İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÜNİTE YÜKLENME PROBLEMİNİN PARÇACIK SÜRÜ OPTİMİZASYON YÖNTEMİ İLE ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

Yasemin ZEYBEKOĞLU

Anabilim Dalı : Elektrik Mühendisliği

Programı : Elektrik Mühendisliği

HAZİRAN 2011

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÜNİTE YÜKLENME PROBLEMİNİN PARÇACIK SÜRÜ OPTİMİZASYON YÖNTEMİ İLE ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ
Yasemin ZEYBEKOĞLU
(504081033)

Tezin Enstitüye Verildiği Tarih : 06 Mayıs 2011
Tezin Savunulduğu Tarih : 10 Haziran 2011

Tez Danışmanı : Doç. Dr. Belgin TÜRKAY (İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. Ayşen DEMİRÖREN (İTÜ)
Prof. Dr. İbrahim EKSİN (İTÜ)

HAZİRAN 2011

Aileme,

ÖNSÖZ

Tez çalışmalarım boyunca her türlü destek ve yardımlarından dolayı danışmanım Doç. Dr. Belgin Türkay'a teşekkür ederim.

Tezin oluşması ve hazırlanması aşamasında faydalandığım internet kaynaklarını hazırlayanlara ve paylaşanlara, emeklerinden ve paylaşımlarından ötürü teşekkür ederim.

Yüksek lisans çalışmalarım boyunca gösterdikleri sabır ve anlayıştan ötürü çalışma arkadaşlarıma ve müdürlerime teşekkürü borç bilirim.

Hayat boyu bütün imkânlarını çocukları için seferber eden, Anneme ve Babama şükranlarımı sunarım.

Tezimin kontrolü sırasındaki yardımlarından ötürü sevgili kardeşim Mehmet Can Zeybekoğlu'na teşekkür ederim.

Manevi desteğini hiçbir zaman esirgemeyen, tez çalışmam sırasında sık sık fikrini aldığım Coşkun Baygar'a sonsuz teşekkürler.

Haziran 2011

Yasemin Zeybekoğlu

(Elektrik Mühendisi)

İÇİNDEKİLER

Sayfa

KISALTMALAR

PSO : Parçacık Sürü Optimizasyonu
APSO : Ayrık Parçacık Sürü Optimizasyonu
ÖLY : Öncelik Listesi Yöntemi
DP : Dinamik Programlama
GLM : Gevşetilmiş Lagrange Yöntemi
DSA : Dal Sınır Algoritması
GA : Genetik Algoritma
EP : Evrimsel Programlama
BT : Benzetimli Tavlama
KKO : Karınca Kolonisi Optimizasyonu
ÜYP : Ünite Yüklenme Problemi
KTP : Karmaşık Tamsayılı Programlama
KP : Kuadratik Programlama
LİY : Lambda İterasyonu Yöntemi
İPSO : İyileştirilmiş Parçacık Sürü Optimizasyonu
HPSO : Hibrit Parçacık Sürü Optimizasyonu
KTPLO : Karmaşık Tamsayı Programlamalı Lineer Olmayan
Lbest : Lokal En İyili
US : Uyum Sağlayabilen
EPSO : Evrimsel Parçacık Sürü Optimizasyonu
ARAA : Açgözlü Rastgele Adapte Olabilen Arama
MA : Memetik Algoritma
AFEA : Ayrık Farksal Evrim Algoritması
ES : Evrimsel Stratejiler
DDGA : Durgun Durum Genetik Algoritma
TKGA : Tamsayı Kodlanmış Genetik Algoritma

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

ÜNİTE YÜKLENME PROBLEMİNİN PARÇACIK SÜRÜ OPTİMİZASYON YÖNTEMİ İLE ÇÖZÜMÜ

ÖZET

Tüketicilerin elektrik enerjisi ihtiyacı günün farklı saatlerinde değişim göstermekte, sistem işletmecilerinin de günlük enerji üretimini değişen talebe göre planlamaları gerekmektedir. Enerji ihtiyacı genellikle öğleden önce ve öğleden sonraki saatlerde endüstriyel yüklerin devrede olması ve işyerlerinin açık olması nedeniyle yüksektir. Ticari ve endüstriyel kuruluşların kapalı olduğu geceleri ve gündüz erken saatlerde ise enerji talebi oldukça düşüktür.

