İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

TERMİK SANTRALLERDE ÜNİTE PROGRAMLAMASI PROBLEMİNİN EVRİMSEL STRATEJİLER ALGORİTMASI İLE ÇÖZÜMLENMESİ

Bitirme Ödevi

Meryem Uzun
040030023

Bölüm : Bilgisayar Mühendisliği

Danışman : Yrd.Doç.Dr.Şima Etaner-Uyar

Mayıs 2007

İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

TERMİK SANTRALLERDE ÜNİTE PROGRAMLAMASI PROBLEMİNİN EVRİMSEL STRATEJİLER ALGORİTMASI İLE ÇÖZÜMLENMESİ

Bitirme Ödevi

Meryem Uzun
040030023

Bölüm : Bilgisayar Mühendisliği

Danışman : Yrd.Doç.Dr.Şima Etaner-Uyar

Mayıs 2007

Özgünlük Bildirisi

1. Bu çalışmada, başka kaynaklardan yapılan tüm alıntıların, ilgili kaynaklar referans gösterilerek açıkça belirtildiğini,
2. Alıntılar dışındaki bölümlerin, özellikle projenin ana konusunu oluşturan teorik çalışmaların ve yazılım/donanımın benim tarafımdan yapıldığını

bildiririm.

İstanbul, 23.05.2007

Meryem Uzun

TERMİK SANTRALLERDE ÜNİTE PROGRAMLAMASI PROBLEMİNİN EVRİMSEL STRATEJİLER ALGORİTMASI İLE ÇÖZÜMLENMESİ

(ÖZET)

Bu bitirme ödevinin amacı, termik santrallerdeki ünite programlamasının çözülmesidir. Ünite Programlaması Problemi (Unit Commitment Problem(UC Problemi)), elektrik enerjisi üretme birimlerinin açma kapama programlarını, toplam üretim maliyetini en aza indirgemek amacıyla, tüm kısıtları sağlarken, yük gereksinimlerini karşılayacak şekilde belirlemektir.

Açma kapama programları belirlenirken dikkat edilmesi gereken en önemli husus maliyet terimlerinin toplamını en aza indirmektir. Maliyet terimlerinden birincisi, çalışan jeneratörlerin çalışması sırasında ürettikleri enerjinin maliyeti; ikinicisi jeneratörlerin çalışmaya başlatılma maliyetleridir, ki bu jeneratörlerin o anki genel sıcaklıklarına bağımlıdır. UC Probleminde kısıtlar şu şekildedir; jeneratörlerin en fazla ve en az ne kadar güç üretebileceği, jenerator açılırsa en az kaç dakika açık kalması gerektiği, jenerator kapatılırsa en az kaç saat kapalı kalması gerektiği, bir jeneratörün tamamen soğuması için gerekli olan süre, jeneratör henüz soğumamışken tekrar açılırsa sebep olunan maliyet, jeneratör soğumuşken tekrar açılırsa sebep olunan maliyet ve program çalıştırılmaya başlandığında jeneratörlerin o ana kadar kaç saattir açık ya da kapalı olduğu .

Bu çalışmada problem Evrimsel Stratejiler Algoritmasıyla çözümlenmiştir. Evrimsel Stratejiler(ES) Algoritmasında belli sayıda bireyden oluşan rastgele bir toplum oluşturulur. Bu toplumdan çocuklar üretilir. Bu çocuklar rekombinasyon ve mutasyona uğratılır. Sonra Darwin'in kuramındaki ortama daha iyi adapte olanların hayatta kalması gibi çocuk bireylerden de çözüme daha uygun olarak görülen bireyler kalırken, diğerleri yok olur. Bu durum yeterince iyi bireyler üretilene kadar devam eder.

ES, UC Probleminde jeneratörlerin hangi saat diliminde açık, hangi saat diliminde kapalı olması gerektiği problemini çözer. Jeneratörlerin açık kalması gereken o saat dilimlerinde ne kadar enerji üretmeleri gerektiği Lambda İterasyonu ile çözülür.

Sonuç olarak, problem beklenen şekilde çözülmüş ve daha önce bu problemi çözen başka algoritmaların ürettiği sonuçlarla kıyaslanmıştır. Sonuçların oldukça iyi olduğu, verilen optimum değerlerle en kötü değerler arasında değerler bulunduğu görülmüştür.

SOLVING UNIT COMMITMENT PROBLEM WITH EVOLUTIONARY STRATEGIES

(SUMMARY)

This graduation project's aim is solving Unit Commitment (UC) Problem. Unit Commitment Problem, determining generator's open or close schedule to meet load demand, while satisfying all constraints. The total product cost must be as minimum as it could be.

Power doesn't used same for all hours in a day. Midnight and early hours in the morning power is used very few. So generators should be planned hour by hour for producing efficient power, while spending minimum money.

UC Problem generally can be divided two half. First, for all hour it should be determined that which generator is open, which generator is closed. Second, while generator is open, how much power it should produce. UC problem aims minimizing cost terms. The first cost term is fuel cost, the second term is start up cost.

Fuel cost is cost that is caused while generator is working. There are some contraints for that.

- While generators are working they should meet the power demand.
- All generators have minimum and maximum capacity to produce power.

This sub problem is called Economic Dispatch (EC) Problem. This problem can be formuled by Lagrange Multiplier and it can be solved by Lambda Iteration. This iteration aims calculating how much power generators need to produce to meet the power demand. Then by fuel cost formula cost is calculated for that hour. This iteration is repeated for all hours. The sum of all fuel costs for each hour gives the total fuel cost.

Start up cost is cost that is caused while opening the generators. It depends heat of generator. After closing a generator, it needs some hours to get cold. In this hours, if the generator is opened again it cause less cost. If the generator is opened after getting cold it causes much cost.

The aim of UC Problem is minimizing the fitness. Fitness is sum of total fuel cost, total start up cost and penalty. Penalty occur because of unmet constraints. We want a fitness value which doesn't have any penalty.

Penalty appears because of unmet power demand and unmet up/down time constraints. Unmet power demand and reserve causes a cost. This cost is calculated by multiplying the unmet power by a coefficient. The other penalty is, to run generators healthful they have to stay open for some hours after generator is opened and also stay close for some hours after generator is closed. So because of unmet up/down time constraints a penalty occurs. This penalty also depends the stage of evolution. It increases for each generation. And it also converted to cost by multiplying a coefficient.

UC problem is a complex optimization problem. This problem is solved by a lot of methods. These are; integer programming (IP), dynamic programming (DP), branch and bound, Benders' decomposition, Lagrangian relaxation (LR), simulated annealing (SA), tabu search and genetic algorithms (GA). In this project UC problem is solved by Evolutionary Strategies.

Evolutionary Algorithms are created by drawing inspiration from Charles Darwin's the theory of evolution. In Evolutionary Algorithms problems are solved by adapting the problem to living beings' evolution process. For Evolutionary Strategies fistly a random population is created. A population is represented by chromosomes. A chromosome is created with two parts. A half is for genes which is 0 or 1, the other half is for mutation probability which is real numbers. For first population all of these values are created randomly. Then from this population it is created λ children.

For creating a children a mother and a father individuals are selected from population randomly which are different from each other. All individuals have same probability to be chosen. After choosing parents, these individuals are combined each other by recombination. By recombination it can be created one or two child. In this project a child is created. Child's genes are created by discrete(dominant) recombination, mutation probabilities are created by intermediary recombination.

Discrete recombination is chosing a value which is from mother or father individuals. Intermediary recombination is calculating a value which is average of mother and father.

After recombination, children are changed by mutation. Children's genes are changed according to mutation probability. If children's a gene's mutation probability is greater than a random real number, the gene is converted to 0 if it is 1, 1 if it is 0.

According to Charles Darwin, the individuals which can adapt the environment stay and live, the others disappear. This is called natural selection. In ES the next generation is selected from children or children and individuals. If it is selected from both children and individuals the best individuals doesn't disappear but for next generations it seemed the generartions are nearly same with each other. So selecting the new generation from children is better. But this time the best individuals can disappear. Because of that elitism method could be applied. With Elitism the best individual from gained past generations, is added to next generation. So the best individual and its effects can continue for next generations. In this project next generation is selected from only children and for the worst individual elitism method is applied. So the best solution is always protected.

Herewith, the next generation is created. Then the new generation will create the next generation by repeating all steps. This repetations finish when the best individual is found which solves the problem. If it can't be decided that the individual is the best or not, it is repeated for constant value.

While solving UC problem with Evolutionary Strategies, the chromosome will show open or close schedule of generators. For solving the problem firstly the chromosome is converted to a matris whose rows show hours, columns shows generators. Then the fitness of matris is calculated. All chromosomes pass from same process. Then the best child whose fitness value is less is chosen to generate the next generation. ES algorithm is repeated for a constant number. This iteration number must be big as much as it could be

because, if ES is repeated very much for a population, then this population produces a better solution.

These iterations are also repeated for different populations. All population's best individual and its infos will saved. After all iterations, the best, the average and the worst of best individuals of populations are calculated. The best solution gives the solution. These informations are written in an output file. Also the power which generators need to produce for an hour will be written in the output file. Lastly if solution doesn't supply up/down time constraints, it will be arranged to supply up/down constraints.

In the code the individual number is 20, children number is 140. More children means better generations. The population number is 5 and ES process's iteration number is 3000. If iteration number increases it means we could find a better solution. But for testing the population number and iteration number can be changed.

In UC problem there are a lot of constraints. All constraint will be read from input file. The infos which are need to read from file are minimum and maximum power limit of generators which they can produce, coefficients which are needed while calculating the fuel cost, up/down time constraints for all generators, the time period which generators get cold, cold start cost, hot start cost, initial state of generators.

The power which is produced by generators could be acceptable if it supplies the power demand with 0.0001 tolerances. Lambda iteration finishes if power demand is supplied. But if the power demand could not be supplied, the program could run in infinity loop. To avoid infinity loop, this iteration will be limited with a constant number which is 1000 in this code.

