösterime dönüşümü inceleyelim. En anlamlı bitin değeri olan 0 standart gösterimde de sabit kalmıştır. Ayrıca döngü başlamadan önce en anlamlı bitin değeri, yani 0 saklanır. Daha sonra standart gösterimde ikinci sıradaki bitin değerinin saptanmasına geçilir. Gray kodunda ikinci bitin değeri 1'dir bu durumda saklanmış olan değerin tersi standart gösterimde ikinci sıradaki bite atanmalıdır. Saklanmış olan 0'ın tersi, yani 1 ikinci bite atanır. Son bit için döngü tekrarlanmadan önce Gray kodundaki bitin değeri olan 1 bu kez saklanır. Gray kodundaki son bitin değeri 1'dir. Bu nedenle saklanmış olan bitin tersi, yani 0, standart gösterimdeki son bite atanır. Gray kodundaki son bitin değeri olan 1 de saklanır ancak tüm bitler saptandığından döngü sonlanır.

Tablo 2.4'ten de görülebileceği gibi Gray kodu ile ardışık tamsayıların ikili düzende aralarındaki Hamming uzaklığı bir olan sayılara dönüşmesi sağlanmış olur [2-4].

2.4.3.2 F1 fonksiyonu

F1 fonksiyonu tek bir optimuma sahiptir. Fonksiyon ve kısıt koşulu 2.8 denkleminde gösterilmiştir [7].

$$f1 = \sum_{i=1}^{2} x_i^2 \qquad ,-5.12 < x_i \leq 5.12 \qquad (2.8)$$

Problemin amacı değer aralığı içerisinde en küçük $f1$ değerini veren x_1 , x_2 sayılarını bulmaktır. Fenotip uzayı ondalık sayılardan oluşmakta oysa genotip uzayı ikili düzende çözüm adaylarına sahip olacaktır. Bu durumda ondalık sayılardan ikili düzene geçiş olmalıdır. Fenotip uzayından genotip uzayına geçişte öncelikle çözüm adayı tamsayıya çevrilir. Bu çevrimde virgülden sonra iki rakamlık bir hassasiyet seçilmiştir. Daha sonra her tamsayı ikili düzende bir sayıya karşılık düşülmüştür. (-512,512] aralığında 1024 adet tamsayı bulunur. Bu durumda genotip uzayına dönüşümde -511 tamsayısı ikili düzende 0'a karşılık gelecek şekilde dönüşüm yapılır. 1024 adet tamsayı ikili düzende gösterileceğinden gösterimlerde ikili düzende on bitlik sayılar kullanılmıştır. Fenotip uzayından genotip uzayına dönüşüm Şekil 2.19'da görülmektedir.

Fenotip Uzayı	Tamsayıya Geçiş	Genotip Uzayı	Tamsayı Karşılığı
- 5.11	→ -511 →	0000000000	0
- 5.10	→ -510 →	0000000001	1
.	.	.	.
.	.	.	.
.	.	.	.
5.11	→ 511 →	1111111110	1022
5.12	→ 512 →	1111111111	1023

Şekil 2.19 F1'in tanımlı olduğu aralık için fenotipler ve genotip karşılıkları

Şekilden de anlaşılacağı gibi fenotip uzayından genotip uzayına bu şekilde bir dönüşüm yapılması sayesinde kısıt koşulunun sağlanması problemi ortadan kalkmaktadır. Görüldüğü gibi genotip uzayındaki her çözüm adayı fenotip uzayında kısıt koşulunu sağlayan bir çözüm adayına karşılık gelmektedir, yani optimum $f1$ değerini verecek çözüm adayı fenotip uzayında mutlaka kısıt koşulunu sağlamış olacaktır.

Problemin evrimsel algoritmalar ile çözümünde x_1 ve x_2 sayılarının herbiri ikili düzende on bitlik birer sayı ile ifade edilmelidir. Optimum çözümü verecek her iki sayı da aranmaktadır. O halde birer bitlik değerler taşıyan 20 genden oluşan bir kromozom çözüm adayı olarak kullanılabilir. Kromozomdaki ilk 10 gen x_1 sayısının değerini, son 10 gen ise x_2 sayısının değerini ifade eder (B.k. Şekil 2.20). Çözüm adayının uygunluk derecesi kromozomdan elde edilecek x_1 ve x_2 sayılarının fenotip uzayında temsil ettikleri değerlerin bulunup $f1$ fonksiyonunda yerine yazılması ile hesaplanabilir.

Kromozom : **1100101010**1100000000
x_1 x_2

Şekil 2.20 Kromozomun genotipini oluşturan yapı

2.4.3.3 F2 fonksiyonu

F2 fonksiyonu da F1 fonksiyonu gibi tek bir global optimuma sahiptir. Fonksiyon ve kısıt koşulu 2.9 denkleminde gösterilmiştir [7].

$$f2 = 100.\left(x_1^{\,2} - x_2\right)^2 + \left(1 - x_1\right)^2 \qquad ,-2.048 < x_1, x_2 \le 2.048 \qquad (2.9)$$

Fenotip uzayından genotip uzayına dönüşümde ondalık sayıların tamsayılara çevirilmesinde virgülden sonra üç rakamlık bir hassasiyet seçilmiştir. Bu durumda değer aralığında 4096 tamsayı bulunacağından ikili düzende oniki bitlik sayılar kullanılmıştır. Dönüşümün nasıl gerçeklendiği Şekil 2.21'de gösterilmiştir.

Fenotip Uzayı	Tamsayıya Geçiş	Genotip Uzayı	Tamsayı Karşılığı
- 2.047	→ -2047 →	000000000000	0
- 2.046	→ -2046 →	000000000001	1
.	.	.	.
.	.	.	.
.	.	.	.
2.047	→ 2047 →	111111111110	4094
2.048	→ 2048 →	111111111111	4095

Şekil 2.21 F2'nin tanımlı olduğu aralık için fenotipler ve genotip karşılıkları

Optimum çözümün bulunması için oluşturulan kromozomlar x_1 sayısını temsilen 12 adet ve x_2 sayısını temsilen 12 adet olmak üzere toplam 24 adet birer bitlik değerler taşıyan ikili tipte genlere sahiptir. Çözüm adaylarının uygunluk derecelerinin

hesaplanması kromozom üzerindeki sayıların genotipleri kullanılarak fenotiplerinin bulunması ve bu değerlerin $f2$ fonksiyonunda yerine yazılması ile bulunur.

2.4.3.4 F3 fonksiyonu

Tek bir optimuma sahip olan fonksiyon ve kısıt koşulu 2.10 denkleminde gösterilmiştir [7].

$$f3 = \sum_{i=1}^{5} tamsayı(x_i) \qquad ,-5.12 < x_i \leq 5.12 \qquad (2.10)$$

Fonksiyonun fenotip uzayındaki çözüm adayları virgülden sonra iki rakam hassasiyeti ile tamsayılara dönüştürülüp bu tamsayılar Şekil 2.19'da gösterildiği gibi genotip uzayında ikili düzende sayılara karşılık getirilmiştir. Fonksiyon için optimum çözümün aranmasında kullanılan kromozomlar i = 1,2,3,4,5 olmak üzere her x_i sayısı için 10'ar adet bir bitlik gen taşırlar. Yani bir kromozom toplamda 50 genden oluşur. Uygunluk derecesinin hesaplanması x_i sayılarının kromozomda taşınan genotip değerlerinden fenotip değerlerine geçilerek bulunan sayıların $f3$ fonksiyonunda yerine yazılması ile gerçekleşir.

2.4.3.5 F4 fonksiyonu

Tek bir optimuma sahip olan fonksiyon ve kısıt koşulu aşağıda gösterilmiştir [7]:

$$f4 = \sum_{i=1}^{30} \left(i.x_i^{\,4} + Gauss(0,1)\right) \qquad ,-1.28 < x_i \leq 1.28 \qquad (2.11)$$

Fonksiyona eklenen Gauss gürültüsü fonksiyonun aynı noktada aynı değeri alamamasına neden olmaktadır. Bu test fonksiyonunda başarılı olamayan test fonksiyonları gürültülü veriler konusunda da başarılı olamayacaklardır.

Fonksiyonda fenotip uzayından genotip uzayına dönüşümde öncelikle ondalıklı sayılar 100 ile çarpılarak tamsayı haline getirilir. 256 adet tamsayı genotip uzayında ikili düzende 8 bit ile ifade edilebilir. Dönüşüm Şekil 2.22'de görülmektedir.

Fenotip Uzayı	Tamsayıya Geçiş	Genotip Uzayı	Tamsayı Karşılığı
- 1.27	→ -127 →	00000000	0
- 1.26	→ -126 →	00000001	1
.	.	.	.
.	.	.	.
.	.	.	.
1.27	→ 127 →	11111110	254
1.28	→ 128 →	11111111	255

Şekil 2.22 F4'ün tanımlı olduğu aralık için fenotipler ve genotip karşılıkları

Kromozom yapısı incelenecek olursa kromozom, her bir grup bir x_i sayısının değerini gösteren ve 8 adet birer bitlik genlerden oluşan 30 gen grubuna sahiptir. Yani her kromozom 30*8 = 240 genden oluşur. Çözüm adaylarının uygunluk derecelerinin hesaplanması x_i değerlerinin fenotip uzayındaki karşılıklarının bulunup $f4$ fonksiyonunda yerine yazılması ile gerçekleştirilir.

2.4.3.6 F5 fonksiyonu

Diğer fonksiyonlardan farklı olarak $f5$ fonksiyonu global optimumun yanında çok sayıda yerel optimuma da sahiptir. Toplumun yakınsayarak bir yerel optimuma takılmamasına dikkat edilmelidir. Fonksiyon ve kısıt koşulu 2.12 denkleminde gösterilmiştir [7].

$$f5 = 0.002 + \sum_{j=1}^{25}\left(\frac{1}{j} + \sum_{i=1}^{2}\left(x_i - a_{ij}\right)^6\right)^{-1} \qquad ,-65536 < x_i \le 65536 \qquad (2.12)$$

Fonksiyondaki a_{ij} 2 satırlı, 25 sütunlu bir matristir.

$$[a_{ij}] = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & \cdots & 32 & 32 & 32 \end{bmatrix}$$

Fonksiyonda fenotip uzayı tamsayılardan oluşmaktadır. Fenotip uzayında 131072 adet çözüm adayı bulunduğundan her çözüm adayı için ikili düzende 17 bitlik bir gösterime ihtiyaç olur. Fenotip uzayından genotip uzayına dönüşüm Şekil 2.23'te gösterilmiştir.

Fenotip Uzayı	Genotip Uzayı	Tamsayı Karşılığı
- 65535	00000000000000000	0
- 65534	00000000000000001	1
.	.	.
.	.	.
.	.	.
65535	11111111111111110	131070
65536	11111111111111111	131071

Şekil 2.23 F5'in tanımlı olduğu aralık için fenotipler ve genotip karşılıkları

x_1 sayısı için 17, x_2 sayısı için 17 olmak üzere her kromozom toplam 34 genden oluşur. Çözüm adaylarının uygunluk dereceleri x_1 ve x_2 sayılarının fenotiplerinin bulunup bu değerlerin $f5$ fonksiyonunda yerine konulması ile hesaplanır.

2.4.3.7 F7 fonksiyonu

F7 fonksiyonu geniş bir araştırma uzayına ve çok sayıda yerel minimuma sahip olması nedeniyle evrimsel algoritmalar için oldukça zor bir problemdir. Fonksiyon ve kısıt koşulu aşağıda gösterilmiş olup fonksiyon verilen aralıkta milyonlarca yerel optimuma sahiptir [7].

$$f7 = 20A + \sum_{i=1}^{20}\left(x_i^{\,2} - 10.\cos(2\pi.x_i)\right) \quad , A = 10 \qquad ,-5.12 < x_i \le 5.12 \qquad (2.13)$$

Fonksiyonun çözüm adaylarının değerlerinin fenotip uzaydan genotip uzaya dönüşmesinde önceki bölümlerde anlatıldığı gibi her ondalık sayı 100 ile çarpılarak tamsayıya çevrilmiş ve genotip uzayında bir noktaya karşılık getirilmiştir. Fenotip uzayından genotip uzayına dönüşüm Şekil 2.19'dan da incelenebilir.

Problemin optimum çözümünün aranmasında kullanılan kromozomlar her x_i için 10 gen olmak üzere aranan 20 x_i sayısı olduğundan 20*10 = 200 genden oluşurlar. Çözüm adayının uygunluk derecesi fenotip uzayında değerleri bulunan x_i değişkenlerinin değerlerinin $f7$ fonksiyonunda yerine konulması ile hesaplanır.