Değişen taleplere göre planlama yapılırken, ünitelerin toplam yakıt maliyeti minimum olacak şekilde kombinasyonlarla devreye alınması işletmeciler için en önemli hususlardan biridir. Talep edilen enerjiyi ekonomik olduğu kadar güvenli bir şekilde sağlayabilmek için en uygun çözümlerden biri ise termal ünitelerin devreye alınmasıdır.

Ünite yüklenme problemi, üretim ünitelerinin fiziksel kısıtları ve sistemin kısıtları dâhilinde gerekli ünitelerin minimum maliyetle devreye alınmasının planlanmasıdır. Ünite üretim maliyeti, yakıt maliyeti, devreye alma maliyeti ve bakım maliyetlerinin toplamından oluşur. Kısıtlar ise talep edilen güç, rezerv güç, ünitelerin minimum devrede, devre dışı olma süreleri ve ünitelerin üretebilecekleri güç sınırlarından oluşur. Ünite yüklenme probleminin, karmaşık kombinasyonel ve sürekli bir problem olması çözümü zorlaştırmaktadır. Lineer olmayan amaç fonksiyonu ve sağlanması gereken kısıtların çokluğu problemin çözümünü daha da zorlaştırır.

Tez kapsamında termal ünite yüklenme problemi ele alınmıştır. Problem formüllere dökülmüş ve popülasyon temelli bir arama ve optimizasyon yöntemi olan Parçacık Sürü Optimizasyon yöntemi ile ünite yüklenme problemi çözümü için bir algoritma geliştirilmiştir.

İki alt problemden oluşan ünite yüklenme problemi için zincirleme bir Parçacık Sürü Optimizasyon yöntemi uygulanmıştır. İlk alt problem olan ünitelerin devreye alınma ve devre dışı bırakılma planlaması için Ayrık Parçacık Sürü Optimizasyon yöntemi kullanılmıştır. İkinci alt problem olan devreye alınmış ünitelerin optimal yüklenmelerinin belirlenmesi için ise klasik bir Parçacık Sürü Optimizasyonu yöntemi geliştirilmiştir. Algoritmanın başarısı, 8 ünite ile 40 ünite arasında değişen büyüklükte sistemlerde test edilmiş, hesaplama süreleri verilerek sonuçlar üretim maliyeti açısından literatürdeki çalışmalarla karşılaştırılmıştır.

SOLVING UNIT COMMITMENT PROBLEM BY PARTICLE SWARM OPTIMIZATION TECHNIQUE

SUMMARY

As the power demand varies in different periods of a day, power generation companies need to plan the operation periods of the power units accordingly. The power demand is especially high during the daytime, since the industrial and commercial facilities consumes most of the generated electricity in that period. However, demand decreases significantly during early morning or late evening when industrial and commercial facilities are out of working hours.

An important criterion in power system operation is to meet the demand at minimum fuel cost using an optimal mix of different power units. Relaible supply of demaned load is an other important criteria in power systems which is solved by using thermal power units.

Unit Commitment Problem is the problem of determining the generation schedule of generating units subject to the device and operational constraints. The production costs include fuel, startup and maintanence costs. The constraints are capacity reserve, minimum up/down time and operating limits of power units. Unit commitment problem is a mixed combinatorial and continuous optimization problem, which is very complex to solve because of its enormous dimension, a non-linear objective function and large number of constraints.

The thermal unit commitment problem has been discussed in the thesis work. The formulation of unit problem has been discussed and the solution is obtained by a Particle Swarm Optimization which is a population based global search and optimization technique. An Enchained Particle Swarm Optimization technique has been developed to solve the unit commitment problem.

A Binary Particle Swarm Optimization Algorithm is developed for first subproblem, determining the on-off scheduls of power units. Moreover, a conventional Particle Swarm Optimization algorithm is developed to solve the second subproblem, determining the optimal loading of commited power units. The success of the algorithm has been tested on systems containing 8 units to 40 units. Computation times for each test system are given and production cost results are compared with other algorithms in literature.