For converting penalty to money some coefficients are also needed. For unmet power demand and reserve demand we multiply the unmet power with 200. For unmet up/down time constraints the product will multiplied by 100.

This project is solved with different algorithms. And for some test datas optimum value was found. This project ran with these datas and the solution is compared with the given solutions. The given solutions and the solutions which are gained are below:

For 4 generators and 8 hours test system, which is taken from Valenzuela and Smith's article [1] was solved by Genetic and Memetic Algorithm and its optimum solution was found. These solutions and the solution solved by Evolutionary Strategies are below:

Memetic Algorithm			Evolutionary Strategies		
Best	Average	Worst	Best	Average	Worst
74675	74959	75012	74676	74825	75008

The solutions are very similar to each other. Evolutionary Strategies's average cost is less than memetic algorithm's average cost.

The other test inputs are also from the same article. In this test 10 generators' 24 hours power demand is given. This problem was solved by Dynamic Programming (DP), Lagrangian relaxation (LR), Genetic Algorithms (GA) and Memetic Algorithms (MA). Also in this project it is solved by Evolutionary Strategies (ES).

MA			GA		
Best	Average	Worst	Best	Average	Worst
565827	566453	566861	565866	567329	571336

DP	LR			ES		
Optimum	Best	Average	Worst	Best	Average	Worst
565827	566107	566493	566817	566597	568211	569225

The solutions gained from ES is bad than the best solutions. But its worst solution is better than Genetic Algorithm's worst solution.

The other test system is from Turkiye's 8 generators' 8 hours power demand. This system's input datas will be given in the Experimental Solutions chapter of this report. The project ran for 100 populations. The gained best, worst and average solution is below:

best : 504502.9260378304
average : 504562.84181683103
worst : 504835.79147671885

If we look the output file of this project we saw all power demand can meet by produced power.

İÇİNDEKİLER

1 GİRİŞ

Ünite Programlaması Problemi (Unit Commitment Problem(UC Problemi)), elektrik enerjisi üretme birimlerinin açma kapama programlarını, toplam üretim maliyetini en aza indirgemek amacıyla, tüm kısıtları sağlarken, yük gereksinimlerini karşılayacak şekilde belirlemektir [1,2,3,4].

Elektrik enerjisi günün her saatinde aynı ölçüde kullanılmamaktadır. Gece ve sabahın ilk saatlerinde gündüze oranla daha az kullanılır [1,2]. Bunun için santrallerin ne kadar enerji üretmeleri gerektiği saat saat planlanmalıdır [1].

UC problemi genel olarak ikiye ayrılabilir. Birincisi, her saat diliminde hangi jeneratörlerin çalışacağını belirlemek; ikincisi, her jeneratörün üretmesi gereken ekonomik güç seviyesini belirlemektir (Economic Dispatch Problem(ECP)) [1,4]. UC Probleminde amaç maliyet terimlerinin toplamını en aza indirmektir. Maliyet terimlerinden birincisi, çalışan jeneratörlerin çalışması sırasında ürettikleri enerjinin maliyeti; ikinicisi jeneratörlerin çalışmaya başlatılma maliyetleridir, ki bu jeneratörlerin o anki genel sıcaklıklarına bağımlıdır. Belli başlı kısıtlar şu şekilde sıralanabilir; santralden istenen enerji miktarı, santralin açık ya da kapalı kalma süresinin sağlanması, üretilen enerjinin santralin üretebileceği enerji sınırının içinde kalması gibi [1].

UC problemi karmaşık bir matematiksel en-iyileme problemidir [3]. Bu problem çeşitli yöntemlerle çözülmüştür. Bunlardan birkaçı şu şekilde sıralanabilir; Tamsayı Programlama(IP), Dinamik Programlama, Dal-Sınır Algoritması, Lagrangian Çarpanları, Yasak Arama (Tabu Search), Benzetimli Tavlama (Simulated Annealing), Memetik Algoritma ve Genetik Algoritmalar [1].

Bu çalışmada problem Evrimsel Stratejiler Algoritmasıyla çözümlenecektir. Evrimsel algoritmalar Darwin'in evrim teorisinden etkilenerek üretilmiştir. Problemler canlıların doğasında var olan evrimleşme sürecine uyarlanarak çözümlenir. Tüm evrimsel algoritmalar altında yatan fikir aynıdır. Belli sayıda bireyden oluşan rastgele bir toplum oluşturulur. Bu toplumdan yeni bireyler üretilir. Bu bireyler rekombinasyon ve mutasyona uğratılır. Sonra Darwin'in kuramındaki ortama daha iyi adapte olanların hayatta kalması gibi çocuklardan da çözüme daha uygun olarak görülen bireyler kalırken, diğerleri yok olur. Bu durum yeterince iyi bireyler üretilene kadar devam eder [7]. Evrimsel Stratejiler Algoritması da aynen bu mantıkta çalışır.

Sonuç olarak, hangi jeneratörlerin hangi saat diliminde çalışacağı ve hangi saat diliminde ne kadar güç üretmeleri gerektiği hesaplanır. Problem Venezuela'nın makalesinde yer

verilen iki çözülmüş problem üzerinde test edilmiştir [1]. Makalede verilen optimum değerlere oldukça yakın sonuçlar elde edilmiştir.

Raporun ikinci kısımında, proje için hazırlanmış proje planı verilecektir. Proje planında, projenin amacı, kapsamı, projeye ilişkin büyüklük, zaman tahminleri, riskler ve bunların çözümlenmesi için öneriler, iş paketleri ve projenin zamanında bitmesi için öngörülen zaman programı verilecektir.

Raporun üçüncü kısmında Ünite Programlaması Probleminin ne olduğu, bu probşemin çözümüne ilişkin teorik bilgiler, Evrimsel Stratajiler Algoritmasının tarihçesi, geliştirimine ilişkin ayrıntılı bilgiler verilecektir.

Raporun dördüncü kısmında analiz ve modelleme yapılacak, projenin UML sınıf diyagramı verilecektir.

Raporun beşinci kısmında tasarım, gerçekleme ve test aşamalarında neler yapıldığı anlatılacaktır.

Raporun altıncı kısmında projenin test edilmesi sonucunda elde edilmiş olan sonuçlar, problemin daha önceden başka algoritmalarla çözülerek elde edilmiş sonuçlarıyla kıyaslanarak verilecektir.

Raporun yedinci kısmında sonuç ve öneriler yer alır.

Raporun sekizinci kısmında kullanılan kaynaklar verilmiştir.

2 PROJENİN TANIMI VE PLANI

1. Projenin amacı

Projede amaç termik santrallerde ünite programlaması probleminin çözümlenmesidir. Ünite Programlaması Problemi (Unit Commitment Problem(UCP)), elektik enerjisi üretme birimlerinin açma kapama programlarını, bir grup işlemsel kısıtı sağlayarak, yük gereksinimlerini karşılayacak şekilde, en düşük maliyetli olarak belirlemektir.

Ürün maliyeti yakıt, başlatma ve çalışmayı sürdürme maliyetlerini içerir. Elektrik enerjisi günün her saatinde aynı ölçüde kullanılmamaktadır. Elektrik akımı için bu gereksinim gün içerisinde fazla, gecenin ilerleyen saatleri ve sabahın ilk saatlerinde azdır. Bu periyodik gereksinim, şirketlerin ne kadar enerji üretmeleri gerektiğini saat saat planlamasını gerektirir. Problem öncelikle erişilebilir birimlerden hangisinin açık olacağına karar verme ve sonra birimlerin ekonomik yük dağılımı programını tanımlamaktır. Standart UCP'de amaç maliyet terimlerinin toplamını en aza indirmektir. Birinci terim üretim birimleri tarafından üretilen enerjinin yakıt harcamasına bağımlı maliyeti, ikinici terim ise üretim birimlerinin başlatılma maliyetidir (termal birimler kazanların genel sıcaklığına bağımlıdır). Kısıtlar şu şekilde sıralanabilir: kapasite rezervi, minimum açık/kapalı olma zamanı, iletim hatlarında maksimum enerji akışı ve işletme limitleri. [1]

2. Projenin kapsamı

UCP doğrusal olmayan hedef fonksiyonları, çok sayıda kısıtları ve çok büyük boyutları olması yönüyle çözümlemesi zor bir eniyileme problemidir. Bu problem 'Evrimsel Stratejiler Algoritması' kullanılarak çözümlenecektir. Evrimsel algoritmalar, rassal çeşitlendirmeye ve seçime dayalı populasyon temelli bir yaklaşımı içinde barındıran metotlar grubudur. Evrimsel Stratejiler Algoritmasında μ adet bireyden oluşan bir toplum üzerinde çalışılır. Baslangıç
toplumu rasgele olarak üretilir. Her bireyin sıfır ve birlerden oluşan bir vektör olan kromozomu bulunur. (Problemimizde '0' elektrik enerjisi üretim biriminin kapalı olduğunu, '1' açık olduğunu simgeler.) Her kromozom algoritmanın olası bir çözümünü temsil eder. μ toplumdaki bireylerin sayısını, yani ebeveyn sayısını simgeler. λ bu toplumdan üretilen çocukların sayısını temsil eder. λ adet çocuğun üretilmesi için, ebeveyn seçimi, rekombinasyon ve mutasyon adımları birbirini izleyecek sekilde λ kez tekrarlanır. Çocukların üretilmesinden sonra hayatta kalacakların seçimi gerçekleştirilir. Hayatta kalacakların seçilmesinden sonra, seçilen bireyler yeni toplumu meydana getirirler. Bütün bu adımlar, hedefe yeterince uygun olan bir birey bulunana kadar gerçekleştirilir. Aranılan birey bulunduktan sonra ekonomik yük paylaşımı fonksiyonları bu birey üzerinde uygulanır.

Evrimsel Stratejiler Algoritması asıl probleme uyarlanmadan önce sonucu bilinen örnek bir fonksiyon üzerinde test edilecektir. Doğruluğundan emin olunduktan sonra asıl probleme uyarlanacaktır. Daha sonra asıl problem için gerekli testler yapılacaktır.