3. PyLEA PAKETİ

3.1 Giriş

Gerçekleştirilen PyLEA yapısı daha geniş kapsamlı bir projenin parçasıdır.Uzun vadede çalışmalar internet üzerinden bir optimizasyon çözüm platformu olarak hizmet verecek bir yapının tasarlanması ve kullanıcıların geliştirilecek olan XML tabanlı bir dil sayesinde optimizasyon problemini belirterek problemine ilişkin çözümü elde etmesi üzerinde yoğunlaşacaktır. Bu projenin bir parçası olarak PyLEA yapısından önce kullanıcının evrimsel algoritma yaklaşımını gerçeklemesini sağlayan bir yapı ve yapının kullanımı için internet tabanlı bir arayüz tasarlanmıştır. Bu yapı ile ikili tipte genler taşıyan bireylerden oluşan bir toplum yaratılarak yapıda tanımlanmış her operatör için tanımlanmış metotlardan biri seçilip evrimsel algoritma koşturulabilmektedir [9]. Bu çalışmadan sonra daha esnek ve daha gelişmiş bir iskelet yapı olan PyLEA tasarlanmıştır.

Bu bölümde evrimsel algoritmalar için uygulama ve geliştirme yazılım altyapısı olan PyLEA paketi tanıtılmıştır. Öncelikle PyLEA hakkında genel bir bilgi verilmiş daha sonra PyLEA paketinin yapısı, paketi oluşturan alt paketler ve yapıdaki hiyerarşi açıklanmıştır. Sonrasında PyLEA paketinin alt paketlerinde tanımlanmış olan sınıflar tanıtılarak bu sınıfların metotları hakkında bilgi verilmiştir. Bir maksimizasyonu probleminin çözümü için PyLEA paketi ile geliştirilen basit genetik algoritma kodu örnek olarak gösterilmiş ve kod açıklanmıştır. Ayrıca PyLEA paketinden başka kullanıcının evrimsel algoritma yaklaşımını geliştirmesine yardımcı olmak amacıyla oluşturulmuş benzer iskelet yapılar tanıtılmıştır. Bölümün sonunda ise PyLEA paket yapısının sahip olduğu avantaj ve dezavantajlar tartışılmıştır.

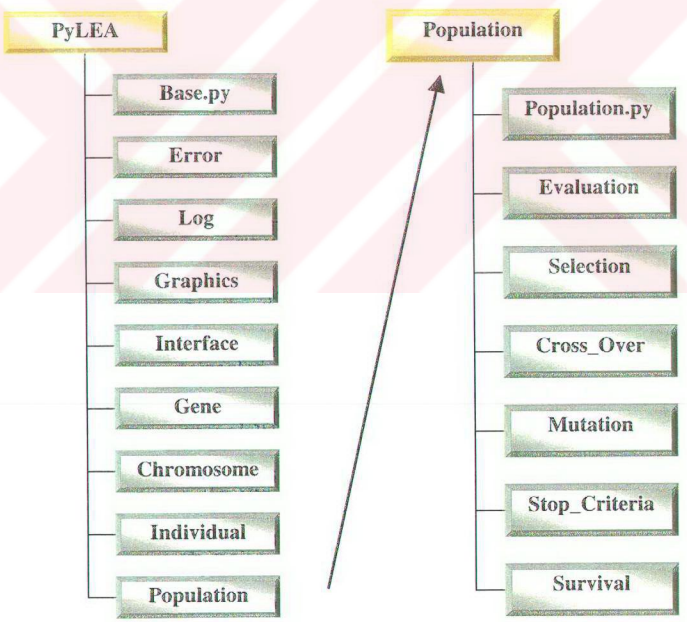
3.2 PyLEA Tanıtımı

PyLEA evrimsel algoritmaların eğitimi, çeşitli problemlerin çözümünde uygulanması ve akademik çalışmalarda kullanılması amacıyla Python dili kullanılarak geliştirilmiş bir paket yapısıdır. Python dili C diline benzer, öğrenilmesi ve kullanımı kolay olan

nesneye dayalı bir dildir. Python dili kullanılması ile oldukça esnek bir yapı oluşturulabilmiştir. PyLEA paketinde kullanıcının kendi evrimsel algoritma yaklaşımını oluştururken kullanabileceği pek çok operatör ve metotların yanısıra bazı örnek problemler de gerçeklenmiştir. Kullanıcı kendi algoritmasında paketin operatör ve metotlarını istediği sırada kullanabilir. Varolan operatörler ve örnek problemlerin yanısıra kullanıcının kendi geliştirdiği operatörleri ve kendi tanımlayacağı problemini de yapıya dahil etmesine imkan verecek şekilde PyLEA tasarlanmıştır.

3.3 Paket Yapısı

Python dilinde paket modüller veya alt paketlerden oluşan bir yapıdır. Paket yapısı modülerlik ve hiyerarşi sağlar. PyLEA paket yapısı .py uzantılı dosyalardan ve alt dizinlerden oluşmuş olan bir dizindir. Şekil 3.1'de PyLEA paket yapısı ve PyLEA paketinin bir alt paketi olan *Population* paketinin yapısı gösterilmiştir.

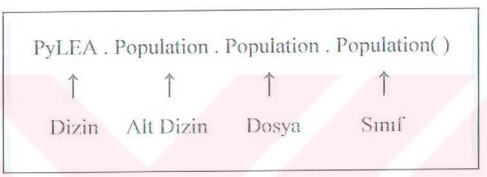


Şekil 3.1 PyLEA paketi ve *Population* altpaketi

PyLEA paketinde Base.py dosyası gen, kromozom, birey ve toplum sınıflarınca kullanılacak ortak metotları içerir. Diğer alt paketler ise gruplaşma sağlamak

amacıyla kullanılmıştır. Örneğin PyLEA paketinin *Interface* adlı alt paketi EA arayüzünü oluşturan Python dosyaları ve bazı resim dosyalarını içerir. Bu alt paket arayüze ilişkin dosyaları bir araya getirmekte kullanılmıştır. Benzer şekilde *Population* alt paketinde toplumu oluşturacak olan *Population* sınıfını bulunduran Population.py dosyası ve topluma uygulanabilecek çeşitli operatörleri bulunduran alt paketler bulunmaktadır. Örneğin *Selection* adlı alt paket içinde bulunan Selection.py dosyası toplumdaki bireylere uygulanabilecek rulet çarkı, turnuva seçimi, v.b. gibi çeşitli ebeveyn seçimi metotlarını bulundurmaktadır.

Paket yapısında modüller ile dizinler arasındaki hiyerarşi nokta (.) operatörü ile tanımlanır. Şekil 3.2 bir örnek üzerinde hiyerarşik yapıyı göstermiştir.



Şekil 3.2 PyLEA paketinin hiyerarşik yapısı

Belirtilen özelliklerinin yanısıra kullanıcı istediği paket yapısını PyLEA paketine bir alt paket olarak ekleyerek PyLEA yapısını genişletebilir. Kullanıcının kendi paket yapısını oluşturması için yaratacağı dizinine __init__.py dosyasını eklemesi ve bu dosyada da eğer varsa dahil etmek istediği dosyaları belirtmesi yeterlidir. Kullanıcı PyLEA paketinin __init__.py dosyasındaki dahil edilecek dosya ve alt paketler arasına kendi paketinin ismini de yazarak PyLEA kodda dahil edildiğinde alt paketinin de beraberinde dahil edilmesini sağlayabilir.

PyLEA paketi dahil edildiğinde gen, kromozom, birey, toplum, istatistik ve grafik sınıfları da dahil edilmiş olur. Bu sınıflar, metotları ve özellikleri sonraki bölümlerde detaylı bir biçimde açıklanacaktır.

3.4 Sınıflar, Metotları ve Özellikleri

PyLEA paketi ile kullanıcı istediği tip ve sayıda genlerden oluşan kromozomlar, istenen sayıda kromozoma sahip bireyler ve istenen sayıda bireylerden oluşan bir toplum yaratabilir. Pakette tanımlanmış gen tiplerinin yanısıra kullanıcı tarafından yeni bir gen tipi de tanımlanabilir. Ayrıca kullanıcı gen, kromozom, birey ve toplum

sınıflarının metotlarının yanısıra kendi metotlarını da dahil ederek diğer metotlarla birlikte kullanabilir. Bu özelliklerin yanında kullanıcı yaratacağı toplum üzerinde pakette tanımlanmış operatörler ve metotlarını uygulayabileceği gibi kendi tanımlayacağı operatör ve metotları da uygulayabilir. Kullanıcı bu şekilde oluşturacağı evrimsel algoritma yaklaşımında algoritmaya ilişkin bazı istatistiksel verileri kaydedip bu verileri grafikler üzerinde inceleme imkanına da sahiptir. Gen, kromozom, birey ve toplum nesnelerinin nasıl oluşturulabileceği ve bütün bu özelliklerin nasıl gerçeklenebileceği sonraki bölümlerde anlatılmıştır.

3.4.1 Gen sınıfı

Gen sınıfı PyLEA paketinin *Gene* adlı alt paketinin Gene.py dosyasında tanımlanmıştır. Gen sınıfını kullanarak dört tipte gen nesnesi yaratılabileceği gibi kullanıcı tarafından tanımlanan yeni bir gen tipi de oluşturulabilir. Tablo 3.1'de gen tipleri ve taşıyabilecekleri değerlere örnekler gösterilmiştir. İkili tipte genler 0 veya 1 değerlerinden birini alabilir. Tamsayı tipte genler oluşturabilmek için ise bir tamsayı aralığı belirtilmelidir. Gen bu aralıkta rasgele seçilmiş bir tamsayı değerine sahip olacaktır. Kayan noktalı gen tiplerinde de genin değer alabileceği bir aralık belirtilir. Gen, aralığın ilk değeri dahil, son belirtilen değeri ise dahil olmamak üzere verilen aralıkta rasgele bir kayan noktalı sayı değerine sahip olacaktır. Küme tipi genlerde ise genin değer alması istenen küme belirtilmelidir. Genin değeri küme elemanlarından rasgele birisi olacaktır. Bu tiplerin dışında kullanıcı isteyeceği bir gen tipini tanımlayıp genin değerini belirleyebilir.

Tablo 3.1 Gen tipleri ve sayısal örnekler

Kısaltma	Gen Tipi	Değer Aralığı Örneği	Değer Örneği
b	İkili	0 V 1	0
i	Tamsayı	[2-345]	148
f	Kayan Noktalı	[0.88-6.796)	4.7865
s	Küme	{"Kuzey","Güney","Doğu","Batı"}	"Doğu"
c	Kullanıcı Tanımlı	–	–

Oluşacak gen nesnesine bakılacak olursa gen nesnesi, genin değerini taşıyacak olan bir value değişkeni ve dict adı verilen bir sözlük yapısı bulunduracaktır. Sözlükte tamsayı ve kayan noktalı sayı tipi genler için değer aralığı, küme tipi genler için ise küme listesi saklanır. Kullanıcı sözlüğe istediği anahtar-veri çiftini ekleyebilir.

Örneğin genin mutasyon oranı ya da çaprazlaşma oranının farklı olması isteniyorsa sözlüğe pm ve/veya pc anahtarları ve değer(ler)i eklenebilir ("pm":0.005, "pc":0.76 gibi). Bu değerler mutasyon ve çaprazlaşma operatörlerince dikkate alınacaktır. Ayrıca kullanıcının gene varolan value ve dict yapılarının yanısıra istediği bir veri yapısını da eklemesi mümkündür. Şekil 3.3 tamsayı tipinden bir gen nesnesinin oluşturulmasında kullanılan hiyerarşiyi ve oluşan genin bulundurduğu verileri göstermektedir.

```
Gen = PyLEA.Gene.Gene.Gene( )

    Gen.value = 28
    Gen.dict  = { "aralık" : [ 1-50 ] }
```

Şekil 3.3 Gen nesnesi oluşturmak için hiyerarşik yazılım ve tamsayı tipinde oluşturulmuş bir gen nesnesinin yapısı

Gen sınıfında pek çok metot tanımlanmıştır. Kullanıcı bu metotların yanında kendi tanımlayacağı metotları da dahil ederek kullanabilir. Bu metotlardan önem taşıyan bazıları Tablo 3.2'de gösterilmiştir.

Gen nesnesine istenilen veri yapılarının eklenebilmesi, sınıf metotlarının yanısıra kullanıcı tarafından tanımlanmış metotların dahil edilebilmesi ve kullanıcının yeni gen tipleri yaratabilmesine imkan tanınması gibi özellikler yapıyı oldukça esnek bir hale getirmiştir. Aynı özellikler kromozom, birey ve toplum sınıfları için de geçerlidir.