1. GİRİŞ

1.1 Giriş

Güç sistemlerinin başarılı bir şekilde işletilmesi için göz önünde bulundurulması gereken çok sayıda faktör vardır. Tüketiciler ne zaman, ne kadar güce ihtiyaç duyarlarsa duysunlar, güç sisteminin bu ihtiyacı derhal ve sürekli olarak karşılaması beklenir. Bu esnada verilen gücün gerilim ve frekansının da nominal değerler civarında olması gerekir [1].

Tüketicilerin ihtiyaçlarını karşılamak için çalıştırılan sistemlerin güvenli olması, halka ve çalışanlara zarar vermemesi için gereken önlemler alınmalıdır. Uygun çalışma prosedürleri takip edilmeli ve sistemin içindeki donanıma ve diğer tesislere de zarar verilmemelidir.

Tüm bu üretim beklentileri eşzamanlı olarak karşılanırken üretimin ve dağıtımın mümkün olan en az maliyetle sürdürülmesi istenir. Bununla birlikte gelecekte yapılacak yatırımlar ve planlamalar da göz önünde bulundurulmalıdır [1].

Güç sistemlerinde birbirlerine alternatif olan, enerji üretimini farklı kaynaklardan sağlayan üretim üniteleri mevcuttur. Bunlar klasik üretim üniteleri olan buharlı elektrik santralleri, nükleer santraller, hidroelektrik santraller, jeotermal santraller, gaz türbinleri ve alternatif üretim üniteleri olarak adlandırılan güneş santralleri, rüzgâr santralleri ve yakıt pili enerji santralleridir [1].

Ünite yüklenme problemi, denetim altındaki bir güç sisteminde, talep güç karşılanırken üretim ünitelerinin maliyetlerinin optimizasyonu olarak tanımlanır. Amaç fonksiyonunun üretim maliyeti fonksiyonu olarak tanımlandığı bu tip ünite yüklenme problemleri, maliyet temelli ünite yüklenme problemi olarak adlandırılır. Eğer maliyet temelli ünite yüklenme problemi dâhilinde hedeflenen aynı zamanda şebekenin güvenli işletilmesi ise, problem maliyet-güvenlik temelli ünite yüklenme problemi olarak isimlendirilir. Bu tip ünite yüklenme probleminin üç amacı vardır: talep gücün karşılanması, güvenlik kısıtlarının sağlanması ve maliyetin minimize edilmesi [2].

Güvenli üretimin sağlanması, temel olarak yeterli rezerv gücün sağlanması ile realize edilebilir. Maliyet optimizasyonu ise temel olarak üretim maliyeti daha az olan ünitenin devreye alınması ile sağlanır. Birçok elektrik piyasasında maliyet-güvenlik temelli ünite yüklenme, günlük üretimin planlanmasında kullanılır [2].

Uzun yıllardır enerji endüstrisinde ünite yüklenme probleminin çözümü için çeşitli sayısal optimizasyon yöntemleri kullanılmıştır. Bu sayede yakıt maliyetlerinde onlarca belki de milyonlarca liralık tasarruf sağlanmıştır. Buna rağmen bu yıllar içinde birçok şey de değişmiştir [3].

Farklı ülkelerin elektrik şebekelerinin birbirlerine bağlanarak, enterkonnekte olarak çalışmaya başlaması, dünya enerji krizi ve artan yakıt maliyetleri, tüketicilerin ihtiyacı karşılanmaya devam ederken, yakıt maliyetlerinin optimizasyonunun önemini arttırmıştır [4].

Ünite yüklenme problemini çözmek ve yakıt maliyetinden tasarruf sağlamak amacıyla geliştirilen yöntemler son 20 sene içinde artış göstermiştir. Eğer bir yöntem, belirli bir probleme her uygulanışında aynı sonucu veriyorsa bu yönteme deterministik yöntem denir. Deterministik yöntemler genellikle en iyi bir tek çözümü bulmak için kullanılırlar. Öncelik listesi yöntemi(ÖLY), dinamik programlama(DP), gevşetilmiş Lagrange yöntemi(GLY) ve dal sınır algoritması(DSA) ünite yüklenme problemine uygulanmış, deterministik yöntemlerdir.