3. Projeye ilişkin kestirimler

Proje bir dönem yani 4 ayda bitirilmelidir. Elektrik mühendisliğine ilişkin bilgiler ve problemimiz için gerekli olan gerçek veriler Doç. Dr. Belgin Türkay yardımıyla temin edilecektir. Proje Java programlama dili ile yazılacaktır.
Kodun kaç satır süreceğini tahmin etmek için öncelikle işlev puanı heasplanmalıdır.
İşlev puanı hesaplamak için gerekli formül:

FP = Toplam sayı * (0.65 + 0.01 * Toplam CAF)

Problemin girdileri çıktıları gibi bazı parametreler problemin zorluk derecesine göre belirlenmiş olan sabit katsayıları ile çarpılıp toplanması sonucu bulunur. Bunun için öncelikle problemin ölçüm parametreleri belirlenmelidir.

Girdi Sayısı: - birim zamanda üretilen enerji
- ünitenin devreye girme maliyeti
- birim zamanda talep edilen yük
- rezerv üretim
- kapasite limitleri

Çıktı Sayısı: - üretilen enerji
- açılıp kapatılan üniteler

Sorgulama Sayısı: - Hangi birimler açık
- Hangi birimler kapalı

Dosya Sayısı: -
Dış Arayüz Sayısı:-

Ölçme parametresi	adet		Ortalama		
girdi sayısı	5	x	4	=	20
çıktı sayısı	2	x	5	=	10
sorgulama sayısı	2	x	4	=	8
dosya sayısı	0	x	10	=	0
dış arayüz sayısı	0	x	7	=	0
Toplam Sayı =					38

Karmaşıklık Ayarlama Faktörleri(CAF):

1. Yedekleme ve kurtarma işlemi gerekli mi?	1
2. Veri İletişimi gerekiyor mu?	2
3. Dağıtık İşlem ve Süreçler var mı?	3
4. Çabukluk Önemli mi?	5
5. Sistem mevcut ve fazla yüklü bir ortamda mı çalışacak?	1
6. Çevrimiçi veri girişi gerekecek mi?	0
7. Çevrimiçi giriş, fazla ekranlı veya fazla işlemli mi?	0
8. Ana kütükler çevrimiçi olarak mı güncellenecek?	4
9. Girdi, çıktı, sorgulama ve kütükler karmaşık mı?	3
10. İç süreç (internal process) karmaşık mı?	5
11. Program yeniden kullanılabilir olarak mı tasarlanıyor?	3
12. Dönüştürme (conversion) ve kurma (installation), tasarımın içinde yer alıyor mu?	0
13. Değişik kuruluşlarda çoklu kurmalar tasarlanıyor mu?	0
14. Kullanıcının kolaylığı ve uyarlamasına göre tasarlanıyor mu?	3

Toplam CAF = 30

İşlev Puanı = (Toplam Sayı) * (0,65 + 0,01 * Toplam CAF)
= 38 * (0,65 + 0,01 * 30) = 36,1

LOC = İşlev Puanı * $30_{LOC/FP}$
= 36,1 * 30 = 1083 satır Java kodu

Orta Detayda COCOMO Çaba Formülleri:

Çaba : $pm = a * KLOC^{b} * EAF\_toplam$

Geliştirme zamanı : $t_{dev} = c * pm^{d}$

Çaba Ayarlama Etkenleri **(EAF)** :

Etken	**Aralık**	**Tahmin**
Ürün Özellikleri		
1.Gerekli güvenilirlik	0.75 – 1.40	1,30
2.Veritabanı büyüklüğü	0.94 – 1.16	0,94
3.Ürün karmaşıklığı	0.70 – 1.65	1,55
Donanım Özellikleri		
4.Yürütme zamanı kısıtları	1.00 – 1.66	1,05
5.Bellek yeterliliği	1.00 – 1.56	1,12
6.Sanal makina değişkenliği	0.87 – 1.30	0,87
7.Kullanılabilme süresi	0.87 – 1.15	1,00
İnsan Özellikleri		
8.Çözümleyici yeteneği	1.46 – 0.71	1,00
9.Programcı yeteneği	1.42 – 0.70	1,00
10.Uygulama yeteneği	1.29 – 0.82	1,00
11.Geliştirme ortamı deneyimi	1.21 – 0.90	0,90
12.Programlama dili deneyimi	1.14 – 0.95	0,95
Proje Özellikleri		
13.Yeni programlama teknikleri	1.24 – 0.82	0,95
14.Yazılım araçları kullanımı	1.24 – 0.83	1,00
15.Zamanlandırma	1.23 –1.10	1,20

EAF (tahminlerin çarpımı) = 1,89

YM kategori	a	b	c	d
Organik	3.2	1.05	2.5	0.38
Yarı ayrık	3.0	1.12	2.5	0.35
Gömülü	2.8	1.20	2.5	0.32

Proje yarı ayrık türdendir. O halde,

$pm = 3 * 1{,}083^{1{,}12} * 1{,}89 = 6{,}2$ (Çaba tahmini)

$t_{dev} = 2{,}5 * 6{,}2^{0{,}35} = 4{,}73$ (Zaman tahmini)

Bu proje için gerekli insan sayısı:

$6{,}2 / 4{,}73 = 1{,}3$

4. Risk Yönetimi

Projenin başarıyla tamamlanması önünde birçok risk vardır:

1. Projenin yazılacağı dil olan Java programlama dilinin bilinmemesidir. Bu dili öğrenmek beklenenden fazla zaman alırsa projenin bitirilme zamanı buna bağlı olarak uzamak durumunda kalır. Bu dilin öğrenilmesi konusunda bir sorunla karşılaşılırsa bu dilde programlama deneyimi olan arkadaşlardan yardım alınacaktır.
2. Evrimsel Algoritmalar ile daha önce hiç tanışılmamış olunması. Evrimsel Algoritmalar ile şu ana kadar hiç çalışılmamış olunmasından dolayı bu algoritmalardaki genel mantığı anlamak ve koda dönüştürmek vakit alabilir. Bu işlem de bu iş için belirlenmiş olan süreyi aşmamalıdır. Aksi durumda projenin zamanında bitmesi riske girer. Bu bitirme ödevinin zamanında teslim edilememesine ve bunun sonucu olarak zamanında mezun olamamaya bile neden olabilir. Algoritmada takılınılan bir yerde hemen Yrd.Doç.Dr.Şima Etaner-Uyar'dan yardım alınacaktır.
3. Elektrik enerjisinin optimizasyonuna ilişkin konuları, gerekli formülleri ve kullanım yöntemlerini bilinmemesidir. Bu yüzden algoritmayı yazdıktan sonra asıl problemi algortimaya uyarlamada zorlanabilinir. Bu işlem doğru ve zamanında bitirilmelidir. Doğru uyarlanmadığı takdirde projede başarılı olunmamış olur. Zamanında bitmezse önceki maddede belirtildiği gibi bitirme ödevinin zamanında teslim edilememesine ve bunun sonucu olarak zamanında mezun olamamaya bile neden olabilir. Bu konuda takılınılan bir yerde Doç.Dr.Belgin Türkay'dan yardım alınması düşünülmektedir.

5. Zamanlama

Projede iş paketleri aşağıdaki gibi sıralanabilir:

1. Problemin analizi
2. Java programlama dilinin öğrenilmesi
3. Evrimsel Stratejiler Algoritmasının kodunun yazılması
4. Kodun sonucu bilinen bir örnek üzerinde test edilmesi
5. Kodun probleme uyarlanması
6. Kod üzerinde testler
7. Raporlama

Hafta / Paket	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Problemin analizi	X	X	X													
Java Öğrenme	X	X	X	X												
ES kodu yazma				X	X	X	X									
ES kodu testi							X	X								
Probleme uyarlama									X	X	X					
Testler												X	X	X	X	
Raporlama														X	X	X

6. Proje Kaynakları

Proje için gerekli olan Java dili geliştirme platformu internetten temin edildi. Javayı öğrenmek için gerekli doküman internetten temin edilmiştir. Evrimsel Stratejiler ve Ünite Programlama Problemi ile ilgili gerekli dökümanlar hocalarımızın yardımıyla temin edilmiştir.

7. Proje Grubu organizasyonu

Proje tek başına gerçeklenecektir. Bu proje konusu 4 kişiye verilmiştir. Her kişi problemi farklı algoritmalar kullanarak gerçekleyecektir. Projeyi gerçeklemede kişiler arasında her hangi bir bağlantı yoktur.

3 KURAMSAL BİLGİLER

1. Ünite Programlaması Problemi

Ünite Programlaması Problemi (Unit Commitment Problem(UC Problemi)), elektrik enerjisi üretme birimlerinin açma kapama programlarını, toplam üretim maliyetini en aza indirgemek amacıyla, tüm kısıtları sağlarken, yük gereksinimlerini karşılayacak şekilde belirlemektir [1,2,3,4].

UC problemi genel olarak ikiye ayrılabilir. Birincisi, her saat diliminde hangi jeneratörlerin çalışacağını belirlemek; ikincisi, her jeneratörün üretmesi gereken ekonomik güç seviyesini belirlemektir (Economic Dispatch Problem(ECP)) [1,4]. Birinci sorun Evrimsel Stratejiler Algoritmasıyla, ikinci sorun Lambda İterasyonuyla çözümlenecektir.

UC Probleminde amaç maliyet terimlerinin toplamını en aza indirmektir. Maliyet terimlerinden birincisi, çalışan jeneratörlerin çalışması sırasında ürettikleri enerjinin maliyeti; ikinicisi jeneratörlerin çalışmaya başlatılma maliyetleridir, ki bu jeneratörlerin o anki genel sıcaklıklarına bağımlıdır. Bu maliyet terimleri (yakıt ve başlatılma maliyetleri) aşağıda ayrıntılı olarak açıklanacak, gerekli formüller [1] ve [5] makalelerindeki notasyona uygun olarak aktarılacaktır.