Tablo 3.2 Gen sınıfı metotları

Metot	İşlev
binaryGene	: Gene 0 veya 1 değerlerinden birini rasgele atar
integerGene	: Verilen aralıkta rasgele bir tamsayı değeri seçerek gene atar
setGene	: Verilen küme elemanlarından birini rasgele seçerek genin değerine atar
floatGene	: Verilen aralıkta (üst sınır dahil olmayacak şekilde) rasgele bir kayan noktalı sayı değeri seçerek gene atar
get	: Genin değerini, ya da sözlükteki veri/verilerin değer(ler)ini döndürür
set	: Gene ya da sözlüğündeki verilere istenilen değerin atanmasını sağlar
import_	: Belirtilen dosyadaki fonksiyonun gen nesnesinin metotları arasına dahil edilmesini sağlar
copyDict	: Belirtilen ikinci genin sözlüğündeki verileri genin sözlüğüne kopyalar

3.4.2 Kromozom sınıfı

Genlerin bir araya gelmesiyle oluşan kromozom nesnesini tanımlayan kromozom sınıfı PyLEA paketinin *Chromosomes* alt paketindeki Chromosomes.py dosyasında bulunur. Kromozom kullanıcının istediği sayı ve tipte genden oluşur. Kromozom nesnesi kromozomun taşıdığı değeri belirten, ki bu değer de kromozomun taşıdığı genlerin değerlerinden oluşan bir listedir, bir value listesi, dict sözlüğü ve genleri tutan bir genes listesinden oluşmaktadır. Şekil 3.4 beş adet küme tipinden gen taşıyan bir kromozom yapısını göstermektedir.

```
Kromozom = PyLEA.Chromosomes.Chromosomes.Chromosomes( )

    Kromozom.value = [ "b", "d", "a", "e", "b" ]
    Kromozom.dict  = { }
    Kromozom.genes = [ gen1, gen2, gen3, gen4, gen5 ]
```

Şekil 3.4 Kromozom nesnesi oluşturmak için hiyerarşik yazılım ve küme tipinde genlere sahip bir kromozom nesnesinin yapısı

Gen yapısında olduğu gibi kromozom yapısında da kullanıcı sözlüğe istediği anahtar-veri çiftini ekleyebilir. Kromozomun mutasyon oranı ya da çaprazlaşma

oranının farklı olması isteniyorsa sözlüğe pm ve/veya pc anahtarları ve değer(ler)i eklenir. Mutasyon ve çaprazlaşma operatörlerince önce genin dict sözlüğüne bakılır burada oranlar bulunmadığı taktirde kromozomun dict sözlüğündeki oranlar dikkate alınır. Yine gen yapısında olduğu gibi kullanıcının kromozoma varolan value, dict ve genes yapılarının yanısıra istediği bir veri yapısını da eklemesi mümkündür.

Kromozom sınıfında tanımlanmış olan pek çok metotla birlikte kullanıcı kendi tanımlayacağı metotları da kromozoma dahil edip kullanabilir. Kromozom sınıfının metotlarından bazıları Tablo 3.3'ten incelenebilir.

Tablo 3.3 Kromozom sınıfı metotları

Metot	İşlev
append	: Kromozoma istenilen tipte yeni bir gen ekler
remove	: Kromozomdan istenilen geni çıkarır
get	: Kromozomun, kromozomdaki bir genin, kromozomun sözlüğündeki anahtar veya anahtarların değer(ler)ini döndürür
set	: Kromozomun, kromozomdaki bir/birkaç genin, kromozomdaki genin sözlüğünde bulunan bir anahtarın ya da kromozomun sözlüğünde bulunan bir anahtarın değerine istenilen değeri atar
import_	: Belirtilen dosyadaki kullanıcı tarafından tanımlanmış metodu kromozom nesnesinin metotları arasına dahil eder
insert	: Kromozom üzerinde istenilen yere bir gen eklenmesini sağlar
reverse	: Kromozom üzerinde belirli bir aralıktaki genlerin sırasını ters çevirir
mix	: Kromozom üzerinde belirli bir aralıktaki genlerin sırasını karıştırır
swap	: İki kromozomun aynı sıradaki genlerinin yer değiştirmesini sağlar
len	: Kromozomun kaç gen taşıdığını geri döndürür

3.4.3 Birey sınıfı

Birey sınıfı PyLEA paketinin *Individual* alt paketinde bulunan Individual.py dosyasında tanımlanmıştır. Bir birey istenilen tip ve sayıda genlerden oluşan kromozom ya da kromozomlardan oluşur. Şekil 3.5'te iki kromozomlu ve her kromozomu ikili tip genlerden oluşmuş bir birey gösterilmiştir. Birey sınıfından oluşmuş her bireyin kromozomlarının değerini taşıyan bir value listesi, başlangıçta boş olup istenilen anahtar-veri çiftini bulundurabilecek bir dict sözlüğü ve bireyin kromozomlarını tutan chromosomes listesi bulunur.

```
Birey = PyLEA.Individual.Individual.Individual( )

Birey.value       = [ [ 1,0,1,0,0], [0,1,1,0,1] ]
Birey.dict        = { }
Birey.chromosomes = [ kromozom1, kromozom2 ]
```

Şekil 3.5 Birey nesnesi oluşturmak için hiyerarşik yazılım ve ikili tipte genlere sahip iki kromozomun oluşturduğu birey nesnesinin yapısı

Gen ve kromozom yapılarında olduğu gibi birey yapısında da kullanıcı sözlüğe istediği anahtar-veri çiftini ekleyebilir. Bireyin mutasyon oranı ya da çaprazlaşma oranının farklı olması isteniyorsa sözlüğe pm ve/veya pc anahtarları ve değer(ler)i eklenebilir. Gen ve kromozom yapılarında olduğu gibi kullanıcının bireye varolan value, dict ve chromosomes yapılarının yanısıra istediği bir veri yapısını da eklemesi mümkündür.

Birey sınıfında tanımlanmış olan bazı metotlar Tablo 3.4'te incelenebilir. Bu metotların yanısıra kullanıcının kendi tanımlayacağı metotları da dahil edebilmesi birey sınıfı için de geçerlidir.

Tablo 3.4 Birey sınıfı metotları

Metot	İşlev
append	: Bireye istenilen tip ve sayıda gene sahip yeni bir kromozom ekler
remove	: Bireyden istenilen kromozomu çıkarır
get	: Bireyin, bireydeki kromozomların ya da kromozomlardaki genlerin değerlerini ya da sözlüklerindeki parametrelerin değerlerini geri döndürür
set	: Bireyin, bireydeki kromozomların ya da kromozomlardaki genlerin değerlerine ya da sözlüklerindeki bir parametreye istenilen değeri atar
import	: Belirtilen dosyadaki kullanıcı tarafından tanımlanmış metodun birey nesnesinin metotları arasına dahil edilmesini sağlar
len	: Bireydeki kromozom sayısını geri döndürür

3.4.4 Toplum sınıfı

Şekil 3.6'dan da görülebileceği gibi bir toplum oluşturabilmek için PyLEA paketinin *Population* adlı alt paketinde bulunan Population.py dosyasındaki *Population* sınıfı kullanılır. Toplum sınıfında toplumdaki bireylerin değerlerinden oluşan bir value

listesi, çeşitli parametreleri anahtar-veri çifti biçiminde saklayacak olan dict sözlüğü, bireyleri tutan individuals listesi, ebeveynleri bulunduracak olak matingpool listesi ve çocuklar oluştuktan sonra çocukları tutacak olan offspring listesi bulunmaktadır. Şekil 3.6' da [1-10] aralığında değer alabilen tamsayı genlerden oluşan çift kromozomlu üç bireyden oluşan bir toplum örnek olarak gösterilmiştir. Belirtilmediği taktirde toplumun sözlüğüne mutasyon olasılığı 0.001, çaprazlaşma olasılığı ise 0.80 olarak aktarılır. Ayrıca en iyi bireyin de tanımlanması gerekmektedir. Aksi belirtilmedikçe uygunluk derecesi en yüksek olan birey en iyi birey olarak kabul edilir.

```
Toplum = PyLEA.Population.Population.Population( )

Toplum.value      = [ [[ 2, 5, 3, 3, 1 ], [ 4, 7, 6, 5, 8]],
                      [[ 9, 2, 8, 1, 6 ], [ 1, 5, 3, 2, 2]],
                      [[ 4, 9, 3, 6, 2 ], [ 1, 1, 2, 4, 6]] ]
Toplum.dict       = { "pm":0.001, "pc":0.80, "best":"max" }
Toplum.individuals = [ birey1, birey2, birey3 ]
Toplum.matingpool = [ ]
Toplum.offspring  = [ ]
```

Şekil 3.6 Toplum nesnesi oluşturmak için hiyerarşik yazılım ve tamsayı tipte genlere sahip iki kromozomlu bireylerin oluşturduğu toplum nesnesinin yapısı

Toplum ilk yaratıldığında matingpool ve offspring listeleri boştur. Toplumda ebeveyn seçimi sonucunda seçilen ebeveynler matingpool listesine aktarılır. Çaprazlaşma işlemi matingpooldaki bu ebeveynler üzerinde gerçekleştirilir. Çaprazlaşma işlemi sonucunda oluşan çocuklar offspring listesinde tutulur. Mutasyon offspring listesindeki çocuklar üzerinde gerçekleştirilir. Kullanıcı bu yapıların yanısıra topluma kendi veri yapılarını da ekleyebilir.

Toplum üzerinde gerçeklenebilinecek pek çok metot tanımlanmıştır. Tablo 3.5 bu metotlardan bazılarını göstermektedir. Önceki sınıflarda olduğu gibi toplum sınıfına da kullanıcı tarafından tanımlanmış metotların eklenmesi mümkündür.

Tablo 3.5 Toplum sınıfı metotları

Metot	İşlev
append	: Topluma yeni bir birey ekler
remove	: Toplumdan istenilen bireyi çıkarır
insert	: Toplumdaki birey listesinin istenilen sırasına birey ekler
get	: Toplumun, bireylerin, kromozomların, kromozomlardaki genlerin değerlerini veya sözlüklerindeki parametrelerin değerlerini döndürür
set	: Toplumun, bireylerin, kromozomların, kromozomlardaki genlerin değerlerine veya sözlüklerindeki parametrelerin değerlerine istenilen değeri atar
import_	: Belirtilen dosyadaki kullanıcı tarafından tanımlanmış metodun toplum nesnesinin metotları arasına dahil edilmesini sağlar
len	: Toplumdaki birey sayısını döndürür
getIndv	: Yeni bir birey oluşturur ve bireyi geri döndürür
selection	: Toplumdaki bireylere uygulanacak ebeveyn seçimi metodunun toplum metotları arasına dahil edilmesini sağlar, kullanıcı kendi metodunun bulunduğu dosyayı parametre olarak aktararak kendi metodunu kullanabilir
cross_over	: Toplumdaki bireylere uygulanacak çaprazlaşma metodunun toplum metotları arasına dahil edilmesini sağlar, selection metodunda olduğu gibi kullanıcının kendi metodunu dahil etmesi mümkündür
mutation	: Toplumdaki bireylere uygulanacak mutasyon metodunun toplum metotları arasına dahil edilmesini sağlar, önceki metotlarda olduğu gibi kullanıcı kendi tanımladığı metodunu da dahil edebilir
evaluate	: Toplumdaki bireylerin uygunluk derecesinin hesaplanmasında kullanılacak uygunluk fonksiyonunu toplumun metotları arasına dahil eder, kullanıcı tarafından tanımlanacak uygunluk fonksiyonları da dahil edilebilebilmektedir
survive	: Toplumdaki bireylere uygulanacak hayatta kalış seçimi metodunun toplum metotları arasına dahil edilmesini sağlar, kullanıcı kendi tanımladığı metodunun bulunduğu dosyayı parametre olarak aktararak kendi metodunu da toplum metotları arasına dahil edebilir
stop	: Evrimsel algoritmanın ne zaman sonlanacağını belirten sonlanma metodunu toplum metotları arasına dahil eder, kullanıcının kendi sonlanma metodunu dahil etmesine de imkan tanınır
fitness	: Aktarılan parametrelere bağlı olarak toplumdaki en büyük uygunluk derecesine sahip bireyi, en düşük uygunluk derecesine sahip bireyi, toplumun toplam ya da ortalama uygunluk derecesini döndürür
convergence	: Toplumdaki yakınsamış birey sayısını döndürür
diversity	: Toplumdaki farklı birey sayısını döndürür
diversityH	: Bireyler arasındaki Hamming uzaklıklarının ortalamasını döndürür
initialize	: PyLEA paketindeki tüm operatörlerin metotlarının toplum metotları arasına dahil edilmesini sağlar

3.4.5 İstatistik Sınıfı

İstatistik sınıfı PyLEA paketinin *Log* alt paketindeki Log.py dosyası içinde tanımlanmıştır. Evrimsel algoritma sonlandıktan sonra algoritmaya ilişkin bilgiler ve

toplumun tüm nesilleri için hesaplanmış olan istatistiksel veriler *Log* alt paketinde bulunan Log.txt dosyasından incelenebilir. *Log* alt paketinde bulunan Statistics.txt dosyası ise toplumun nesillerine ait istatistiksel verilerin saklanmasında kullanılan bir dosyadır. Evrimsel algoritmaya ilişkin verileri ve toplumda oluşacak her nesil için istatistiksel verileri saklamak isteyen kullanıcı oluşturacağı topluma istatistik sınıfına ait bir nesne ekler ve algoritmada istatistik sınıfının metotlarından faydalanarak istenen verilerin bulunup kaydedilmesini sağlar. Log.txt dosyasında bulunan evrimsel algoritmaya ilişkin verilerin neler olduğu ait oldukları başlıklara göre aşağıda açıklanmıştır.