Az üniteli sistemlerde optimum çözümü sunabilen deterministik yöntemlerin çok üniteli sistemlerde aynı başarıyı gösteremedikleri görülmüştür. Yöntemlerin bir kısmının, çok üniteli sistemlerdeki atamalarda optimumdan uzak çözümler sunduğu, bir kısmının ise hesaplama süresinin uzunluğu nedeniyle kullanışsız olduğu anlaşılmıştır. Enerji sistemlerinin gün geçtikçe büyümesi, çok üniteli sistemleri çözebilen yöntemlerin geliştirilmesi ihtiyacını doğurmuştur. Stokastik yöntemler ise aynı problemler için farklı sonuçlar verebilir. Fakat bu yöntemler en iyisi olmasa da kabul edilebilir sonuçlara, belirlenen süre içinde ulaşabilmektedir. Bu ihtiyaç sonucunda, parçacık sürü optimizasyonu(PSO), genetik algoritma(GA), evrimsel programlama(EP), benzetimli tavlama(BT), karınca kolonisi optimizasyonu(KKO) ve tabu araması(TA) gibi stokastik yöntemler, ünite yüklenme problemlerine uygulanmaya başlamıştır. Bu yöntemler, kompleks lineer olmayan kısıtlara rağmen çok üniteli ünite yüklenme problemlerinde başarılı sonuçlar verebilmektedir.

Tez kapsamında, kuşların ve balıkların yem arama davranışlarından esinlenerek geliştirilmiş bir stokastik yöntem olan parçacık sürü optimizasyonuyla(PSO) ünite yüklenme probleminin çözümü için bir algoritma geliştirilmiştir. Bu algoritmayla elde edilen sonuçlar, literatürdeki ünite yüklenme problemlerine(ÜYP) uygulanmış ve diğer yöntemlerin literatürde yer alan sonuçlarıyla karşılaştırılmıştır.

1.2 Literatürdeki Çalışmalar

Yapılan çalışma kapsamında parçacık sürü optimizasyonu(PSO) kullanılarak ünite yüklenme problemi(ÜYP) çözülmüştür. Literatürde ünite yüklenme probleminin stokastik yöntemlerle çözüldüğü birçok çalışma mevcuttur. Bu yöntemler içinde en yeni olan ve en az uygulanmış olan yöntem PSO'dur. Bu nedenle literatür çalışmasının ilk kısmında, oldukça eski bir problem olan ÜYP'ne hangi yöntemlerle çözüm arandığı ve bu yöntemlerin avantaj-dezavantajlarından kısaca bahsedilmiştir. Ardından ise, ÜYP'lerine uygulanmış PSO yöntemlerinin yer aldığı çalışmalar özetlenmiştir.

Öncelik listesi yöntemi(ÖLY), ünite yüklenme probleminin çözümünde en çok kullanılan yöntemlerdendir. Uygulaması kolay ve hızlı sonuç veren bir yöntem olmasına rağmen, özellikle çok üniteli sistemlerde çözüm kalitesini garanti etmediği yapılan çalışmalarda görülmektedir [5].

Dinamik programlama(DP), birçok optimizasyon problemine uygulanan ve sıkça kullanılan bir diğer yöntemdir. Küçük sistemlerde sonuç kalitesi yüksek olan bu yöntem, çok kısıtlı ve çok üniteli sistemlerin çözümünde, hesaplama zamanı nedeniyle dezavantajlıdır [6].

Dal-sınır algoritması(DSA) da dinamik programlama gibi, kısıtlar ve ünitelerin arttığı sistemlerde, uzun hesaplama süresine ihtiyaç duymakta, bu da yöntemin verimini düşürmektedir [7].

Tamsayı ve karmaşık tamsayı programlama yöntemi(KTP), lineer programlama yönteminin tamsayı sonuçlu problemlerin çözümü için geliştirilmiş bir versiyonudur. Bu yöntem de, büyük sistemlerin çözümünde ihtiyaç duyduğu hafızanın büyüklüğü ve hesaplama süresinin üssel olarak artması sebebiyle büyük sistemler için kullanışsız hale gelmektedir.

Juste, her boyuttaki ÜYP'ni optimize edebilecek bir yöntem geliştirme amacıyla evrimsel programlama(EP) kullanmıştır. Rastgelelik, seçim, mutasyon gibi EP yöntemlerini kullandığı algoritmayı en büyüğü 100 üniteli olan test sistemlerinde denemiş ve sonuçları gevşetilmiş Lagrange yöntemi(GLM), dinamik programlama(DP), ve genetik algoritma(GA) yöntemlerinin sonuçlarıyla karşılaştırmıştır [8].