Problemi ayrıntılı olarak açıklamaya geçmeden önce kullanılan değişkenlerin ne anlama geldiklerini yazmakta fayda görülmüştür. Formülasyonda kullanılacak tüm değişkenler aşağıdaki gibidir:

$CF_i(p)$: p birim güç üretilmesiyle sebep olunan maliyet
SU_i: i. jeneratörün başlatılma maliyeti
$u(t)$: t saatinde üretilmesi istenen güç
$r(t)$: t saatinde rezerv edilmesi istenen güç
$P_i(t)$: i. jenerator tarafından t saatinde üretilen güç
$v_i(t)$: 0 ise jeneratör kapalı, 1 ise jeneratör açık
$x_i(t)$: jeneratörün kaç saattir açık(+) ya da kapalı(-) olduğu
$I(x)$: x yanlış ise 0, doğru ise 1 verir

1.1 Yakıt maliyeti:

Jeneratör çalışırken sebep olunan maliyettir. Bunun için bazı kısıtlar bulunmaktadır. Bunlar;

- Üretilen güç üretilmesi istenen gücü karşılamalıdır,

$$\sum_{i=1}^{N} P_i = \mathrm{u} \qquad (1)$$

- Her jeneratörün üretebileceği maksimum ve minimum güç değeri vardır. Jeneratörün bu sınırlar içinde bir güç üretmesi gerekir.

$$P_i^{\min} \le P_i \le P_i^{\max} \quad i = 1,2,...,N \qquad (2)$$

Bu probleme Ekonomik Yük Paylaşımı Problemi (Economic Dispatch Problem (EC Problemi)) denir.
EC Problemi Lagrange Çarpanları kullanılarak aşağıdaki gibi formüle edilebilir.

$$\frac{\partial}{\partial P_i(t)}\left\{\sum_{i=1}^{N} CF_i\left(P_i(t)\right)+\lambda(t)\left[u(t)-\sum_{i=1}^{N} P_i(t)\right]\right\} v_i(t)=0 \quad i=1,...,N \qquad (3)$$

EC probleminin amacı yakıt maliyetini minimize etmektir.

$$\min CF = \sum_{i=1}^{N} CF_i\left(P_i\right) \qquad (4)$$

Bir saat diliminde üretilen gücün maliyeti aşağıdaki formülle hesaplanabilir.

$$CF_i(p) = a_{0i} + a_{1i}\,p + a_{2i}\,p^2 \qquad (5)$$

Her jeneratörün üretmesi gereken güç miktarını bulmayı hedefleyen EC Problemi ((3) formülü) Lambda İterasyonu yöntemiyle çözümlenir. Lambda İterasyonunun algoritması aşağıda verilmiştir. Sonrasında açıklama yapılacaktır.

```
λ(t), Δ ve sayaca ilk degerlerini ver
Tekrala
       i = 1'dan N'ye kadar
            dCFi (Pi (t))/ dPi (t) =λ(t) denkleminden Pi(t)'yi
hesapla
       Hesapla  φ =u(t) - ∑ Pi (t)
       Eğer (φ <0) ise  λ(t) = λ(t) -Δ
            yoksa λ(t) = λ(t) + Δ
       Δ = Δ /2
|φ| > tolerance ve sayaç < 1000 olduğu sürece
```

(3) formülünde yer alan λ değerine iterasyonun başında ilk değer verilmelidir. λ'nın ilk değeri λ_{min} ve λ_{max} değerlerinin ortalamasıdır. Δ'nın ilk değeri ise λ_{max} ve λ_{min} değerlerinin farkının yarısıdır.

$$\lambda = \frac{\lambda_{\min} + \lambda_{\max}}{2} \quad , \quad \Delta = \frac{\lambda_{\max} - \lambda_{\min}}{2} \qquad (6)$$

λ_{min} jeneratörlerde üretilen minimum gücün maliyetlerinin güce türevinin minimumudur. λ_{max} jeneratörlerde üretilen maksimum gücün maliyetlerinin güce türevinin maksimumudur. Aşağıdaki formüllerle gösterilebilir:

$$\lambda_{\min} = \min_{i=1,n}\left\{\frac{dCF_i\left(P_{i,\min}\right)}{dP_i}\right\}, \qquad \lambda_{\max} = \max_{i=1,n}\left\{\frac{dCF_i\left(P_{i,\max}\right)}{dP_i}\right\} \qquad (7)$$

Her jeneratörün üretmesi gereken güç miktarı o anki saat diliminin maliyetinin güce türevinin o saat dilimindeki λ'ya eşitlenmesiyle bulunur. λ toplam gücün istenilen gücü karşılayıp karşılamamasına göre güncellenir. İstenen gücün karşılanmış olması belli bir tolerans ile kabul edilebilir. Ayrıca iterasyondan bu tolerans dahilinde bir güç sağlanıp çıkılamıyorsa, programın sonsuz döngüye girmemesi açısından belli bir sınır değeri koyulur. İterasyon en fazla bu sınır değeri kadar tekrarlanabilir.

İterasyonun tamamlanmasıyla elde edilmiş olan P_i değerleri jeneratorlerin o saat diliminde üretecekleri güç miktarını verir. (5) formülüyle o saat diliminde üretilecek gücün maliyeti hesaplanır. Toplam maliyete eklenir. Bu iterasyon her saat dilimi için tekrarlandıktan sonra, toplam yakıt maliyeti hesaplanmış olur.

1.2. Başlatılma Maliyeti:

UC Probleminde maliyeti etkileyen diğer bir terim ise başlatılma maliyetidir. Başlatılma maliyeti jeneratörün ne zaman kapatıldığına göre değişir. Çünkü jeneratör kapatıldıktan sonra soğumaya başlar. Jeneratör kapatıldıktan sonra tam soğumadan tekrar açılırsa bunun getirdiği maliyet daha azdır. Her jeneratör için soğuma süresi farklıdır. Bu süreyi t_{cold_start} temsil etmektedir. S_c, jeneratör soğukken açılırsa sebep olunan maliyet; S_h jenerator sıcakken tekrar açılırsa sebep olunan maliyettir. Aşağıdaki gibi formüle edilebilir:

$$S(t) = \begin{cases} S_h, & -x(t) \le t_{cold\_start} \\ S_c, & otherwise \end{cases} \qquad (8)$$

1.3. Başarım:

UC Probleminde başarım; yakıt maliyeti, başlatılma maliyeti ve eğer bazı kısıtlar sağlanamamışsa bunlardan dolayı oluşmuş olan ceza maliyetinin toplamıdır. İstenilen ceza maliyetinin hiç oluşmaması ve başarım değerinin mümkün olduğunca küçük olmasıdır.

$$\text{Basarım} = \text{Yakıt\_Maliyeti} + \text{Başlatılma\_Maliyeti} + \text{Ceza\_Maliyeti} \qquad (9)$$

Hangi kısıtların ceza maliyetine sebep olduğu ve bu maliyetin nasıl hesaplandığı aşağıda anlatılacaktır.

1.4. Ceza Maliyeti:

Ceza maliyeti, istenilen yük karşılanmıyorsa olmalı ve jeneratörlerin sağlıklı çalışması açısından açık ya da kapalı kalması gereken minimum süre karşılanmıyorsa oluşur.

Karşılanmayan yükleme ve güç rezervi gerekleri aşağıdaki formülle hesaplanır. Karşılanmamış güç m kat sayısı ile çarpılarak kaybedilen gücün ne kadara mal olduğu hesaplanır.

$$LP = m\left\{\sum_{t=1}^{T}\max\left\{0, u(t)+r(t)-\sum_{i=1}^{N}v_i(t)P_i^{\max}\right\}+\sum_{t=1}^{T}\max\left\{0, u(t)-\sum_{i=1}^{N}v_i(t)P_i(t)\right\}\right\} \qquad (10)$$

Jeneratörlerin açık ya da kapalı kalması gereken zaman kısıtından kaynaklı hata evrimsel algoritmanın kaçıncı nesli ürettiğine bağlı olarak giderek artar. Bu ceza,

$$Min\ up\,|\,down\ \ penalty = k \times n \times \sqrt{ngen} \qquad (11)$$

formülüyle hesaplanır. Burada n her saat diliminde jeneratörlerin açma ya da kapama süresi kısıtını ihlal etmeleriyle birer birer artan bir tam sayıdır. ngen nesil sayısını tutar. k ise tanımlamış olan bir sabittir.

2. Evrimsel Stratejiler Algoritması

2.1 Tarihçesi:

İlk olarak 1963'te Berlin Teknik Üniversitesi'nde aerodinamik tüneli deneyleri yaparken karşılaşıldı ve üzerinde deneyler yapılmaya başlandı. Bu sezgisel tahminler yapma işlemini bir stratejiye dönüştürme fikri ileri sürüldü, ancak başarılı olunamadı.

Sonra Ingo Rechenberg adlı bir öğrenci ki şimdi Biyonik ve Evrimsel Mühendis Profesörü, bu problem için doğal mutasyon örneklerini takip eden bir fikir öne sürdü. Böylece evrimsel stratejiler doğmuş oldu. Sonra üçüncü bir öğrenci Peter Bienert onlara katıldı ve mutasyonun basit kuralları ve seleksiyona göre çalışan program yazmaya başladı. Diğer bir öğrenci Hans-Paul Schwefel bu yeni metotların verimliliğini Zuse Z23 bilgisayarı ile test etti. Maddi destek pek alıyor olmasalar bile Evrimsel Mühendisler Grubu sık sık bir araya geldi.

Ingo Rechenberg doktorasını 1970'te aldı. Doktorasında Evrimsel Stratejilerle ilgili birkaç teoriyi açıklıyordu. Aynı yıl Deutsche Forschungsgemeinschaft (Almanya'nin Ulusal Bilim Kuruluşu) tarafından maddi destek geldi ve 1974'te "Evolutionsstrategie und numerische Optimierung" tezi ile sonuçlandı. Böylece Evrimsel Stratejiler teknik eniyileme problemlerini çözmek üzere bulunmuş oldu. [10].