1. Toplum Parametreleri : Toplumdaki birey sayısı, her bireyin sahip olduğu kromozom sayısı, bir kromozomu oluşturan gen sayısı ve tipi gibi topluma ilişkin veriler hakkında bilgi verir.

2. Toplum Sözlüğü : Toplumun dict sözlük yapısında bulunan anahtarları ve her anahtara karşılık gelen değeri gösterir.

3. Operatörler : Topluma uygulanmak üzere seçilmiş operatörler ve metotlarının neler olduğunu sırası ile belirtir.

4. Uygunluk : Toplumun bireylerinin uygunluk derecesini hesaplayacak fonksiyonun hangisi olduğunu belirtir.

5. Hayatta Kalış : Topluma uygulanacak hayatta kalış metodunu belirtir.

6. Sonlanma Kriteri : Evrimsel algoritmanın sonlanma kriteri metodunu belirtir.

Log.txt dosyasında bulunan ve toplumun her nesli için saklanan istatistiksel veriler ise aşağıda açıklanmıştır.

1. Toplum listeleri : Kullanıcı tarafından verilen parametrelere bağlı olarak toplumun bireylerinin, ebeveyn havuzundaki ebeveynlerin ve oluşan çocukların listelerinin biri, birkaçı ya da tümü saklanabilir. Kullanıcı tarafından belirtilmedikçe toplum listeleri istatistik dosyasında saklanmaz. Listelerin istatistik dosyasında saklanması ile evrimsel algoritmanın sonunda kullanıcı her nesildeki bireylerin, ebeveynlerin ve/veya çocukların genotiplerini inceleyebilme imkanı bulur.

2. En iyi birey : İstatistik dosyasında her nesil için toplumun bireyleri arasındaki en iyi bireye ait genotip kaydedilir.

3. En kötü birey : İstatistik dosyasında her nesil için toplumdaki bireyler arasında bulunan en kötü uygunluk derecesine sahip olan en kötü bireye ait genotip kaydedilir.

4. Maksimum uygunluk : Her nesil için toplumdaki bireylerin uygunluk dereceleri arasındaki en büyük uygunluk derecesi kaydedilir.

5. Minimum uygunluk : Her nesil için toplumdaki bireylerin uygunluk dereceleri arasındaki en küçük uygunluk derecesi kaydedilir.

6. Ortalama uygunluk : Her nesil için toplumdaki bireylerin ortalama uygunluk derecesi hesaplanarak kaydedilir.

7. Çevrim içi (online) performans : Her nesil için çevrim içi performansın hesaplanmasında önceki nesillere ait ortalama uygunluk derecelerinden faydalanılır. Oluşan ilk nesil için çevrim içi performans ortalama uygunluk derecesine eşittir. Sonraki nesiller için ise ilk nesilden başlayarak toplumun ortalama uygunluk dereceleri toplanır ve ortalaması alınır. Bulunan ortalama uygunluk değerlerinin ortalaması çevrim içi performans olarak kaydedilir.

8. Çevrim dışı (offline) performans : Her nesil için çevrim dışı performansın hesaplanması çevrim içi performansın hesaplanmasına benzerlik gösterir. Ancak çevrim dışı performansın hesaplanmasında önceki nesillere ait ortalama uygunluk dereceleri yerine en iyi uygunluk dereceleri kullanılır. Oluşan ilk nesil için çevrim dışı performans nesil içinde bulunan en iyi uygunluk derecesine eşittir. Sonraki nesiller için ilk nesilden başlayarak toplumdaki en iyi uygunluk dereceleri toplanır ve ortalaması alınır. Bulunan en iyi uygunluk derecelerinin ortalaması çevrim dışı performans olarak kaydedilir.

9. Yakınsama : Her nesil için toplumdaki bireyler arasında kaç tanesinin yakınsadığı istatistik dosyasında kaydedilir.

10. Çeşitlilik : İstatistik dosyasına kaydedilecek toplumun çeşitliliğinin hesaplanması iki şekilde olabilir. İlk yöntemde her nesil için toplumda bulunan farklı birey sayısı hesaplanarak kaydedilir. Diğer yöntem ise Hamming çeşitliliğidir. Bu yöntemde toplumdaki bireyler arasındaki Hamming uzaklıkları tek tek hesaplanır. Daha sonra hesaplanan Hamming uzaklıkları toplanır ve ortalama Hamming uzaklığı hesaplanır. Hesaplanan ortalama Hamming uzaklığı toplumun Hamming çeşitliliği olarak kaydedilir [10].

Log.txt dosyasında bulunan son veri evrimsel işlemler süresince bulunmuş en iyi bireyin genotipi ve uygunluk derecesidir.

İstatistik sınıfının bazı metotları ve bu metotların işlevleri Tablo 3.6'da gösterilmiştir.

Tablo 3.6 İstatistik sınıfı metotları

Metot	İşlev
writeParameters	: Toplum parametreleri, operatörler ve metotları v.b. gibi evrimsel algoritmaya ilişkin parametreleri içeren bir karakter katarı geri döndürür
writeStatistics	: En iyi birey, en kötü birey v.b. gibi nesillere ait saklanmış istatistiksel verileri içeren bir karakter katarı geri döndürür
writeBest	: Evrimsel algoritma süresince bulunmuş en iyi bireyin genotipini ve uygunluk derecesini içeren bir karakter katarı geri döndürür
loadStatistics	: İstatistik sınıfının liste yapısında saklanan istatistiksel verilerinin Statistics.txt dosyasına yüklenmesini sağlar
dumpStatistics	: Yüklenmiş olan istatistiksel verilerin Statistics.txt dosyasından okunarak istatistik sınıfının liste yapısına aktarılmasını sağlar
saveInfo	: Evrimsel algoritmaya ilişkin parametrelerin bir dosyaya yazılmasını sağlar
saveStatistics	: Hesaplanan istatistiksel verilerin bir dosyaya yazılmasını sağlar
saveBest	: Algoritma süresince bulunmuş en iyi bireyin genotipi ile uygunluk derecesinin bir dosyaya yazılmasını sağlar
openFile	: İstenen verileri yazmak üzere Log.txt dosyasını açar
closeFile	: Log.txt dosyasına bulunan en iyi bireyin genotipi ile uygunluk derecesini kaydederek dosyayı kapatır

3.4.6 Grafik Sınıfları

PyLEA paketindeki *Graphics* adlı alt paketin Graphics.py dosyasında bulunan grafik sınıfları evrimsel algoritma sonlandıktan sonra algoritmaya ilişkin çeşitli istatistiksel verilerin grafiklerle incelenebilmesine imkan tanımaktadır. Grafiklerin oluşturulabilmesi için evrimsel algoritma süresince topluma ait istatistikler kaydedilmiş olmalıdır. Dosyada *LineChart* ve *BarChart* adı verilen iki sınıf tanımlanmıştır. Grafikler, *LineChart* sınıfı kullanıldığında çizgisel grafikler ve *BarChart* sınıfı kullanıldığında çubuk grafikleri olmak üzere iki farklı biçimde oluşturulabilir. Evrimsel algoritmaya ilişkin istatistiksel veriler grafik(ler) üzerinde incelenmek isteniyorsa topluma istenilen grafik sınıfı kullanılarak bir nesne eklenir.

Kullanıcı grafik üzerinde incelemek istediği istatistiksel veri ya da verileri grafik sınıfının metotlarını kullanarak belirler ve son olarak grafik sınıfının draw metodunu kullanarak istenilen grafiğin Adobe PDF dosyası içinde çizilmesini sağlar. Grafik sınıflarına ilişkin metotlar ve işlevleri Tablo 3.7'de gösterilmiştir.

Tablo 3.7 Grafik sınıfı metotları

Metot	İşlev
bestFit	: Toplumun her neslindeki en iyi uygunluk derecelerinin oluşturulacak grafik üzerinde gösterilmesini sağlar
avrFit	: Toplumun her nesli için ortalama uygunluk derecelerinin oluşturulacak grafik üzerinde gösterilmesini sağlar
online	: Toplumdaki nesillerin çevrim içi performanslarının oluşturulacak grafik üzerinde gösterilmesini sağlar
offline	: Toplumdaki nesillerin çevrim dışı performanslarının oluşturulacak grafik üzerinde gösterilmesini sağlar
convergence	: Toplumun her nesli için yakınsamış birey sayılarının oluşturulacak grafik üzerinde gösterilmesini sağlar
diversity	: Toplumun her nesli için farklı birey sayılarının oluşturulacak grafik üzerinde gösterilmesini sağlar, istatistiklerde Hamming çeşitliliği seçilmiş ise her nesil için hesaplanmış olan Hamming çeşitliliğini grafikte gösterir
draw	: Grafik üzerinde gösterilmesi istenen veriler belirlendikten sonra bu metot kullanılarak grafik .pdf uzantılı bir dosyada çizdirilir

3.5 Bir Maksimizasyonu Problemi İçin SGA Örnek Algoritması

PyLEA paket yapısı kullanılarak gerçeklenen bir basit genetik algoritmanın kodu Şekil 3.7' de gösterilmiştir. Algoritma bir maksimizasyonu problemine çözüm aramak maksadıyla geliştirilmiştir. Çözüm adayları ikili tipte genlerden oluşan tek kromozoma sahip bireylerdir. Basit genetik algoritmada ebeveyn seçimi işlemi rulet çarkı seçiminden, çaprazlaşma işlemi tek noktadan çaprazlaşmadan ve mutasyon işlemi ise ikili mutasyondan oluşur ayrıca hayatta kalış seçimi nesilsel ve sonlanma koşulu ise nesil sayına bağlı olmaktadır.

```
def sga( popsize, chrsize, popPc, popPm, stopParameter ) :
    import PyLEA
    cType = str(chrsize)+"b"
    pop = PyLEA.Population.Population.Population
    p = pop( psize = popsize, isize = 1, chr = cType, pc = popPc, pm =popPm )
    p.initialize( )
    p.log = PyLEA.Log.Log.Log( )
    p.log.openFile( )
    p.evaluate.oneMax( p.individuals )
    continue_ = True
    while continue_ :
        p.select.rouletteWheel( p )
        p.crossover.onePoint( p,popPc )
        p.mutate.binary( p,popPm )
        p.evaluate.oneMax( p.offspring )
        continue_ = p.stop.generationCount( p,stopParameter )
        p.log.saveStatistics( p,0 )
        p.survive.generational( p )
    p.log.closeFile( )
```

Şekil 3.7 PyLEA paketi kullanılarak bir maksimizasyonu probleminin basit genetik algoritma ile çözümünü aramak amacıyla geliştirilmiş kod