Mantawy, ÜYP'ni bir benzetimli tavlama(BT) algoritması geliştirerek çözmüştür. İki alt probleme ayırdığı ÜYP'nin ünite devreye alma kısmını BT ile ekonomik yük dağıtımı kısmını ise kuadratik programlama(KP) ile çözmüştür [9].

Sum-im, termal ÜYP çözümü için bir karınca kolonisi optimizasyonu(KKO) algoritması geliştirmiştir. Sürü zekâsını kullandığı bu yöntemi 10 üniteli sistemde test etmiş ve sonuçlarını, gevşetilmiş Lagrange yöntemi(GLY), genetik algoritma(GA), evrimsel programlama(EP) ve gevşetilmiş Lagrange yöntemli (GLY) genetik algoritma(GA) yöntemlerinden elde edilmiş sonuçlarla karşılaştırmıştır [10].

Sheble, genetik algoritma(GA) temelli bir ünite yüklenme algoritması geliştirmiştir. Alana özgü mutasyon operatörlerini GA'ya dâhil ederek, daha iyi sonuçlar bulmayı hedeflemiştir. 3 farklı test sistemi için elde ettiği sonuçları, gevşetilmiş Lagrange yöntemi(GLY) sonuçlarıyla karşılaştırmıştır [11]

Bakirtsiz, ÜYP'ni, farklı kalite fonksiyonları kullanarak geliştirdiği GA ile çözmüştür. 100 üniteye kadar genişlettiği test sisteminden elde ettiği sonuçları, dinamik programlama ve GLY'nden elde edilen sonuçlarla karşılaştırmıştır [12].

Swarup, GA'yı kullanarak ÜYP için yeni bir çözüm yöntemi geliştirmiştir. Bir hayli etkili olan bu yöntem, büyük boyutlu ÜYP'ni de çözümünde kullanılabilmektedir [13].

Sriyanyong, ÜYP için PSO ile gevşetilmiş Lagrange yöntemini kullanarak hibrit bir algoritma geliştirmiştir. Algoritmada ÜYP çözümünde kullandığı Lagrange çarpanlarını, PSO kullanarak iyileştirmiştir. Yöntemi 4 ve 10 üniteli sistemlerde test etmiştir [14].

Zwe-Lee Gaing, ÜYP çözümü için lambda iterasyonu yöntemini(LİY) ayrık parçacık sürü optimizasyonu(APSO)'na entegre ederek yeni bir yöntem geliştirmiştir. APSO yöntemiyle ilk alt problem olan ünite durumlarını belirlemiştir. Lambda iterasyonunu kullanarak ise devreye alınan ünitelerin toplam maliyetini minimize etmiştir.

Yöntemi 10 ve 26 üniteli sistemlerde test etmiş ve sonuçları GA yöntemi sonuçlarıyla karşılaştırmıştır [15].

Zhao, ÜYP için iyileştirilmiş PSO(İPSO) algoritmasını geliştirmiştir. Parçacıkları arama uzayında homojen olarak dağıtabilmek için geometrik bir dağıtım yöntemi uygulamıştır. İPSO algoritması 10 üniteli test sistemine uygulanmış ve sonuçlar GA ve EP yöntemlerinin sonuçlarıyla karşılaştırılmıştır [16].

Ting, yeni bir yaklaşım getirerek, ÜYP çözümü için, APSO ve klasik PSO yöntemlerini harmanladığı yeni bir HPSO yöntemi geliştirmiştir. Ünite durumlarının belirlenmesini APSO, ekonomik yük dağıtımını ise klasik PSO yöntemiyle çözmüştür [17].

Victoire, HPSO ve ardışık kuadratik programlama(KP) yöntemlerini birleştirerek tabu araması(TA)'na giriş yapan bir ÜYP çözüm yöntemi geliştirmiştir. ÜYP'nin kombinasyonel kısmını TA ile ekonomik yük dağıtımı kısmını ise HPSO-AKP tekniğiyle çözmüştür. Geliştirdiği HPSO yöntemini, 7 üniteli sistemde test etmiştir [18].