2.2 Geliştirimi:

Evrimsel algoritmalar Darwin'in evrim teorisinden etkilenerek üretilmiştir. Problemler canlıların doğasında var olan evrimleşme sürecine uyarlanarak çözümlenir. Tüm evrimsel algoritmalar altında yatan fikir aynıdır. Belli sayıda bireyden oluşan rastgele bir toplum oluşturulur. Bu toplumdan yeni bireyler üretilir. Bu bireyler rekombinasyon ve mutasyona uğratılır. Sonra Darwin'in kuramındaki ortama daha iyi adapte olanların hayatta kalması gibi çocuklardan da çözüme daha uygun olarak görülen bireyler kalırken, diğerleri yok olur. Bu durum yeterince iyi bireyler üretilene kadar devam eder [7].

Evrimsel Stratejiler Algoritması [6,8,9] da genel evrimsel algoritmalar mantığında işler. İlk toplum rastgele olarak üretilir. Bir toplum μ adet bireyden oluşmaktadır. Bireyi bir kromozom temsil eder. Kromozom iki bölümden oluşmaktadır; 0 ve 1'lerden oluşan genler ve herbir gen için ayrı ayrı tutulan ve gerçel sayılardan oluşan mutasyona uğrama olasılıkları. İlk toplum için tüm bu değerler rastgele olarak üretilir. Bu toplumdan λ adet çocuk üretilir. Bir çocuğu üretmek için her seferinde toplumdan bir anne ve baba seçilir. Bu bireyler rekombinasyon ile birleştirilir ve mutasyona uğratılır. Oluşturulmuş olan her çocuk çözüme adaydır. Ancak çocuklardan çözümü gerçeklemeye daha uygun olan μ birey hayatta kalır, diğerleri yok olur. Hayatta kalanlar yeni toplumu oluşturmuş olur. Bu yeni toplum da aynı adımları izleyerek bir sonraki toplumu oluşturur. Bu işlem yeterince iyi birey bulunana kadar ya da yeterince iyinin ne olduğunun belirlenemediği durumlarda belirli bir sayıda tekrarlanır. Evrimsel Stratejiler Algoritması süreci aşağıdaki gibi verilebilir. Algoritmanın her adımının nasıl gerçeklendiği sonrasında ayrıntılı olarak anlatılacaktır.

```
Rastgele toplum üret
Sonlanma kriteri gerçekleşene kadar tekrarla
        λ kere tekrarla
                Rastgele anne ve baba seç
                Rekombinasyonu gerçekleştir
                Mutasyona uğrat
        bitir
        Doğal seçilime uğrat
bitir
```

2.2.1 Rastgele Toplum Üretme:

İlk toplum rastgele olarak üretilir. Bunun için kromozomun genleri rastgele olarak 0 ya da 1 şeklinde belirlenir. Kromozomun mutasyon olasılıkları

$$1/\text{gensayısı} + N(0,1) \times \tau$$

formülünden hesaplanır. N(0,1); 0 ortalamalı 1.0 standart sapmalı Gaussian dağılımına göre belirlenmiş random bir sayıdır. τ belirlenmiş bir sabittir.

2.2.2 Ebeveyn Seçimi:

Toplumdan rastgele iki farklı birey seçilir. Her birey aynı seçilme olasılığına sahiptir.

2.2.3. Rekombinasyon:

Rekombinasyon ile bir seferde bir ya da iki çocuk üretilir. ES'de iki standart rekombinasyon türü kullanılır: Biri, ayrık(baskın) rekombinasyon; diğeri, ara değer rekombinasyondur.

Ayrık rekombinasyonda çocuk kromozomun ilgili değeri için ya annenin ya da babanın ilgili kromozom değeri alınır.
Ara değer rekombinasyonda çocuk kromozomun ilgili değeri için ilgili anne ve baba kromozomlarının ortalaması alınır.

Bu çalışmada çocukların kromozomlarındaki genlerin belirlenmesi için ayrık rekombinasyon, mutayon olasılıklarının belirlenmesi için ara değer rekombinasyon kullanılmıştır ve her eşleştirmede yalnız bir çocuk üretilmektedir.

Çalışmada rekombinasyon işlemi şu şekilde gerçekleştirilir; kromozomun genleri üzerinde herhangi iki nokta seçilir. Kromozomun başından ilk noktaya kadar anne kromozomun, ilk noktadan ikinci noktaya kadar baba kromozomun, ikinci noktadan sonuna kadar ise yine anne kromozomun genleri çocuk kromozomun genlerine kopyalanır. Kromozomun mutasyon olasılığı için ise anne ve baba kromozomun sıradaki mutasyon olasılıklarının ortalamaları alınır, çocuk kromozomun sıradaki mutasyon olasılığı olarak kopyalanır.

2.2.4. Mutasyon:

Mutasyon, gen yapısında meydana gelen değişim demektir [11]. Bu çalışmada kromozomun genleri oluşturulmuş olan mutasyon olasılığına göre değiştirilmektedir. Çocuk kromozomun ilgili geninin mutasyon olasılığı $N(0,1) \times \tau$ kadar artırılır. Bu yeni oluşmuş olan mutasyon olasılığı, random olarak üretilmiş olan double sayıdan büyükse ya da eşitse çocuk kromozomun ilgili geni tümlenir.

2.2.5. Doğal Seçilim:

Darwin'e göre doğal seçilim, ortama daha iyi adapte olanların hayatta kalıp üremesi, geri kalanların yok olması demektir [11]. Evrimsel Stratejiler algoritmasında hayatta kalacaklar iki şekilde seçilebilir. μ kadar birey ya sadece oluşturulmuş olan çocukların ya da çocuklar ve bireylerin toplamının çözüme en uygun olanları arasından seçilir. İkinci yöntemde iyi olan bireyler kaybolmamaktadır, ancak iterasyonlar çok fazla olduğu için bir yerden sonra hep aynı kromozomların toplumu oluşturduğu görülmüştür. Bu yüzden birinci yöntem daha etkilidir ancak yine de iyi bireylerin kaybolmaması ve daha iyi nesiller üretilmesi için elitizm yöntemi kullanılır.

Elitizm; eşleşme, rekombinasyon, mutasyon işlemleri sırasında oluşan çocuklar arasında şuanki toplumdaki en iyi bireyden daha iyi bir birey bulunmayabilir. Bu yüzden iyi bir birey ve onun üretebileceği iyi bir nesil imkanı yok olabilir. Bunu önlemek amacıyla bir önceki neslin en iyi bireyi yeni nesildeki herhangi bir bireyle değiştirilir. Bu işleme elitizm denir [12].

Yapılan bu çalışmada bir sonraki nesil sadece çocuklardan oluşturulmuş ve en kötü birey için elitizm yöntemi kullanılmıştır. Yani, en kötü birey bulunarak bir önceki neslin en iyi bireyiyle yer değiştirilmiştir. Böylece en iyi birey korunmuştur.

4 ANALİZ VE MODELLEME

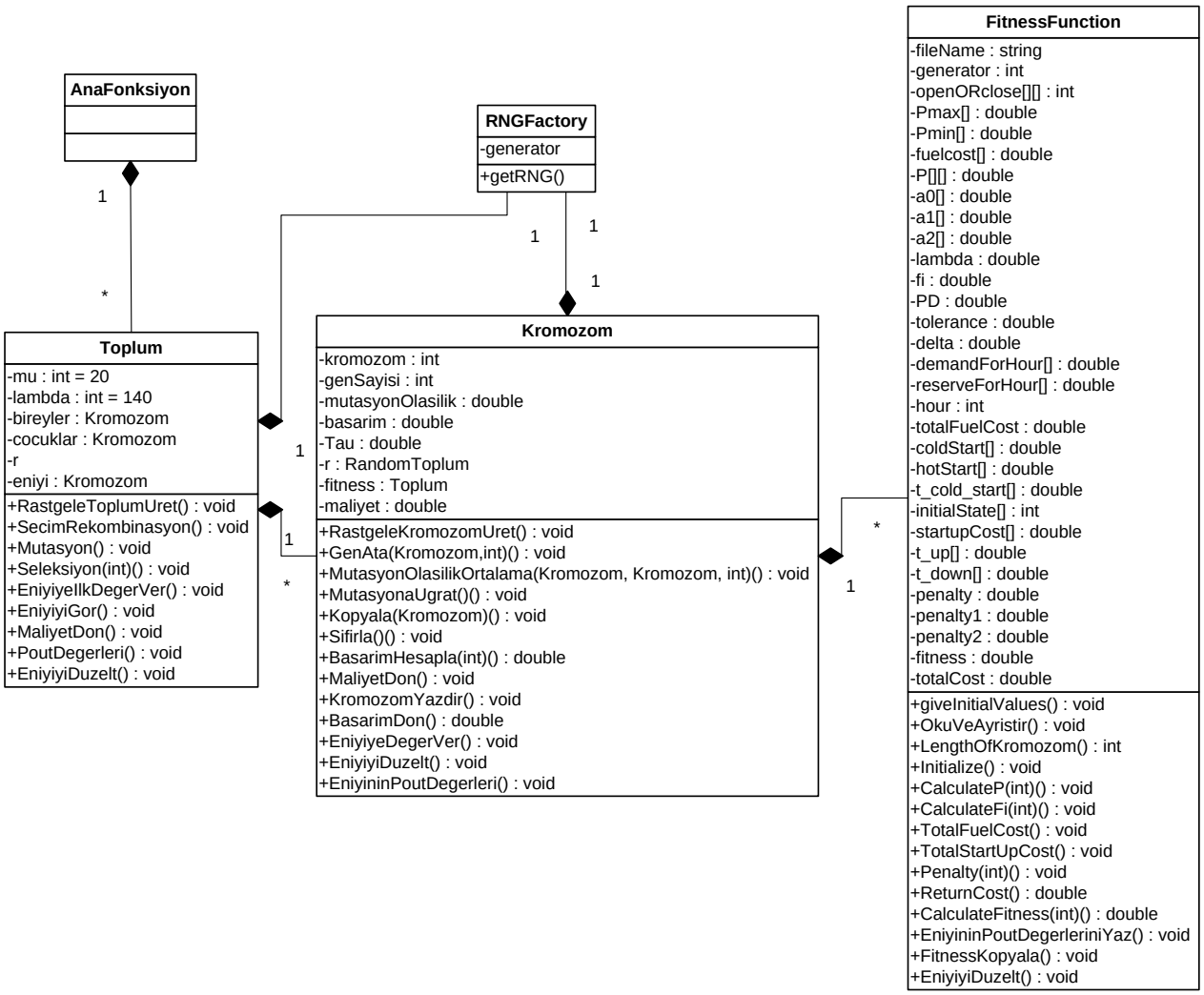
Ünite Programlaması Probleminde gerekli en önemli bilgiler jeneratörlere ilişkin kısıtlardır. Hangi jeneratörün en fazla ve en az ne kadar güç üretebileceği, yakıt maliyeti hesaplanırken gereken a_0, a_1, a_2 katsayıları, jenerator açılırsa en az kaç dakika açık kalması gerektiği, jenerator kapatılırsa en az kaç saat kapalı kalması gerektiği, bir jeneratörün tamamen soğuması için gerekli olan süre, jeneratör henüz soğumamışken tekrar açılırsa sebep olunan maliyet, jeneratör soğumuşken tekrar açılırsa sebep olunan maliyet ve program çalıştırılmaya başlandığında jeneratörlerin o ana kadar kaç saattir açık ya da kapalı olduğu bilgileri dışarıdan alınmak zorundadır.