Basit genetik algoritmanın çalışması için sga fonksiyonuna toplumda varolması istenen birey sayısı (popsize), her bireyde bulunacak kromozomun taşıyacağı gen sayısı (chrsize) , çaprazlaşma olasılığı (p_c), mutasyon olasılığı (p_m) ve algoritmanın kaç nesil sonra sonlanması istenildiği bilgileri aktarılmalıdır. Evrimsel algoritma yaklaşımı PyLEA paketi kullanılarak geliştirileceğinden öncelikle PyLEA paketi dahil edilmiştir. Daha sonra her kromozomun taşıyacağı gen sayısı ve tipi belirlenir. Her kromozom ikili tipte genlerden oluşacaktır. Beşinci satırda ise istenilen sayıda bireye sahip (popsize kadar), her bireyin tek kromozoma sahip olduğu, her kromozomun istenen sayıda (chrsize kadar) ikili tipte gen taşıdığı bir toplum (p) yaratılmıştır. Toplumdaki bireyler için çaprazlaşma ve mutasyon olasılıkları toplumun dict sözlüğünde bulunur. Basit genetik algoritmada kullanılacak operatörlerin metotları PyLEA paketinde tanımlanmış olduğundan diğer bir deyişle kullanıcı tarafından tanımlanmış bir metot kullanılmayacağından toplumun *initialize* metodu çalıştırılmıştır. Bu metot toplum üzerinde kullanılmak üzere paketteki operatörlerin tüm metotlarını (yöntemlerini) toplum metotları arasına katar. Sonraki adımda evrimsel algoritmanın istatistiksel verilerini saklamak amacıyla bir *log* nesnesi topluma eklenir. Görüleceği gibi toplum nesnesinde varolmayan bir nesne toplum yaratıldıktan sonra eklenebilmiştir. İstatistiksel veriler *log* nesnesinin

openFile metodu ile açılan "Log.txt" dosyasına kaydedilecektir. Sonra, toplum yaratıldığında rasgele oluşturulmuş olan ve individuals listesinde bulunan bireylerin uygunluk dereceleri hesaplanır. Sonraki satırda evrimsel döngüye girilmesini sağlayacak olan *continue_* değişkenine *True* değeri atanır. Daha sonra evrimsel döngüye girilir. Sonlanma koşulu sağlanmadığı sürece döngü devam edecektir. Öncelikle rulet çarkı seçimi ile bireyler arasından sonraki nesil için ebeveyn olacaklar seçilir ve eşleşme havuzuna kopyalanır. Sonraki adımda eşleşme havuzundaki ebeveynlere tek noktadan çaprazlaşma uygulanır. Bu işlemin sonunda çocuklar oluşur ve toplumun offspring listesine aktarılırlar. Bu adımdan sonra oluşan çocuklar üzerinde ikili mutasyon işlemi uygulanır. İşlemin sonunda bazı çocuklar değişime uğramışlardır. Mutasyon işleminden sonra oluşmuş olan çocukların uygunluk dereceleri hesaplanır. Sonrasında ise sonlanma koşulu metodu çağırılır. Oluşmuş nesil sayısı parametre olarak aktarılmış nesil sayısına (stopParameter) ulaşmış ise metodun geriye döndüreceği *False* değeri *continue_* değişkenine atanacak ve tekrar döngüye girilmeyecektir, aksi taktirde metot geriye *True* değeri döndürür ve döngü bir nesil için daha devam eder. Döngünün altıncı adımında oluşmuş çocuklar için istatistiksel veriler dosyaya kaydedilir. Sonrasında ise nesilsel hayatta kalış işlemi gerçekleşir ve oluşmuş olan çocuklar bireylerin yerini alırlar. Döngü kullanıcı tarafından belirlenmiş nesil sayısına ulaşılıncaya dek sürer. Döngünün sonlanması ile algoritma süresince bulunmuş en iyi çözüm adayı kaydedilerek dosya kapatılır.

3.6 Benzer Çalışmalar

Evrimsel hesaplamalar ile kullanılması amacıyla pek çok yazılım geliştirilmiştir. Bu amaçla geliştirilen ilk yazılımlardan biri GENESIS olmuştur. GENESIS, C dili ile yazılmış bir sistemdir. Genetik algoritmalar kullanılarak fonksiyon minimizasyonu problemlerine çözüm geliştirilmesine yardımcı olmayı amaçlamaktadır. Bu sistemi kullanırken kullanıcının çözüm uzayında bir nokta verildiğinde bir uygunluk değeri döndürecek olan uygunluk fonksiyonunu yazması gerekmektedir. En son versiyon olan 5.0 versiyonunda genotiplerin karakter katarı şeklinde ya da reel sayılardan oluşan vektörler şeklinde gösterimleri olabileceği belirtilmiştir [11]. Diğer bir yazılım ise GENESIS 4.5'in bir uzantısı şeklinde C dili ile geliştirilen GENEsYs olmuştur. Bu yapıdaki bazı farklılıklar amaç fonksiyonlarının eklenmiş olması,

ebeveyn seçimi, çaprazlaşma ve mutasyon operatörleri için yeni metotlar eklenmiş olması ve genetik algoritmanın performansının takip edilebileceği bir rapor hazırlanması şeklindedir [12]. Kayan noktalı sayılardan oluşan vektörel genotip gösterimine dayanan diğer bir sistem GENOCOP'tur. Bu iki sistemden farklı olarak GENOCOP sistemi kısıt koşulları ile ilgili çeşitli metotlar sağlar. Temel olarak iki amaç bulunur : (1) varsa kısıt koşullarının oluşturduğu kümedeki eşitliklerin elenmesi ve (2) kromozomların geçerli çözümlerin oluşturduğu çözüm uzayı içinde kalmasını garanti edecek özel genetik operatörlerin tasarlanması [3]. C++ dili ile geliştirilmiş, nesneye dayalı bir iskelet yapı olan gpc++ ağaç tabanlı genetik programlamalarda kullanılmak üzere tasarlanmıştır [13]. EO ise C++ dili ile geliştirilmiş bir evrimsel hesaplama iskelet yapısıdır. Linux, Windows ve Solaris gibi çeşitli platformlarda çalışabildiği test edilmiştir [14]. Benzer şekilde Open BEAGLE C++ dili ile geliştirilmiş evrimsel hesaplamalarda kullanılabilecek nesneye dayalı bir iskelet yapıdır. Yapı kullanılarak genetik algoritmalar ve genetik programlamalar oluşturulmuştur. Gelecekteki çalışmalar ile evrimsel stratejilerin gerçeklenebilmesi planlanmaktadır. PyLEA yapısına benzer bir biçimde Open BEAGLE kullanılarak bireylerce oluşturulmuş toplum üzerinde çeşitli işlemler uygulanabilir. Yapıda referans sayacı kullanılarak otomatik çöp toplama (automatic garbage collection) özelliği getirilmiştir. Kullanıcı işlemler sırasında meydana gelen hatalar konusunda bilgilendirilmektedir [15, 16]. Diğer bir yapı olan ECJ ise evrimsel hesaplamalarda kullanılmak üzere Java ile tasarlanmıştır. Javanın yorumlanan bir dil olması nedeniyle yapı hızı konusunda dezavantajlıdır [17].

3.7 Avantaj ve Dezavantajlar

PyLEA Python dili kullanılarak geliştirilmiş nesneye dayalı bir iskelet yapıdır. Python dili yorumlanan bir dil olduğundan PyLEA paketi örneğin C ya da C++ gibi derleyicili dillerle geliştirilmiş iskelet yapılara göre daha yavaştır. Ancak bazı yöntemlerle yapı hızlandırılabilir. Bunun yanısıra Python, C dili ile benzerlik gösteren, nesneye dayalı, esnek, sentaksı basit ve öğrenimi kolay bir dildir. Bu nedenle PyLEA geliştirilmiş benzer yapılara göre öğrenimi ve kullanımı daha kolay olan bir yapıdır. PyLEA'nın Python dili ile geliştirilmiş olması sayesinde yapıda otomatik çöp toplama özelliği de varolmuştur. Python dilinin getirdiği avantaj sayesinde PyLEA ile evrimsel algoritma yaklaşımları diğer yapılardan daha hızlı bir

biçimde geliştirilebilir şöyle ki PyLEA paketinin kendisi de diğer yapılara kıyasla çok kısa bir sürede geliştirilebilmiştir. PyLEA platformdan bağımsız bir yapıdır. Ayrıca PyLEA esnek bir biçimde geliştirilmiştir. Pakette bulunan pek çok operatör ve metodun yanısıra kullanıcı kendi geliştirdiği operatörleri ve metotları da yapıya dahil ederek kullanabilir. Pakette tanımlanmış bazı örnek problemler de bulunmaktadır, kullanıcı bu problemleri ya da tanımlayıp dahil edeceği kendi problemini kullanarak evrimsel algoritma yaklaşımını oluşturabilir. PyLEA paketi kullanılarak oluşturulmuş nesnelere yeni nesneler eklenerek yapılar genişletilebilir. Çok yönlü kullanım sağlayan PyLEA, kullanımı kolay arayüzü, platformdan bağımsız ve esnek yazılımı, öğrenimi ve kullanımı kolay yapısı sayesinde benzer yapılardan üstünlük göstermektedir.

4. EA ARAYÜZÜ VE KULLANIM ÖRNEKLERİ

4.1 Giriş

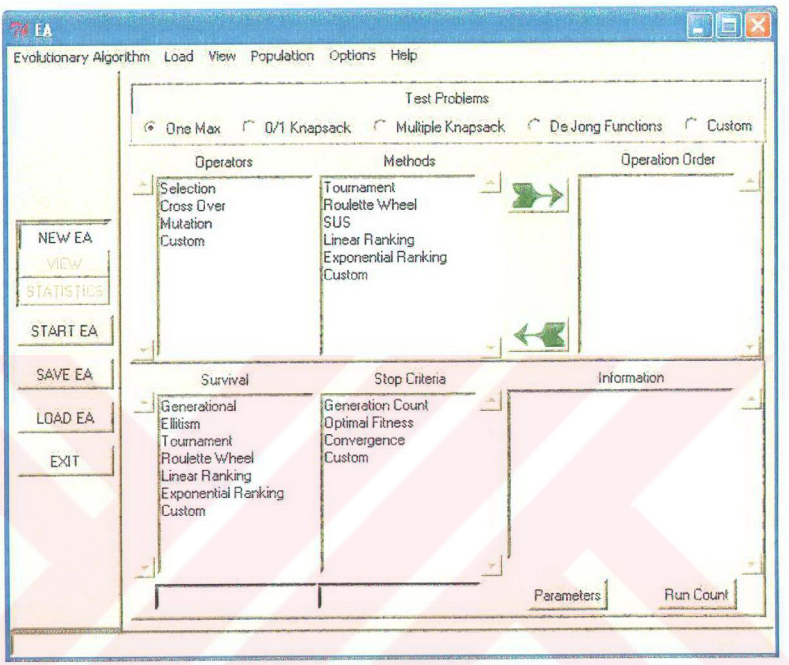
PyLEA paketinin kullanımı için tasarlanan evrimsel algoritma (EA) arayüzü kolay kullanımı ile çeşitli kullanıcı seviyelerine hizmet sunmaktadır. Bu bölümde PyLEA paket yapısının *Interface* alt paketinde gerçeklenmiş olan EA arayüzü tanıtılmıştır. Öncelikle EA arayüzüne ilişkin genel bilgiler verilmiş daha sonra arayüzde bulunan bazı seçenekler ve arayüzün bazı özellikleri anlatılmıştır. EA arayüzü farklı kullanım amaçlarına hizmet edebilmesi düşünülerek tasarlanmıştır. Sonraki bölümlerde bu kullanım amaçlarının neler olabileceği ve bu kullanım amaçları için EA arayüzünün nasıl kullanılacağı anlatılmıştır.

4.2 EA Arayüzü

Şekil 4.1 EA arayüzü çalıştırıldığında oluşan pencere yapısını göstermektedir. Arayüzde en yukarıda menü seçenekleri, solda gerekli pencerelerin sağ tarafta görünmesini sağlayan *NEW EA*, *VIEW* ve *STATISTICS* düğmeleri, evrimsel algoritmayı başlatan *START EA* düğmesi, evrimsel algoritmayı parametreleri ile kaydetmeyi sağlayan *SAVE EA* düğmesi, kaydedilmiş bir evrimsel algoritmanın yeniden yüklenmesini sağlayan *LOAD EA* düğmesi ve arayüzü sonlandıran *EXIT* düğmesi bulunmaktadır. Arayüzün sağ tarafında arayüz ilk çalıştırıldığında yeni bir evrimsel algoritma oluşturmayı sağlayacak olan *NEW EA* penceresi görünür. Bu durumda *NEW EA* butonu da basılmış haldedir.