1.3 Tezin Amacı

Tezin amacı, termal ünite yüklenme problemini kısa sürede ve minimum maliyetle çözecek bir yöntem geliştirmektir. Bu amaçla, ünite yüklenme probleminin çözümü için ayrık parçacık sürü optimizasyonu(APSO) ve klasik parçacık sürü optimizasyonu(PSO) kullanılarak bir algoritma geliştirilmiştir. APSO ile ünitelerin devrede-devre dışı olma durumları belirlenmiş, PSO ile ise ekonomik yük dağıtımı çözülmüştür. Bu algoritma MATLAB ortamında kodlanmıştır. Literatürdeki çalışmalarda sıklıkla kullanılan test sistemlerinde, algoritma test edilmiştir. Test sonuçları için hesaplama süreleri verilmiş ve diğer yöntemlerde bulunan maliyet sonuçlarıyla karşılaştırılmıştır.

1.4 Tezin Yapısı

1.Bölümde, literatürde yapılan çalışmalar anlatılmış, tezin amacı ve yapısından bahsedilmiştir.

2.Bölümde, ünite yüklenme probleminin(ÜYP) tanımı yapılmıştır. Termal ünite yüklenme problemi için dikkate alınması gereken kısıtlar ve maliyetler formülleriyle birlikte anlatılmıştır.

3. Bölümde, parçacık sürü optimizasyonu(PSO)'nun klasik versiyonu detaylı olarak anlatılmıştır. Geliştirilmiş PSO versiyonlarının yer aldığı çalışmalar ve klasik versiyondan ayrılan kısımları özetlenmiştir.

4. Bölümde, ünite yüklenme probleminin PSO ile çözülmesi içi oluşturulmuş algoritma ve bu algoritmanın detaylı çalışma prosedürü açıklanmıştır.

5. Bölümde, ünite yüklenme problemini konu edinen çalışmalarda sıklıkla kullanılan 10, 20, 40 üniteli sistemler ve Türkiye şebekesinden alınmış 8 üniteli alt sistem test sistemi olarak ele alınıp çalışmanın sonuçları incelenmiştir. Elde edilen sonuçların, literatürdeki farklı algoritmalarla bulunan sonuçlarla maliyet, hesaplama süresi ve yakınsama hızı açısından karşılaştırılmasına yer verilmiştir.

6. ve son bölümde ise sonuçlar yorumlanmış ve çalışmanın nasıl geliştirilebileceği üzerinde durulmuştur.

2. BİRİM YÜKLENME PROBLEMİ

2.1 Giriş

Ünite yüklenme, ön dağıtım olarak da adlandırılan bir operasyon planlama problemidir. Ünite yüklenme ile üretim ünitelerinin devrede ve devre dışı olacağı zamanlar belirlenerek, güç sistemi kısıtları dâhilinde üretim maliyetleri optimize edilir [19].

Güç sistemlerinin enerji ihtiyacı yıllar içinde arttığı gibi günden güne ve saatten saate de değişiklik göstermektedir. Enerji ihtiyacı genellikle öğleden önce ve öğleden sonraki saatlerde daha yüksektir. Endüstriyel yüklerin bu saatlerde devrede olması, işyerlerinin aydınlatmaları ve tarihi mekânların aydınlatmaları bu saatlerde duyulan enerji ihtiyacının yüksek olmasının temel sebepleridir. Ticari ve endüstriyel kuruluşların kapalı olduğu geceleri ve gündüz erken saatlerde ise enerji ihtiyacı oldukça düşüktür. Aynı nedenlerden ötürü iş günlerinde enerji ihtiyacı yüksekken tatil günlerinde daha düşüktür. Bununla birlikte Hindistan gibi sıcak ülkelerde yazları ortaya çıkan, elektrikli soğutucuların kullanımından kaynaklanan, enerji ihtiyacının diğer mevsimlere göre oldukça yüksek olması gibi coğrafyaya özgü durumlar da vardır [20].

Mevcut bütün üretim ünitelerinin her daim devre tutulması da değişen enerji talebinin karşılanmasını sağlamanın yollarından biri olmasına rağmen, pek ekonomik bir çözüm değildir. Bu durumda enerji ihtiyacının düşük olduğu saatlerde devredeki kimi ünitelerin üretime pek katkısı olmazken, bu üniteleri devrede tutmak için harcanan yakıt ise masraflı olmaktadır. Kimi ünitelere ihtiyaç olmadığında devreden çıkarmak ise yakıt maliyetlerinde tasarruf sağlamaktadır. Ünite yüklenme bu nedenle, gerekli sayıda üniteyi devrede tutarak mevcut ihtiyacın karşılanmasını sağlarken ani ihtiyaç değişikliklerine cevap verebilecek güç rezervini de göz önünde bulundurarak enerji üretiminin sürekli, güvenilir ve ekonomik olarak sağlanmasını hedefler [21].