Evrimsel Stratejiler Algoritmasında, kromozom bireyleri oluşturmaktadır. Bir bireyi bir kromozom temsil eder. Kromozomun gen, mutasyon olasılığı, maliyet, basarim gibi değişkenleri bulunmak zorundadır. Problemi çözmek için tek bir birey yeterli olmaz. Bir toplum oluşturmak gerekir. Toplum bireylerden meydana gelir. Bir toplumun kaç bireyi bulunduğu ve bir toplumda kaç çocuk üretilebildiği bilgileri de toplumun bilgisi dahilinde olmalıdır. Toplumdaki birey sayısının sabit olası gerekmektedir. Bunun için üretilen bir sürü çocuktan toplumdaki birey sayısı kadarı seçilmek zorundadır. Bu çocuklardan çözüme en uygun olanları seçilmelidir. Çocuk kromozomların çözüme uygun olup olmadığı test edilmelidir. Bunun için kromozomların uygunluğunu hesaplayan bir sınıf daha bulunmalıdır. Bu sınıf ünite programlaması problemi için gerekli olan jeneratörlere ilişkin bilgilerin tutulduğu sınıf olmalıdır.

Uygun çocukların kaldığı geri kalanların yok olduğu bir toplumda her neslin en iyisi bulunmalı ve en iyi bireyin nesillerce aktarılması sağlanmalıdır. Kendisinden daha iyi bir bireyle karşılaşıldığında en iyi olarak bu yeni birey tutulmalı ve aynı şekilde sonraki nesillere aktarılmalıdır.

Ayrıca Evrimsel Stratejilerin doğası gereği bazı rastgele sayılar elde edilmek zorundadır. Bu rastgele sayılar farklı diziler üzerinde olmamalıdır. Hepsi aynı tohum değeri üzerinde başlamış dizi üzerinden seçilmelidir. Bunun için rastgele sayı üretmek üzere gerekli olan dizi bir kez yaratılmalı ve bu dizi üzerinden rastgele değerler elde edilmelidir.

Tüm bu gereksinimler için sınıflar oluşturulması düşünülmüştür. Gerekli modelleme için UML sınıf diyagramı aşağıdaki gibidir:


AnaFonksiyon
RNGFactory
-generator
+getRNG()
FitnessFunction
-fileName : string
-generator : int
-openORclose[][] : int
-Pmax[] : double
-Pmin[] : double
-fuelcost[] : double
-P[][] : double
-a0[] : double
-a1[] : double
-a2[] : double
-lambda : double
-fi : double
-PD : double
-tolerance : double
-delta : double
-demandForHour[] : double
-reserveForHour[] : double
-hour : int
-totalFuelCost : double
-coldStart[] : double
-hotStart[] : double
-t_cold_start[] : double
-initialState[] : int
-startupCost[] : double
-t_up[] : double
-t_down[] : double
-penalty : double
-penalty1 : double
-penalty2 : double
-fitness : double
-totalCost : double
+giveInitialValues() : void
+OkuVeAyristir() : void
+LengthOfKromozom() : int
+Initialize() : void
+CalculateP(int)() : void
+CalculateFi(int)() : void
+TotalFuelCost() : void
+TotalStartUpCost() : void
+Penalty(int)() : void
+ReturnCost() : double
+CalculateFitness(int)() : double
+EniyininPoutDegerleriniYaz() : void
+FitnessKopyala() : void
+EniyiyiDuzelt() : void
Toplum
-mu : int = 20
-lambda : int = 140
-bireyler : Kromozom
-cocuklar : Kromozom
-r
-eniyi : Kromozom
+RastgeleToplumUret() : void
+SecimRekombinasyon() : void
+Mutasyon() : void
+Seleksiyon(int)() : void
+EniyiyeIlkDegerVer() : void
+EniyiyiGor() : void
+MaliyetDon() : void
+PoutDegerleri() : void
+EniyiyiDuzelt() : void
Kromozom
-kromozom : int
-genSayisi : int
-mutasyonOlasilik : double
-basarim : double
-Tau : double
-r : RandomToplum
-fitness : Toplum
-maliyet : double
+RastgeleKromozomUret() : void
+GenAta(Kromozom,int)() : void
+MutasyonOlasilikOrtalama(Kromozom, Kromozom, int)() : void
+MutasyonaUgrat()() : void
+Kopyala(Kromozom)() : void
+Sifirla()() : void
+BasarimHesapla(int)() : double
+MaliyetDon() : void
+KromozomYazdir() : void
+BasarimDon() : double
+EniyiyeDegerVer() : void
+EniyiyiDuzelt() : void
+EniyininPoutDegerleri() : void
1
*
1
1
1
1
1
*
1
*
1

5 TASARIM, GERÇEKLEME VE TEST

1. Tasarım ve Gerçekleme:

Ünite Programlaması Probleminin çözülmesi için, Evrimsel Stratejiler(ES) koduna uyarlanmalıdır. Ünite Programlaması Problemi için öncelikle hangi jeneratörlerin hangi saat diliminde açık, hangi saat diliminde kapalı olduğu belirlenmelidir. ES kodunun ürettiği toplumdaki her bir kromozom jeneratörlerin her saat dilimi için açık ya da kapalı olduğu bilgisini tutar. Bunun için öncelikle kromozomlar, satırları saat bilgisini gösteren, sütunları jeneratör bilgisini gösteren matrise dönüştürülür. Bu işlem her çocuk kromozomu için yapılır. Oluşturulan matris için başarım değeri kuramsal bilgilerde anlatıldığı gibi hesaplanır. Başarım değeri düşük olan çocuklar, problemi çözmek için daha uygundur. ES kodunun diğer bir adımı olan Doğal Seçilim işleminde başarım değeri düşük olan μ kadar çocuk bir sonraki neslin bireylerini oluşturmak üzere seçilir, geri kalan çocuklar yok olur. ES algoritması belirlenen sayıda tekrarlanır. Bu sayının mümkün olduğu kadar fazla olması daha verimli sonuçlar elde edebilme adına faydalı olur. Oluşturulan her nesil için en iyi birey seçilir ve bilgileri saklanır. Tüm iterasyonlar sonlandıktan sonra saklanmış olan en iyi birey o toplum için çözümü verir. Bu işlemler birkaç toplum için tekrarlanır.

Üretilmiş olan toplumların en iyi bireylerinin maliyetlerinin en kötü değeri, en iyi değeri ve ortalama değeri hesaplanır. Sonuçlar dosyaya yazılır. En iyi birey için her saatte hangi jeneratörde ne kadar güç üretildiği de dosyaya yazılır. Sonuçta her toplum için bulunmuş olan en iyi bireylerin en iyisi problemin çözümünü verir. Bu çözüm için son olarak elde edilmiş sonuçta eğer ceza oluşmuşsa jeneratörlerin açık ya da kapalı kalması gereken sürede sağlaması gereken bu koşulu sağlayıp sağlamadığına bakılır. Eğer koşulu sağlamıyorsa sağlayacak şekilde düzeltilir. Bunun için bu işlemi yapan bir düzeltme fonksiyonu yazılmıştır. Elde edilen bu yeni sonuç için tekrar maliyet ve başarım değeri hesaplanır.

Kodda hangi değişkenlerin olduğu ve hangi değerleri aldıkları aşağıda verilecektir. Ana fonksiyonda kaç tane toplum oluşturulacağına karar verilir. Toplum sayısı 5 olarak alınmıştır. Üretilen bir toplum üzerinde rekombinasyon, mutasyon, doğal seçilim işlemleri belli bir sayıda tekrarlanır. Bu iterasyon sayısı çok daha iyi nesiller elde edilebilmesi adına 3000 alınmıştır. Ancak testler yapılırken bu değerler üzerinde oynanmıştır. En iyi değerler elde edilmeye çalışılmıştır.

Toplum sınıfında tutulan birey sayısı 20, bireylerden üretilen çocukların sayısı 140 olarak alınmıştır. Ne kadar çok çocuk üretilirse en iyiye erişme şansımız o kadar artar. Her iterasyonda 20 bireyden rekombinasyon ve mutasyon ile 140 çocuk üretilecektir. Üretilen bu çocukların başarımı sınandıktan sonra çözüme en uygun olan 20 tanesi seçilip, aynı işlemler iterasyon sayısı tamamlanana kadar devam edecektir. Bu sırada her seferinde oluşturulmuş olan neslin en iyi bireyi, tutulan en iyi birey ile karşılaştırılacak eğer yeni neslin en iyi bireyi şimdiye kadarki en iyi birey ise en iyi birey artık o olacaktır.