EA arayüzünün sağ tarafında bulunan *NEW EA* penceresinin üst kısmında gerçeklenmiş olan örnek problemler görülmektedir. Başlangıç olarak bir maksimizasyonu seçeneği seçili durumdadır. Kullanıcı evrimsel algoritmaları kullanarak çözümünü aramak istediği problemini *Test Problems* bölümünü kullanarak seçer. Bu bölümde bulunan *Custom* seçeneği kullanıcının çözümünü bulmak istediği kendi probleminine ait uygunluk fonksiyonunu sisteme yüklemesini sağlamak amacıyla eklenmiştir. Örnek problemini seçen kullanıcı probleme ilişkin

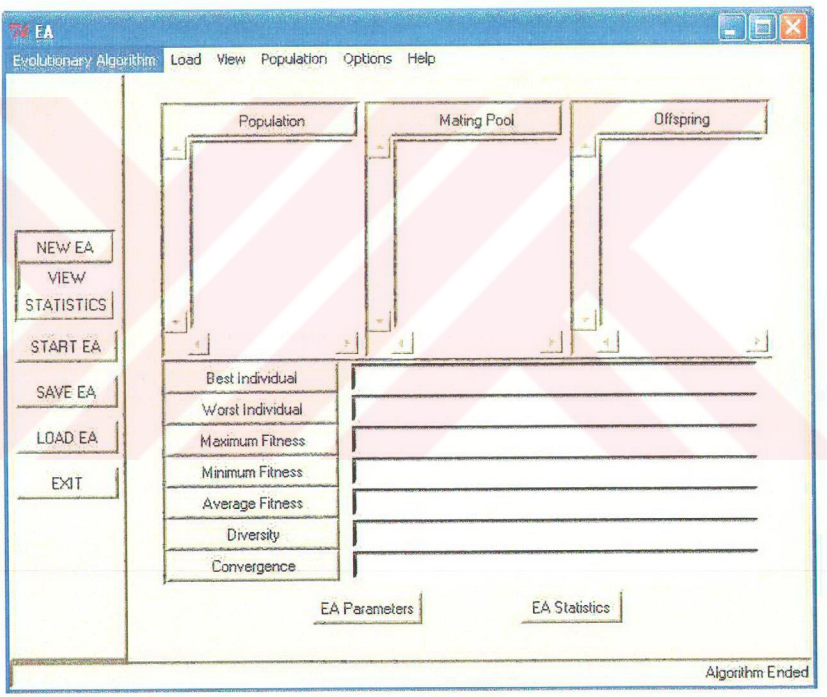
parametreleri belirtmek amacıyla *Parameters* düğmesine tıklar. Her problem için varsayılan parametrik değerler vardır, kullanıcı isterse bu değerleri değiştirebilir.



Şekil 4.1 EA arayüzü ilk çalıştırıldığında oluşan pencere

Test problemleri bölümünün hemen altında evrimsel algoritma operatörleri görülmektedir. Başlangıç olarak seçim operatörü (*Selection*) seçili durumdadır ve seçim operatörüne ait metotlar *Methods* adlı liste kutusunda listelenmiştir. Kullanıcı evrimsel algoritma yaklaşımında topluma uygulamak istediği operatörü belirler ve sonrasında *Methods* liste kutusunda listelenen operatöre ait metotlardan birini seçer. Bu işlemi istediği sırada ve istediği sayıda tekrarlayabilir. Her seçim sonucu seçilen operatör ve metodu *Operation Order* adlı liste kutusunda sırayla belirir. Operatör ve metotlar liste kutusunda belirdikten sonra kullanıcı gerekli satıra gidip üzerinde farenin sağ tuşuna tıklayarak metotlara gerekli parametreleri aktarabilir. Kullanıcı önceden seçmiş olduğu bir operatörü isterse bu listeden çıkarabilir. Bu bölümün altında hayatta kalış seçimi ve sonlanma kriteri operatörlerinin metotlarını bulunduran bir bölüm vardır. Kullanıcının seçtiği metot liste kutusunun altındaki

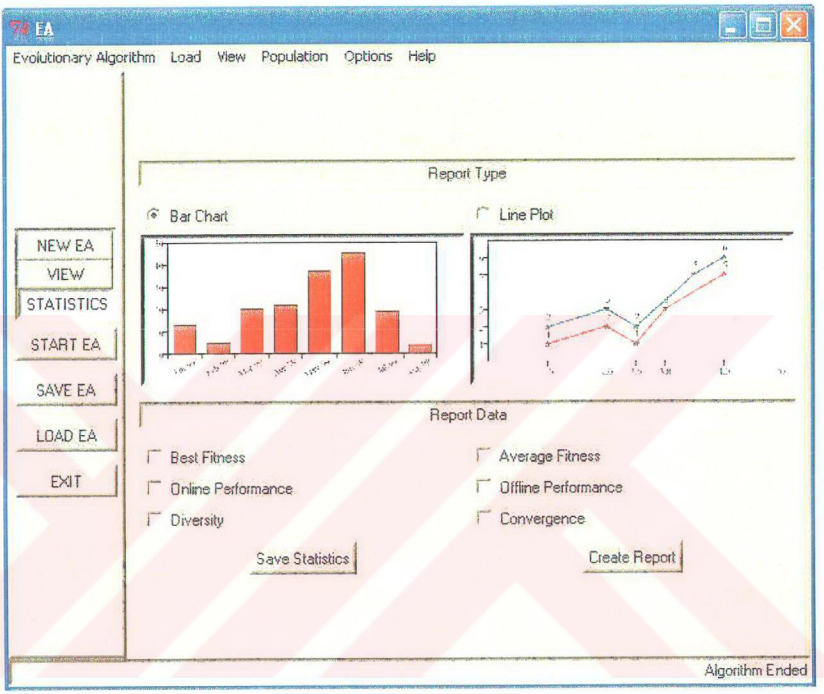
satırda görünecektir. Bu satırlarda farenin sağ tuşuna tıklama yapıldığında metotlara gerekli parametreler aktarılabilmektedir. Varolan operatörlerin yanısıra kullanıcının tanımladığı kendi operatörünü yükleyebilmesi için *Custom* seçeneği de *Operators* liste kutusunda listelenmiştir. Ayrıca kullanıcı varolan operatörlerin metotlarına yine *Custom* seçeneklerini kullanarak yeni metotlar ekleyebilir. Penceredeki *Information* metin kutusu örnek problemlere ilişkin bilgi vermesi amacıyla eklenmiştir. Metin kutusunun altında bulunan *Run Count* düğmesi ise evrimsel algoritmanın kullanıcının istediği sayıda ard arda koşturulmasını sağlar. Aksi belirtilmedikçe evrimsel algoritma bir kez çalışacaktır.



Şekil 4.2 EA arayüzünün görünüm penceresi

Şekil 4.2'de EA arayüzünün görünüm (*VIEW*) penceresi görülmektedir. Pencere evrimsel algoritma donduğunda ya da sonlandığında açılabilmektedir. Pencere ve özellikleri ile ilgili detaylı bilgi Bölüm 4.3.2'de anlatılmıştır.

Şekil 4.3'te EA arayüzünün istatistik (*STATISTICS*) penceresi görülmektedir. Pencere evrimsel algoritma donduğunda ya da sonlandıktan sonra açılabilmektedir. Pencere ve özellikleri ile ilgili detaylı bilgi Bölüm 4.3.3'te anlatılmıştır.



Şekil 4.3 EA arayüzünün istatistik penceresi

4.3 Seçenek ve Özellikler

4.3.1 İzleme Seçenekleri

İzleme seçeneği kullanılmasının amacı evrimsel algoritmanın kullanıcının seçimine göre belirli zamanlarda dondurulup sonra kaldığı yerden devam etmesini sağlamaktadır. İzleme seçeneği başlangıçta kapalıdır. Diğer bir deyişle evrimsel algoritma sonlanıncaya dek durmaksızın çalışır. Özellikle eğitim amaçlı kullanımlar için düşünülmüş olan izleme seçeneği evrimsel algoritma toplum üzerinde gerçekleştirilen her operatörden sonra ya da belirlenen sayıda nesil aralıklarında donacak şekilde ayarlanabilir. İşlemsel izleme seçeneğinin seçilmesi durumunda

evrimsel algoritma toplum üzerinde gerçekleştirilen her işlem ya da operatör sonunda donar. Kullanıcıya algoritmanın dondurulduğuna dair bilgi verilir. Algoritmanın devam etmesi kullanıcının isteğine bağlıdır, kullanıcı devam tuşuna bastığında algoritma kaldığı yerden devam eder ve bir sonraki işlemde tekrar donar. Nesilsel izleme seçeneğinin seçilmesi durumunda ise kullanıcı bir nesil sayısı belirtir. Evrimsel algoritma toplumun nesil sayısı kullanıcı tarafından belirlenmiş nesil sayısına ulaşınca donar ve devam etmek için kullanıcıyı bekler. Kullanıcının devam tuşuna basmasıyla kaldığı yerden devam eden evrimsel algoritma, toplum belirlenmiş nesil sayısı kadar nesil ilerledikten sonra tekrar donar. Evrimsel algoritma donduğunda kullanıcı toplumdaki bireylerin, varsa eşleşme havuzundaki ebeveynlerin ve oluşmuşsa çocukların genotiplerini inceleme şansı bulur. Bunun yanısıra kaydedildikten sonra istatistiksel verilerin incelenmesi de mümkündür.

4.3.2 Görünüm Özelliği

Görünüm, evrimsel algoritma sonlandığında ya da donduğunda kullanılabilir. Görünüm penceresinde toplumun bireylerinin, eşleşme havuzundaki ebeveynlerin ve çocukların genotiplerinin incelenebileceği liste kutuları bulunmaktadır. Ayrıca algoritma donduğunda veya sonlandığında toplumda bulunan en iyi ve en kötü bireyin, toplumdaki en büyük, en küçük ve ortalama uygunluk derecesinin, toplumdaki farklı birey sayısı ve toplumdaki yakınsamış birey sayısının incelenmesine olanak tanır. Görünüm penceresinde bulunan *EA Parameters* düğmesine tıklanarak toplumdaki birey sayısı, bir bireyin kromozom sayısı, kromozomlardaki gen sayısı ve tipi, topluma uygulanan operatörlerin sırası ve metotları ve toplumun sözlüğündeki veriler gibi evrimsel algoritmaya ait parametreler görülmektedir. *EA Statistics* düğmesi ile ise algoritma çalıştığı sürece bulunmuş en iyi birey ve algoritma donduğunda ya da sonlandığında varolan nesile ait istatistiksel veriler incelenebilmektedir.

4.3.3 İstatistik Özelliği

İstatistik penceresi evrimsel algoritma sonlandıktan sonra algoritmaya ilişkin çeşitli istatistiksel veriler kullanılarak bazı işlemler gerçekleştirilmesini sağlar. Bu işlemlerden biri kullanıcının seçeceği istatistiksel verileri kullanarak bir pdf dosyasına her nesil için seçilen verilerin değişimini gösteren bir grafik çizmektir. Grafikler kullanıcının tercihine göre çubuk grafiği ya da çizgisel grafik şeklinde

olabilir. Seçilebilecek veriler her nesil için en iyi uygunluk dereceleri, ortalama uygunluk dereceleri, çevrim dışı performans, çevrim içi performans, toplumdaki farklı birey sayıları ve toplumdaki yakınsamış birey sayılarıdır. İstatistiksel verilerden bir ya da birkaçı seçilerek verilere ilişkin grafikler oluşturulabileceği gibi her nesil için tüm istatistiksel verileri içeren bir text dosyası da oluşturulup kaydedilebilir. Kullanıcı isterse her nesil için toplumun bireylerinin, eşleşme havuzundaki ebeveynlerin ve/veya çocukların genotiplerinin de istatistiksel veriler ile birlikte kaydedilmesini sağlayabilir.

4.4 Eğitim Amaçlı Kullanımlar

En basit şekliyle eğitim amaçlı kullanımlar program kodu yazmaya gerek kalmadan birkaç fare tıklaması ile gerçekleştirilebilir. Bu kullanım şeklinde kullanıcı bir maksimizasyonu, 0/1 çanta, multiple çanta veya De Jong fonksiyonları ile gösterilen optimizasyon problemlerinden birisini seçer. Kullanıcı seçilen test problemi için önceden belirlenmiş değerleri kullanabileceği gibi bu değerleri kendisi de belirleyebilir. Ayrıca kullanıcı seçtiği test probleminin çözümünü basit genetik algoritma ile arayabileceği gibi, istediği operatör ve metodunu istediği sırada seçerek ve yine istediği hayatta kalış seçimi ve sonlanma kriteri metotlarını seçerek kendi evrimsel algoritma yaklaşımını koşturabilir. Eğitim amaçlı kullanımlarda izleme seçeneği önem taşır. Kullanıcı istediği izleme seçeneğini seçerek çalışan algoritmanın seçilen izleme seçeneğine göre istenilen zamanlarda donmasını sağlar. Böylece evrimsel algoritma donduğunda görünüm penceresi kullanılarak toplumun bireylerinde, ebeveynlerde veya çocuklarda meydana gelen değişimler ve topluma ait istatistiksel veriler incelenebilir. Ayrıca kullanıcı çeşitli parametrelerin algoritma üzerindeki etkisini de inceleyebilir. Örneğin mutasyon ve/veya çaprazlaşma olasılıklarını, toplumdaki birey sayısını, kromozomdaki gen sayısını v.b. değiştirerek bulunan istatistiksel sonuçları kaydedip diğer sonuçlar ile karşılaştırabilir.