Yakıt maliyetlerinin gün geçtikçe artması, enerji sistemlerinin ekonomik olarak işletilmesinin önemini arttırmaktadır. Türkiye gibi büyük üretim kapasitesine sahip bir ülkenin, ekonomik ünite tahsisiyle senelik yakıt maliyetini %1 oranında düşürmesi, milyonlarca liranın tasarruf edilmesi anlamına gelir. Bu nedenlerden ötürü ünite yüklenme problemi, güç sistemlerinin optimizasyonundaki en önemli problemlerden biridir.

Bu bölümün 1. kısmında, ünite yüklenme probleminin maliyetlerinden bahsedilmiştir. 2. kısımda ise sistemin güvenli olarak işletilmesi için dikkate alınması gereken önemli kısıtlar anlatılmıştır.

2.2 Problem Maliyetleri

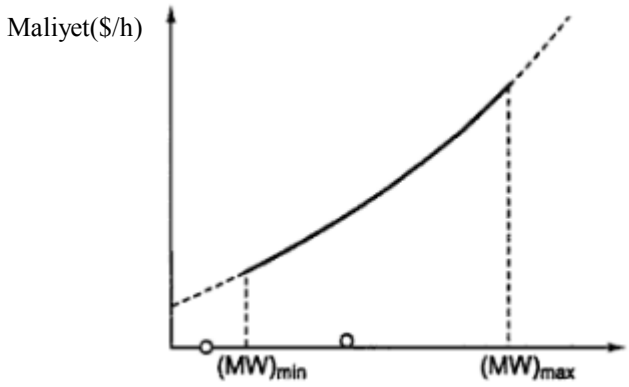
Bir ünitenin işletme maliyeti; yakıt maliyeti, personel maliyeti ve bakım maliyetlerinin toplamıdır. Ünitenin yakıtı ne olursa olsun (kömür, doğalgaz, nükleer vb.), yakıt maliyeti toplam işletme maliyetine en çok katkısı olan kalemdir. Diğer maliyetleri belirlemek zor olduğu için ve optimizasyon problemlerinin çözümünde kolaylık sağlaması açısından personel ve bakım maliyetleri yakıt maliyetinin içinde dikkate alınır [22]. Yakıt maliyeti, termal santraller için anlamlı olmasına rağmen hidroelektrik santraller, rüzgâr santralleri vb. yakıtı neredeyse bedava olan santraller için elbette ki anlamsızdır. Tezin kapsamında ÜYP'nin öneminin en büyük olduğu, yakıtı en maliyetli olan termal santraller ele alınacağı için bu kısımda, termal santraller için söz konusu olan maliyetlere yoğunlaşılacaktır [23].

2.2.1 Yakıt maliyeti

Üretim ünitesinin tüketim-üretim eğrisi, saatlik enerji tüketimi veya tüketilen yakıtın maliyetine karşılık üretilen güçten oluşur. Bu eğri ancak deneysel olarak elde edilebilir. Örnek bir tüketim-üretim eğrisi Şekil 2.1de görülmektedir. Yakıt birim maliyeti, aylık ve hatta günlük olarak değişebildiği için tüketim-üretim eğrisinde genellikle yakıt maliyeti $F_i(P_i)$ yerine, tüketilen yakıt miktarı $T_i(P_i)$ kullanılır [24].

Tüketim-üretim eğrisinden, üretilen güce göre tüketilen yakıt miktarı kuadratik olarak **(2.1)** denklemiyle ifade edilir.

$$T_i(P_i) = a_i' + b_i' \cdot P_i + c_i' \cdot P_i^2 \qquad \textbf{(2.1)}$$



Şekil 2.1 : Yakıt tüketimi-enerji üretimi eğrisi [4].