Çocukların başarımı hesaplanırken analiz ve modelleme kısmında dışarıdan alınması gerektiği söylenen kısıtlar dosyadan okunur. Bir jeneratorun en fazla ya da en az ne kadar enerji üretilebileceği, yakıt maliyeti hesaplanırken gereken a0, a1, a2 katsayıları, jenerator

açılırsa en az kaç dakika açık kalması gerektiği, jenerator kapatılırsa en az kaç saat kapalı kalması gerektiği, bir jeneratörün tamamen soğuması için gerekli olan süre gibi bilgiler sabittir ve bir sefer dosyadan okutulması ve sonraki her kromozom için bu değerlerin uygulanması yeterlidir. Yakıt maliyeti hesaplanırken, karşılanması istenen maliyet belli bir toleransla karşılandığı kabul edilebilir, bu tolerans 0.0001 olarak alınmıştır. Eğer bu tolerans sağlanarak döngüden çıkılamıyorsa, sonsuz döngüye girmemesi açısından iterasyon sayısı 1000 ile sınırlandırılmıştır.

Ceza değeri hesaplanırken jeneratörün üretmesi istenilen güç ve rezervler sağlanmıyorsa bundan dolayı karşılanamamış bir güç açığı oluşur. Bu güç açık ‘m’ katsayısı ile çarpılarak paraya dönüştürülür. Bu değer 200$/MWH olarak alınmıştır. Cezaya sebep olan bir diğer etki ise jeneratörlerin açık kalması gereken saat diliminde kapanması, kapalı kalması gereken zaman diliminde açılmasıdır. Bu kuralın kaç defa ihlal edildiğine bağlı olarak ceza maliyeti artar. Ceza değerini hesaplarken formülde yer alan ‘k’ katsayısı 100 olarak alınmıştır. Yine formülde yer alan ngen değişkeni nesil sayısını simgeler. Yani nesil sayısı arttıkça ceza daha fazla olacaktır.

2. Test:

Bu proje daha önceden başka algoritmalarla çözülmüş bir projedir. Ve bazı test verilerinin optimum değerleri de elde edilmiştir. Optimum değer, o jeneratörler grubunun sebep olabileceği minimum maliyet değeridir. Yapılan projenin test edilmesi için optimum değerleri elde edilmiş test verileri kullanılarak sonuçlar kıyaslanmıştır. Ayrıca başka algoritmalar kullanılarak çözülmüş olan veriler üzerinde de çalışılarak o algoritmadan elde edilmiş sonuçlarla bu çalışmadan elde edilen sonuçlar karşılaştırılarak test edilmiştir.

6 DENEYSEL SONUÇLAR

Proje çalıştırılırken veriler input.txt dosyasından okunur, sonuçlar output.txt dosyasına yazılır. Aşağıda projenin çalıştırıldığı bir kaç test sistemi verilecektir.

1. Test - 1:

Valenzuela ve Smith'in makelelerinde [1] yer verdiği aşağıdaki verilerle yapılan proje test edilmiştir. Bu test sisteminde 4 jeneratörün, 8 saat boyunca çalışması üzerine bir sistem oluşturulmuştur.

	1. jeneratör	2. jeneratör	3.jeneratör	4.jenerator
*P*max (MW)	300	250	80	60
*P*min (MW)	75	60	25	20
*a*0	684.74	585.62	213.00	252.00
*a*1	16.83	16.95	20.74	23.60
*a*2	0.0021	0.0042	0.0018	0.0034
*t*up (h)	5	5	4	1
*t*down (h)	4	3	2	1
*S*h($) (hot start)	500	170	150	0.00
*S*c($)(cold start)	1100	400	350	0.02
*t*cold start (h)	5	5	4	0
Initial state (h)	8	8	−5	−6

Hour :	1	2	3	4	5	6	7	8
Demand (MW) :	450	530	600	540	400	280	290	500
Reserve (MW) :	45	53	60	54	40	28	29	50

Yine aynı makalede bu test verilerinin farklı algoritmalarla çözülmüş değeri ve optimum değeri verilmiştir. Bu değerler ve Evrimsel Stratejilerle elde edilmiş sonuçlar aşağıda birlikte verilecektir.

Memetik Algoritma			Evrimsel Stratejiler		
En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
74675	74959	75012	74676	74825	75008

Sonuçlar Memetik Algoritma ile elde edilmiş olan sonuçlarla oldukça yakındır. Elde edilen ortalama maliyet değeri memetik algoritmada elde edilen ortalamaya nazaran daha iyidir. Bu da Evrimsel Stratejiler Algoritmasının daha iyi sonuçlar veren bir algoritma olmasından kaynaklanır.

Bu sistemin oluşmuş örnek bir output dosyası aşağıdaki gibi verilebilir. Output dosyasında hangi jeneratörün hangi saat diliminde ne kadar güç üreteceği, üretilmiş toplumların en iyilerinin maliyetlerinin en iyisi, ortalaması ve en kötüsü, bu toplumların standart sapması ve standart hatası yer almaktadır.

```
Hour  P0    P1    P2    P3
Hour0 300.0 150.0 0.0   0.0
Hour1 300.0 205.0 25.0  0.0
Hour2 300.0 250.0 30.06 20.0
Hour3 300.0 215.0 25.0  0.0
Hour4 300.0 0.0   80.0  20.0
Hour5 255.0 0.0   25.0  0.0
Hour6 265.0 0.0   25.0  0.0
Hour7 300.0 200.0 0.0   0.0

best    :   74676.09590136255
average :   74825.60715830351
worst   :   75008.34313900908

standard_deviation:     165.29091373583765
standard_error    :     36.96017191763867
```

Bu verilere bakıldığında herhangi bir saat diliminde o saatte üretilmesi istenen güç miktarının üretildiği görülmüştür.

Ayrıca hangi jeneratörün açık hangi jeneratörün kapalı olduğunu görmek amacıyla son olarak meydana gelmiş olan kromozom dizisi yukarıda verilen sonuçlar için aşağıdaki gibidir. Aşağıdaki dizi her saat dilimi için 4 jeneratörün de durumunu gösterir şekilde yazılmıştır. Bu sonuç için düzeltme fonksiyonu çağırıldığında ise değişen bir şey olmamıştır. Yine aynı dizi elde edilmiştir.

```
1 1 0 0  1 1 1 0  1 1 1 1  1 1 1 0  1 0 1 1  1 0 1 0  1 0 1 0  1 1 0 0
```

2. Test - 2:

2. test verileri de Valenzuela ve Smith'in makelelerinde yer verdiği 10 jeneratörün bağlı bulunduğu, 24 saati kapsayan bir sistemdir. Bu sisteme ilişkin veriler aşağıda verilmiştir.

Jn	Pmax	Pmin	a0	a1	a2	t_up	t_down	H_Strt	C_Strt	Tcold start	İnitial State
1	455	150	1000	16.19	0.00048	8	8	4500	9000	5	8
2	455	150	970	17.26	0.00031	8	8	5000	10000	5	8
3	130	20	700	16.60	0.00200	5	5	550	1100	4	-5
4	130	20	680	16.50	0.00211	5	5	560	1120	4	-5
5	162	25	450	19.70	0.00398	6	6	900	1800	4	-6
6	80	20	370	22.26	0.00712	3	3	170	340	2	-3
7	85	25	480	27.74	0.00079	3	3	260	520	2	-3
8	55	10	660	25.92	0.00413	1	1	30	60	0	-1
9	55	10	665	27.27	0.00222	1	1	30	60	0	-1
10	55	10	670	27.79	0.00173	1	1	30	60	0	-1

Hour	0	1	2	3	4	5	6	7
Load MW	700	750	850	950	1000	1100	1150	1200
Reserve MW	70	75	85	95	100	110	115	120
Hour	8	9	10	11	12	13	14	15
Load MW	1300	1400	1450	1500	1400	1300	1200	1050
Reserve MW	130	140	145	150	140	130	120	105
Hour	16	17	18	19	20	21	22	23
Load MW	1000	1100	1200	1400	1300	1100	900	800
Reserve MW	100	110	120	140	130	110	90	80

Aynı makalede bu test verilerinin farklı algoritmalarla çözülmüş değeri ve optimum değeri verilmiştir. Aşağıda Dinamik Programlama, Lagrangian Çarpanları, Memetik Algoritma, Genetik Algoritma ve bu projede uygulanmış olan Evrimsel Stratejiler Algoritmasıyla elde edilmiş sonuçlar aşağıda birlikte verilecektir.

Memetik Algoritma			Genetik Algoritma		
En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
565827	566453	566861	565866	567329	571336

Dinamik Programlama	Lagrangian Çarpanları			Evrimsel Stratejiler		
Optimum	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
565827	566107	566493	566817	566597	568211	569225

Algoritmamızdan elde edilen sonucun en iyi değeri, diğer algoritmalardan elde edilmiş en iyi sonuçların en iyi değerlerinden daha kötüdür. Ancak ES algoritmasının ortalama ve en kötü maliyet değerleri Genetik algoritmaya göre daha iyidir.

Bu sistemin oluşmuş örnek bir output dosyası aşağıdaki gibi verilebilir.