4.5 Uygulama Amaçlı Kullanımlar

Evrimsel algoritmalar kullanılarak çözümü bulunmak istenen problemlerin çözümlerinin aranması EA arayüzünün başka bir kullanım amacına örnek olarak gösterilebilir. Eğer çözümü aranan problem arayüzde varolan test problemlerinden biri ile özdeşleştirilemeyen yeni bir problem ise kullanıcı kendi test problemi için bir

uygunluk fonksiyonu kodu yazar. Fonksiyon toplumdaki bireyler için bir uygunluk derecesi hesaplamalıdır. Bunun yanısıra kullanıcı mutasyon ve çaprazlaşma olasılıkları, toplumu oluşturacak birey sayısı, bireylerin taşıyacağı kromozom sayısı, kromozomları oluşturacak genlerin sayısı ve tipi, problemin minimum çözümü bulma problemi mi yoksa maksimum çözümü bulma problemi mi olduğu gibi bilgileri de sisteme girmelidir. Kullanıcı tarafından değiştirilmediği sürece toplumun mutasyon olasılığı 0.001 ve çaprazlaşma olasılığı 0.8 olur, toplum ise [1-10] aralığında tamsayı değerler alacak tamsayı tipten 10 gene sahip tek kromozom taşıyan 10 adet bireyden oluşur. Kullanıcı kendi fonksiyonunu arayüz vasıtasıyla yükler, istediği sırayla bireylere uygulanacak operatör ve metodunu seçer, hayatta kalış seçimi ve sonlanma kriteri operatörleri için de istenen metotlar seçilip gerekli parametreler verildikten sonra evrimsel algoritma koşturulabilir. Algoritma sonlandıktan sonra kullanıcı istatistik penceresi vasıtasıyla algoritma süresince hesaplanmış olan istatistiksel verilere ait grafikleri inceleyebilir. Ayrıca kullanıcı istatistik penceresindeki *Save Statistics* butonuna tıklayarak her nesile ait istatistiksel verilerin oluşturulacak bir dosyada kaydedilmesini sağlayabilir. Bu durumda dosyada algoritma süresince bulunmuş en iyi çözüm adayı da bulunacaktır. Kullanıcı algoritmayı istediği sayıda koşturarak sonuçları kıyaslayabilir.

4.6 Geliştirme / Akademik Amaçlı Kullanımlar

EA arayüzünün başka bir kullanım şekli kendi operatörünü geliştirmiş kullanıcılarca gerçekleştirilebilir. Kullanıcı geliştirdiği operatörünü seçeceği bir problemin evrimsel algoritmalar ile çözümünde test edebilir. Bunun için kullanıcı öncelikle bir test problemini seçmelidir. Kullanıcı geliştirdiği operatörüne ait metodun kodunu EA arayüzü vasıtası ile yükler ve toplumun bireylerine uygulanmak üzere seçer. Kullanıcı isterse bireylere uygulanmak üzere başka operatör(ler) ve metot(lar) da seçebilir. Probleme ve operatörlere ilişkin parametreler belirlendikten sonra evrimsel algoritma koşturulabilir. Kullanıcı istediği izleme seçeneğini seçerek görünüm penceresi vasıtasıyla adım adım işlemleri ve değişimleri inceleyebileceği gibi algoritma sonlandıktan sonra istatistik penceresinden faydalanarak toplumdaki her nesil için hesaplanmış istatistiksel verileri grafik üzerinde inceleyebilir ve bir dosyaya kaydedebilir. Kullanıcı evrimsel algoritmayı birden çok kez koşturup her çalışma sonunda kaydedilen istatistiksel verileri birbirleri ile karşılaştırabilir.

4.7 Örnek Bir Uygulama : DomGA

PyLEA paketindeki çaprazlaşma operatörleri toplumun tek kromozoma sahip bireylerden oluştuğu düşünülerek tasarlanmıştır. Bu sebeple birden fazla kromozoma sahip bireylerden oluşan bir toplumda seçim ve mutasyon operatörlerinin metotları bireyler üzerinde sorunsuz bir şekilde uygulanabilirken çaprazlaşma metotlarının uygulanması hata oluşumuna sebep olacaktır. Ancak bu PyLEA paketi ile geliştirilecek evrimsel algoritma yaklaşımları için yaratılacak toplumun tek kromozomlu bireylerden oluşması gerektiği anlamını taşımaz. Toplumdaki bireylerin birden fazla sayıda kromozoma sahip olması istendiğinde kullanıcı bu bireyler için geliştireceği çaprazlaşma operatörünü arayüzü kullanarak yükleyerek kendi evrimsel algoritma yaklaşımını oluşturabilir.

PyLEA paketindeki *Custom* adlı dizinde, geliştirilmiş bir diploid yapı uygulamasının gerçeklenmesi için DomGA adı verilen örnek oluşturulmuştur. Yapıda her birey ikili tipte genlerden oluşan iki kromozoma sahiptir. Bu kromozomlar bireyin genotipleridir. Kromozomlar üzerindeki bazı genler baskın iken bazı genler çekinik olmaktadır. Bireyin kromozomlarının genleri arasındaki etkileşim sonucu bireyin dış dünyada görünen fiziksel özellikleri, yani fenotipi meydana gelmektedir. Bireylerin uygunluk dereceleri ise fenotiplerine göre hesaplanmaktadır [18]. DomGA dosyasında bireylerin uygunluk derecelerini hesaplayacak olan *evaluation* adlı uygunluk fonksiyonu ve bireylerin tek noktadan çaprazlaşmasını sağlayacak *onePoint* metodu bulunmaktadır. Paketteki bir örnek problemin bu diploid yapı ile çözümünü araştırmak isteyen bir kullanıcı EA arayüzü üzerinde aşağıdaki adımları gerçeklemelidir :

1. DomGA içinde tanımlı uygunluk fonksiyonu (*evaluation*) EA arayüzü vasıtasıyla yüklenir, fonksiyona gerekli parametreler aktarılır ve evrimsel algoritmaya ilişkin parametreler belirtilir (toplumdaki birey sayısı v.b.),

2. Bireylere uygulanmak üzere EA arayüzünde listelenmiş seçim operatörü metotlarından biri seçilir,

3. DomGA içinde geliştirilmiş çaprazlaşma operatörü metodu (*onePoint*) arayüz vasıtasıyla yüklenir, bireylere uygulanmak üzere seçilir ve metoda gerekli parametreler aktarılır,

4. Bireylere uygulanmak üzere EA arayüzündeki ikili mutasyon metodu seçilir,

5. Arayüzde listelenmiş hayatta kalış metotlarından birisi seçilir,

6. Arayüzde listelenmiş sonlanma kriteri metotlarından birisi seçilir ve gerekli parametre aktarılır,

Bu adımlardan sonra EA arayüzü üzerindeki *START EA* butonuna tıklanması ile evrimsel algoritma koşturulur. Algoritma sonlandıktan sonra sonuçlar ve istatistiksel veriler incelenebilir.

5. SONUÇLAR

Tezde evrimsel algoritmalar için uygulama ve geliştirme yazılım altyapısı olan PyLEA paketi ve EA arayüzü anlatılmıştır. PyLEA paketi ve EA arayüzü Python dili kullanılarak hazırlanmıştır. PyLEA paketi öğrenimi ve kullanımı kolay olan esnek bir yapı olarak geliştirilmiştir. PyLEA platformdan bağımsızdır. Paket, kullanıcının kendi evrimsel algoritma yaklaşımını hızlı ve kolay bir biçimde geliştirebilmesini sağlamaktadır. Python dilinin yorumlanan bir dil olması nedeniyle şu an için yapı ile geliştirilen evrimsel algoritmalar yürütme hızı konusunda dezavantajlı olmaktadır. Hızın arttırılması için düşünülen bazı çözümler yapılabilecekler bölümünde anlatılmıştır. Pakette kullanıcının evrimsel algoritma yaklaşımını oluşturmasını kolaylaştıracak bazı örnek problemler, çeşitli operatörler ve metotları hazır olarak bulunmaktadır. Kullanıcı pakette tanımlanmış örnek problemlerden faydalanabileceği gibi kendi optimizasyon problemini de tanımlayabilir, benzer şekilde kullanıcı paketteki hazır operatörler ile birlikte kendi geliştirdiği operatörleri de yükleyip evrimsel algoritma yaklaşımını oluşturmada kullanabilir. Bunların yanısıra kullanıcı pakette gerçeklenmiş olan gen, kromozom, birey, toplum gibi sınıfları kullanarak oluşturacağı nesnelere başka nesneler ilave ederek yapıyı genişletebilir ve bu sınıfların metotlarına tanımlayacağı yeni metotları dahil edebilir. PyLEA paketi kullanılarak geliştirilmiş olan EA arayüzü programlama bilgisi, hatta çok fazla evrimsel hesaplama bilgisi dahi gerektirmeden kullanılabilmektedir. EA arayüzünden eğitim, uygulama ve geliştirme gibi çok amaçlı kullanımlarda faydalanılabilir. EA arayüzü sayesinde kullanıcının PyLEA paket yapısından faydalanarak kendi evrimsel algoritma yaklaşımını oluşturabilmesi oldukça kolaylaşmıştır.

6. YAPILABİLECEKLER

PyLEA paketi ile ilgili olarak üzerinde durulan en önemli dezavantaj yapının yorumlanan bir dil olan Python dili kullanılarak gerçekleştirilmiş olması nedeniyle hızının düşük olmasıdır. Gelecekte gerçekleştirilmesi düşünülen ilk çalışma yapının hızının arttırılması için Psyco kütüphanesi kullanımı ve bazı bölümlerinin C kodu ile yazılmasıdır.

PyLEA paketi için Python dili ve Tk kütüphanesi kullanılarak EA arayüzü geliştirilmiştir. Kullanıcı paket yapısını kendi bilgisayarına kurarak arayüzü çalıştırabilir. Yani kullanıcının oluşturacağı evrimsel algoritma kendi bilgisayarında koşturulur. Gelecekteki çalışmalarda PyLEA paket yapısı için internet tabanlı bir arayüz düşünülmektedir. Kullanıcı evrimsel algoritmasını bu arayüz sayesinde oluşturacak ve algoritma uzaktaki bir bilgisayarda koşturulacaktır. Kullanıcı evrimsel algoritmasını belirli aralıklarla koşturabilecektir. Örneğin 500 nesil boyunca sürecek bir algoritmayı 100'er nesillik aralıklarla koşturabilecek, her 100 nesilin sonunda algoritmayı kaydedip istediği zaman tekrar yükleyerek algoritmanın kaldığı yerden devam etmesini sağlayabilecektir.

PyLEA paket yapısında kullanıcı pakette tanımlanmış olan problemlerin dışında bir problemin çözümünü aramak istiyorsa Python dili ile kod yazarak problemi tanıtması gerekmekteydi. Gelecekteki çalışmalarda problem tanıtımı için XML tabanlı bir dil geliştirilmesi ve oluşturulacak yeni yöntemle kullanıcının bu biçimde kod yazmasına gerek kalmadan problemi tanıtabilmesi düşünülmektedir.

PyLEA paket yapısında evrimsel algoritma sürdükçe oluşan her nesil için istatistiksel veriler bir dosyada saklanır. Kullanıcı EA arayüzünden faydalanarak bu istatistikleri grafikler üzerinde inceleyebilir ya da kendi dosyasına kaydedebilir. Ayrıca kullanıcı evrimsel algoritmayı birden fazla kez koşturarak oluşan dosyalar vasıtasıyla istatistiksel sonuçları karşılaştırabilir. Gelecekteki çalışmalarda kullanıcının evrimsel algoritmayı birden fazla kez koşturması durumunda her adıma ait istatistiksel verileri karşılaştıran grafiklerin oluşturulabilmesi düşünülmüştür.

KAYNAKLAR

[1] **Curtis, H. and Barnes,** N. S., 1994. Invitation to Biology, Fifth Edition, Worth Publishers, New York.

[2] **Eiben, A. E. and Smith, J. E.,** 2003. Introduction to Evolutionary Computing, Springer-Verlag, Heidelberg.

[3] **Michalewicz, Z.,** 1999. Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Heidelberg.

[4] **Uyar, Ş.,** 2003. Evolutionary computation lecture notes, www.cs.itu.edu.tr/%7Eetaner/courses/FB_EvoComp03_04/index.html

[5] **GEATbx**, Genetic and Evolutionary Algorithm Toolbox for use with Matlab, www.geatbx.com/docu/algindex.html.

[6] **Khuri, S., Bäck, T. and Heitkötter, J.,** 1994. The zero/one multiple knapsack problem and genetic algorithms, *Proceedings of the 1994 ACM symposium on Applied computing*, Phoenix, Arizona, United States, 1994, 188-193.

[7] **Digalakis, J. G. and Margaritis, K. G.,** 2002. An experimental Study of benchmarking functions for genetic algorithms, *Intern. J. Computer Math.*,**79(4)**, 403-416.

[8] **MP-TESTDATA**, Description of data format for 0/1-Multiple Knapsack Problems, http://elib.zib.de/pub/Packages/mp-testdata/ip/sac94-suite/.

[9] **Tüfekçioğlu, F. and Orbayı, M.,** 2002. A framework for applying genetic algorithms as a web service, *The Genetic and Evolutionary Computation Conference* , New York City, New York, USA, July 9-13.

[10] **Louis, S. and Rawlins, G.,** 1992. Syntactic Analysis of Convergence in Genetic Algorithms, *Foundations of Genetic Algorithms* , Vail, Colorado, USA, July 26-29.

[11] **Grefenstette, J. J.,** 1990. A User's Guide to GENESIS Version 5.0, http://www.genetic-programming.com/c2003genesisgrefenstette.txt.

[12] **Baeck, T.,** 1992. A User's Guide to GENEsYs 1.0, Department of Computer Science, University of Dortmund.

[13] **GPC++ site**, www.cs.ucl.ac.uk/staff/W.Langdon/ftp/weinbenner/gp.html.

[14] **EO site**, http://eodev.sourceforge.net/.

[15] **Open Beagle site**, www.gel.ulaval.ca/~beagle/.

[16] **Gagne, C. and Parizeau, M.**, 2002. Open Beagle : A new versatile C++ framework for evolutionary computations, *The Genetic and Evolutionary Computation Conference* , New York City, New York, USA, July 9-13.

[17] **ECJ site,** http://cs.gmu.edu/~eclab/projects/ecj/.

[18] **Uyar, Ş. and Harmancı, A. E.**, 2004. Comparison of Domination Approaches for Diploid Binary Genetic Algorithms, Applications and Science in Soft Computing, Advances in Soft Computing series, pp. 75-80, Springer-Verlag, Heidelberg.

[19] **Python official site,** www.python.org.

[20] **Downey, A. B., Elkner, J. and Meyers, C.,** How to Think Like a Computer Scientist: Learning with Python, www.ibiblio.org/obp/thinkCSpy/.

[21] **Rossum, G.**, Python Tutorial, www.python.org/doc/current/tut/tut.html.

[22] **Rossum, G.**, Python Library Reference, www.python.org/doc/current/lib/lib.html.

[23] **Beazley, D. M.**, 2001. Python Essential Reference, Second Edition, New Riders Publishing, USA.

[24] **Reportlab official site**, www.reportlab.org.

[25] **Lundh, F.,** An Introduction to Tkinter, www.pythonware.com/library/tkinter/introduction/.

[26] **New Mexico Tech Computer Center,** Tkinter reference: A GUI for Python, http://infohost.nmt.edu/tcc/help/pubs/tkinter.pdf.

EK A

PYTHON

Python yüksek seviyeli, yorumlanan ve nesneye dayalı bir programlama dilidir. Python yorumlanan bir dil olduğundan hızı düşüktür. Bunun yanısıra Python'un sağladığı pek çok avantaj vardır. Öncelikle Python Linux, Windows ve Macintosh gibi farklı platformlarda çalıştırılabilen, platformdan bağımsız bir dildir. Python açık kodlu bir dildir. İnternet üzerinden ücretsiz indirilip dilin kolayca öğrenilebileceği pek çok kaynak mevcuttur. Dilin programlama bilmeyen kişilerce bile öğrenilmesi kolay ve hızlıdır. C diline benzediğinden C dilini bilenler için öğrenimi ve kullanımı çok daha kolaydır. Ayrıca Python ile uygulamalar diğer dillere kıyasla daha hızlı ve kolay bir biçimde geliştirilebilmektedir. Python diğer dillerde olmayan gelişmiş bazı veri yapılarına sahiptir. Python'ın yorumlanan bir dil olması kullanıcıyı derle – koştur – tekrar derle döngüsünden kurtarmaktadır. Python sentaksı basit olan bir dildir. Bu dilde değişken veya argümanların önceden bildirimine gerek kalmamaktadır. Python ile geliştirilmiş programlar başka Python programlarınca da kullanılabilecek şekilde modüllere bölünebilir. Bunların yanısıra programların, C kodu ile yazılmış modüllerin yorumlayıcıya eklenmesi ile hızlandırılmasına veya genişletilmesine imkan tanınır.

Python yorumlayıcısı iki farklı kipte (modda) çalıştırılabilir. Komut satırı kipi veya etkileşimli kipte yorumlayıcı bir hesap makinası olarak çalıştırılabilir. Satıra komut girilir ve yorumlayıcının komutu yorumlaması sağlanır. Ayrıca bu kipte fonksiyonlar veya sınıflar da oluşturulabilir. Diğer kip komut dosyası kipidir. Bu kipte program kodu .py uzantılı bir dosyaya yazılır (komut dosyası), ve yorumlayıcının dosyanın içeriğini yorumlaması sağlanır.

Python dilinde diğer dillerde bulunmayan bazı özellikler mevcuttur. Örneğin Python'da bir şeye referans olabilen bütün değişkenler bir nesne olarak kabul edilir. Ayrıca Python'da diğer dillerde bulunmayan bazı veri yapıları mevcuttur. Bunlardan birisi liste yapısıdır. Liste yapısı C'deki bağlantılı liste yapısına benzer. Listenin elemanları farklı tiplerden olabilir. Örneğin bir liste başka bir liste, bir çoklu (tuple), bir sözlük, katar, tamsayı, kayan noktalı sayı v.b. tipten verileri bir arada

bulundurabilir. Listeler sabit değildir. Listeye eleman eklenip çıkarılabilir, ya da elemanlar değiştirilebilir. Diğer bir veri yapısı çokludur (tuple). Çoklular virgüllerle ayırılmış değerler listesi olarak düşünülebilir. Kenarlarında parantezler ile gösterilir. Liste yapısında olduğu gibi çoklularda da farklı tipten veriler bulunabilir. Ancak liste yapısından farklı olarak çoklular sabittir. Bir çokluya eleman eklenemez ya da çıkartılamaz, varolan elemanlar değiştirilemez. Ayrıca Python'da sözlük yapısı vardır. Sözlük yapısı süslü parantezler ile gösterilen bir liste gibi düşünülebilir ancak listenin her elemanı bir anahtar ile bir değer çiftinden oluşur. Sözlük yapısı çoklular gibi sabit değildir. Yeni anahtar-değer çiftleri eklenebilir ya da bir anahtarın değeri değiştirilebilir. Sözlükler içindeki değerlere değerin anahtarı kullanarak ulaşılır. Şekil A.1 bahsedilen üç veri yapısını göstermektedir. Şekilde veri yapısına göre isimlendirilmiş bir liste, bir çoklu ve bir sözlük oluşturulmuştur.

```
Liste   = [ "merhaba", 2.0, 5, [10, 20] ]
Çoklu   = ( 'a', 'b', 'c', 'd', 'e' )
Sözlük  = { 'x': 525, 'y': 430, 'z': 217 }
```

Şekil A.1 Pythondaki veri yapıları örnekleri

Python dilinde for döngüleri C dilinden daha farklı işler. For döngüsünde liste, çoklu ya da sözlük yapıları kullanılabilir. Döngünün her adımında veri yapısının elemanları arasında ilerlenir.Örneğin Şekil A.2'deki döngüde tek tek Liste adlı liste yapısının elemanları ekrana yazdırılır.

```
Liste   = [ "m", "e", "r", "h", "a", "b", "a"]
for eleman in Liste :
    print eleman
```

Şekil A.2 For döngüsü örneği

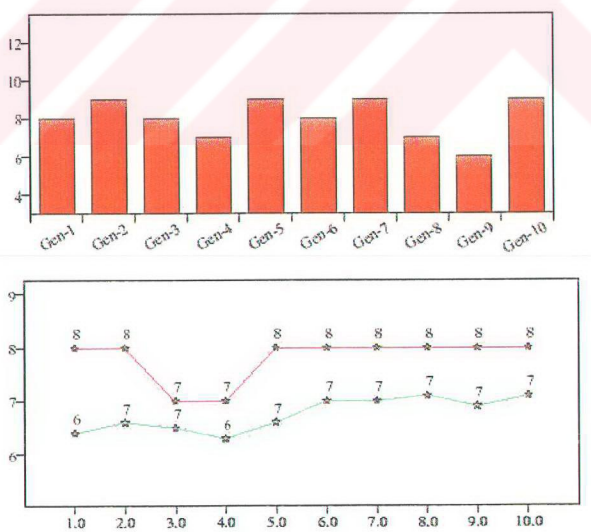
Ekte Python diline ilişkin genel bilgiler anlatılmıştır. Daha detaylı bilgi [19-23] kaynaklarından edinilebilir.

EK B

REPORTLAB KÜTÜPHANESİ

Reportlab, kullanıcıların Python dili ile gerçeklenmiş grafik komutlarını kullanarak Adobe PDF dökümanları yaratmasını sağlayan açık kodlu bir yazılım kütüphanesidir. Reportlab ile raporlar diğer rapor hazırlama araçlarına kıyasla daha hızlı oluşturulabilmektedir [24].

Reportlab kütüphanesinde raporların hızlı çizilmesini sağlayan ve sahip olduğu grafik özellikleri ile raporların geliştirilmesine imkan tanıyan Reportlab Graphics alt paketi bulunmaktadır. Tez çalışmasında istatistiksel verilere ait grafiklerin oluşturulmasında Reportlab Graphics alt paketinden faydalanılmıştır. Grafikler kullanıcının tercihine göre çubuk grafiği ve çizgisel grafik olmak üzere iki farklı biçimde oluşturulabilir. Şekil B.1'de EA arayüzü kullanılarak Adobe PDF dosyalarına çizdirilmiş çubuk grafiğine (üstte) ve çizgisel grafiğe (altta) örnek iki grafik gösterilmiştir.



Şekil B.1 EA arayüzü ile oluşturulmuş çubuk grafiği ve çizgisel grafik örnekleri

PyLEA paket yapısı ve EA arayüzünü kullanarak kullanıcı evrimsel algoritma süresince toplumda oluşan her nesil için :

- toplumdaki en iyi bireyin uygunluk derecesi,
- toplumun ortalama uygunluk derecesi,
- çevrim içi performans,
- çevrim dışı performans,
- toplumdaki yakınsamış birey sayısı,
- toplumdaki farklı birey sayısı,
- toplumdaki bireylerin birbirlerine olan Hamming uzaklıklarının ortalaması

gibi istatistiksel verilere ait grafikleri evrimsel algoritma sonlandıktan sonra istediği grafik tipi üzerinde inceleyebilir. Kullanıcıya grafik üzerinde incelemek istediği verileri seçme imkanı tanınmıştır.

Ekte Reportlab kütüphanesi ile ilgili genel bilgiler verilmiş ve tez çalışmasında kullanılan Reportlab Graphics alt paketi tanıtılarak oluşturulacak grafiklerin içerebileceği veriler belirtilmiştir. Reportlab kütüphanesi hakkında detaylı bilgi www.reportlab.org adresinden edinilebilir.

EK C

TK KÜTÜPHANESİ

Tk uzun zamandır Python'un bir parçası olmuştur. Tk kütüphanesi Python programcılarına varolan Tkinter modülü ve onun uzantısı olan Tix modülü sayesinde platformdan bağımsız bir pencereleme aracı sunmaktadır. Tk ve Tkinter modülü Unix, Windows ve Macintosh sistemleri için uygundur. Çalışmada EA arayüzünü gerçekleştirirken Tk kütüphanesinin Tkinter modülünden faydalanılmıştır.

Tk Python için tek grafiksel kullanıcı arayüzü (GUI) olmasa da en sık kullanılanıdır. Tk kütüphanesi ile ilgili daha detaylı bilgi www.python.org adresinden ve [25, 26] kaynaklarından edinilebilir.

ÖZGEÇMİŞ

Fulya DEMİRKIRAN, 1980 yılında İstanbul' da doğdu. Ahmet Şimşek Kolejinden sonra lise eğitimini Üsküdar Fen Lisesinde tamamladı. 1997 yılında İstanbul Teknik Üniversitesi Elektonik ve Haberleşme Mühendisliği bölümünde lisans eğitimine başladı. 2001 yılında bu bölümden mezun olduktan sonra yüksek lisans eğitimini İstanbul Teknik Üniversitesi Bilgisayar Mühendisliği bölümünde sürdürdü. Halen bu bölümde yüksek lisans eğitimine devam etmektedir.