Hour	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
Hour0	455.0	245.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Hour1	455.0	295.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Hour2	455.0	375.0	0.0	0.0	0.0	20.0	0.0	0.0	0.0	0.0
Hour3	455.0	345.0	0.0	130.0	0.0	20.0	0.0	0.0	0.0	0.0
Hour4	455.0	395.0	0.0	130.0	0.0	20.0	0.0	0.0	0.0	0.0
Hour5	455.0	360.0	130.0	130.0	25.0	0.0	0.0	0.0	0.0	0.0
Hour6	455.0	410.0	130.0	130.0	25.0	0.0	0.0	0.0	0.0	0.0
Hour7	455.0	455.0	130.0	130.0	30.0	0.0	0.0	0.0	0.0	0.0
Hour8	455.0	455.0	130.0	130.0	85.0	20.0	25.0	0.0	0.0	0.0
Hour9	455.0	455.0	130.0	130.0	162.0	33.0	25.0	10.0	0.0	0.0
Hour10	455.0	455.0	130.0	130.0	162.0	73.0	25.0	10.0	10.0	0.0
Hour11	455.0	455.0	130.0	130.0	162.0	80.0	25.0	43.0	10.0	10.0
Hour12	455.0	455.0	130.0	130.0	162.0	33.0	25.0	10.0	0.0	0.0
Hour13	455.0	455.0	130.0	130.0	85.0	20.0	25.0	0.0	0.0	0.0
Hour14	455.0	455.0	130.0	130.0	30.0	0.0	0.0	0.0	0.0	0.0
Hour15	455.0	310.0	130.0	130.0	25.0	0.0	0.0	0.0	0.0	0.0
Hour16	455.0	260.0	130.0	130.0	25.0	0.0	0.0	0.0	0.0	0.0
Hour17	455.0	360.0	130.0	130.0	25.0	0.0	0.0	0.0	0.0	0.0
Hour18	455.0	455.0	130.0	130.0	30.0	0.0	0.0	0.0	0.0	0.0
Hour19	455.0	455.0	130.0	130.0	162.0	33.0	25.0	10.0	0.0	0.0
Hour20	455.0	455.0	130.0	130.0	85.0	20.0	25.0	0.0	0.0	0.0
Hour21	455.0	455.0	0.0	0.0	145.0	20.0	25.0	0.0	0.0	0.0
Hour22	455.0	425.0	0.0	0.0	0.0	20.0	0.0	0.0	0.0	0.0
Hour23	455.0	345.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
best   :    566597.2488274978
average:    568211.2966785885
worst  :    569224.6804409439

standard_deviation: 886.1574959474866
standard_error    : 396.30167994191487
```

Bu verilere bakıldığında herhangi bir saat diliminde o saatte üretilmesi istenen güç miktarının üretildiği görülmüştür.

Ayrıca hangi jeneratörün açık hangi jeneratörün kapalı olduğunu görmek amacıyla son olarak meydana gelmiş olan kromozom dizisi yukarıda verilen sonuçlar için aşağıdaki gibidir. Aşağıdaki dizi her saat dilimi için 10 jeneratörün de durumunu gösterir şekilde yazılmıştır. Bu sonuç için düzeltme fonksiyonu çağırıldığında ise değişen bir şey olmamıştır. Yine aynı dizi elde edilmiştir.

```
1 1 0 0 0 0 0 0 0 0  1 1 0 0 0 0 0 0 0 0  1 1 0 0 0 1 0 0 0 0
1 1 0 1 0 1 0 0 0 0  1 1 0 1 0 1 0 0 0 0  1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0  1 1 1 1 1 0 0 0 0 0  1 1 1 1 1 1 1 0 0 0
1 1 1 1 1 1 1 1 0 0  1 1 1 1 1 1 1 1 1 0  1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 0  1 1 1 1 1 1 1 1 0 0  1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0  1 1 1 1 1 0 0 0 0 0  1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0  1 1 1 1 1 1 1 1 0 0  1 1 1 1 1 1 1 0 0 0
1 1 0 0 1 1 1 0 0 0  1 1 0 0 0 1 0 0 0 0  1 1 0 0 0 0 0 0 0 0
```

3. Test - 3:

3. test Türkiye üzerinde bulunan 8 santral için yapılmıştır. Aşağıda bu santrallerin 8 saat dilimi için gerekli verileri verilmiştir.

	Hamitabat	Ambarlı	Bursa Doğal G.Ç.	SeyitÖmer	SomaB	Yeniköy	Kemerköy	Yatagan
	1	2	3	4	5	6	7	8
P_{max}(MW)	1120	1350	1432	600	990	420	630	630
P_{min} MW)	190	245	318	150	210	110	140	140
α_0	6595,5	7290,6	6780,5	1564,4	5134,1	1159,5	1697	1822,8
α_1	7,0063	7,2592	5,682	3,1288	6,232	3,3128	3,2324	3,472
α_2	0,0168	0,0127	0,0106	0,0139	0,0168	0,021	0,013	0,0147
t_{up}(h)	8	1	1	10	10	10	10	10
t_{down}(h)	2	0,5	0,5	3	3	3	3	3
S_h ($)	800	800	600	400	500	400	400	400
S_c ($)	1600	1600	1200	800	1000	800	800	800
$t_{coldstart}$(h)	8	1	1	10	10	10	10	10
İnitial start	-5	-5	-5	-5	-5	-5	-5	-5

Hour	1	2	3	4	5	6	7	8
Demand (MW)	2000	3000	6500	1500	4200	5100	2700	1750
Reserve	200	300	650	150	420	510	270	175

Bu veriler ile program 100 toplum için çalıştırıldığında aşağıdaki output dosyası elde edilmiştir.

```
Hour  P0    P1    P2    P3    P4    P5    P6    P7
Hour0 0.0   0.0   0.0   549.4 0.0   359.3 583.5 507.8
Hour1 0.0   0.0   741.0 600.0 0.0   420.0 630.0 609.3
Hour2 844.5 1107  1401  600.0 867.5 420.0 630.0 630.0
Hour3 0.0   0.0   0.0   413.4 0.0   269.3 438.1 379.3
Hour4 0.0   614.4 810.5 600.0 495.0 420.0 630.0 630.0
Hour5 0.0   919.1 1175.5 600.0 725.4 420.0 630.0 630.0
Hour6 0.0   0.0   626.3 569.5 0.0   372.5 604.9 526.8
Hour7 0.0   0.0   0.0   481.4 0.0   314.3 510.8 443.5

best   :    504502.9260378304
average:    504562.84181683103
worst  :    504835.79147671885
standard_deviation: 127.88272557301424
standard_error    : 12.788272557301424
```

Bu verilere bakıldığında herhangi bir saat diliminde o saatte üretilmesi istenen güç miktarının üretildiği görülmüştür.

Ayrıca hangi jeneratörün açık hangi jeneratörün kapalı olduğunu görmek amacıyla son olarak meydana gelmiş olan kromozom dizisi yukarıda verilen sonuçlar için aşağıdaki gibidir. Aşağıdaki dizi her saat dilimi için 8 jeneratörün de durumunu gösterir şekilde yazılmıştır. Bu sonuç için düzeltme fonksiyonu çağırıldığında ise değişen bir şey olmamıştır. Yine aynı dizi elde edilmiştir.

```
0 0 0 1 0 1 1 1  0 0 1 1 0 1 1 1  1 1 1 1 1 1 1 1  0 0 0 1 0 1 1 1
0 1 1 1 1 1 1 1  0 1 1 1 1 1 1 1  0 0 1 1 0 1 1 1  0 0 0 1 0 1 1 1
```

7 SONUÇ ve ÖNERİLER

Problem için üretilmiş çözüm ve çözümün performansı daha önce bu konuyu çözen projelerle kıyaslandığında iyi olduğu görülmüştür. Maliyeti çok olan problemler için üretilen maliyet değerleri biraz yüksek olsa da, diğer problemlerde elde edilmiş olan minimum ve maksimum maliyet değerlerini aşmamıştır. Çözümün daha da iyileştirilebilmesi için Yerel Arama (Local Search) kullanılabilirdi. Yerel Arama kullanıldığı takdirde en son elde edilmiş olan en iyi maliyet değerini veren kromozom dizisine 1 hamming uzaklığındaki dizileri arayarak elde edilmiş olan çözümden daha iyi bir çözüm elde edilebilirdi.

Daha sonra bu alanda çalışma yapacakların dikkat etmesi gereken en önemli konu, bu konuda yazılmış makaleleri çok dikkatli okumak ve makalelerdeki eksik kalmış bazı noktaları birbirlerinden tamamlamaktır.

8 KAYNAKLAR

[1] Valenzula J., Smith A.E. (2002). A Seeded Memetic Algorithm for Large Unit Commitment Problems. Journal of Heuristics, 8:173-195.

[2] Padhy,N.P. (2004). Unit Commitment - A Bibliographical Survey. IEEE Transactions On Power Systems, 19(2), 1196-1205.

[3] Kazarlis S.A.,Bakirtzis A.G. & Petridis V. (1996). A genetic algorithm solution to the unit commitment problem. IEEE Transactions On Power Systems, 11(1), 83-92.

[4] Maturana J. & Riff M.C. (2007). Solving the short-term electrical generation scheduling problem by an adaptive evolutionary approach. European Journal of Operational Research 179, 677–691.

[5] Saramourtsis A., Damousis J., Bakirtzis A. & Dokopoulos P. (1996). Genetic algorithm solution to the economic dispatch problem – application to the electrical power grid of crete island. Greece: Aristotle University of Thessaloniki.

[6] Beyer H.G., Schwefel H.P. (2002). *Evolution strategies.* Natural Computing 1: sy. 3–52.

[7] Eiben A. E., Schoenauer M. (2005). *Evolutionary Computing.*

[8] Kuş M.C. (2006). *Interactive Evolutionary Strategies for Automatic Facial Composite Generation.* İstanbul Teknik Universitesi, İstanbul, Türkiye.

[9] Uyar, Ş. (2006). *Nature-Inspired Compting Evolutionary Strategies.* Erişim: 10 Şubat 2007, http://www3.itu.edu.tr/~etaner/courses/NIC0607/ES_new.pdf

[10] Beasley D. (t.y.). *What's an Evolution Strategy (ES)?* Erişim: 10 Şubat 2007, http://www.faqs.org/faqs/ai-faq/genetic/part2/section-4.html

[11] Vikipedi, Özgür Ansikloopedi. (t.y.). Erişim: 15 Mayıs 2007, http://tr.wikipedia.org/wiki/

[12] Kurt M., Semetay C. (t.y.). *Genetik Algoritma ve Uygulama Alanları.* Erişim: 10 Şubat 2007, http://www.mmo.org.tr/muhendismakina/arsiv/2001/ekim/Genetik_Algoritma.htm