Tüketilen yakıt miktarını, yakıt ünite maliyetiyle çarparak da toplam yakıt maliyeti F_i bulunur. Yakıt ünite maliyetinin tüketilen yakıt miktarı denkleminin katsayılarıyla çarpılmasıyla, yakıt maliyeti denklemi $F_i(P_i)$ **(2.2)** elde edilir.

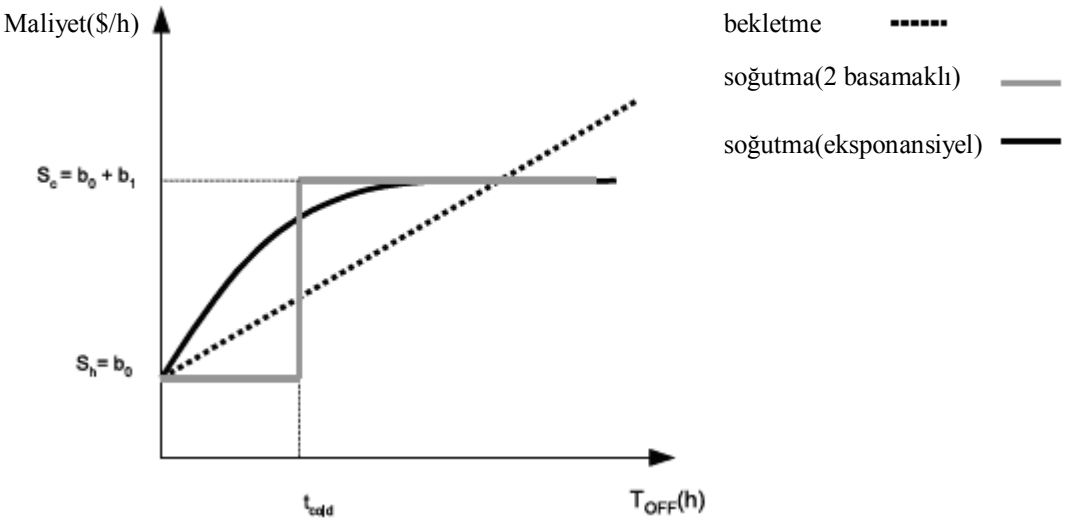
$$F_i(P_i) = a_i + b_i \cdot P_i + c_i \cdot P_i^2 \qquad \textbf{(2.2)}$$

2.2.2 Devreye giriş maliyeti

Termal bir üretim ünitesi, santralin verimliliği göz önünde bulundurulduğunda, ancak ve ancak buhar sıcaklığı ve basıncı uygun değerlere ulaştığı zaman devreye alınarak şebekeye bağlanır. Ünite devreye girdikten sonra, bu uygun değerler sağlanarak şebekeye bağlanma koşullarına ulaşıncaya kadar elektrik üretmese de, yakıt tüketmektedir. Şebekeye bağlanma koşullarını sağlamaya yönelik tüketilen bu yakıt maliyeti devreye giriş maliyeti olarak adlandırılır [4].

Devreye giriş maliyeti, sabit bir değer değil, ünitenin o anki sıcaklık ve basınç değerlerine göre değişebilen bir maliyettir. Ünite belirli bir süredir soğuk durumdaysa, devreye giriş maliyeti maksimum değerindedir. Buna rağmen ünite kısa süre önce devreden çıkarılmış ve basınç ile sıcaklık değerleri devredeki değerlerine yakın ise devreye giriş maliyeti çok daha azdır. Ünitenin soğuk olarak adlandırıldığı ilk durumdaki başlangıç maliyetine "soğuk başlangıç maliyeti", ikinci durumdaki başlangıç maliyetine ise "sıcak başlangıç maliyeti" denir [4].

Başlangıç maliyetinin belirlenmesi, ünitelerin boşta bekletilme yöntemlerine göre değişiklik gösterir. Bu yöntemlerden ilki, ünite devreden çıkarıldıktan sonra soğumaya bırakıldığı durumdur. Bu durumda, üniteyi yeniden devreye almak için yeterli sıcaklığa kadar ısıtılması gerekir.



Şekil 2.2 : Devreye giriş maliyeti [4].

Ünitenin ısıtılması esnasında harcanan yakıt maliyeti **(2.3)** formülüyle tanımlanır. Formülde b_0, sabit başlangıç maliyeti, b_1 ise soğuk başlangıç maliyetidir. Şekil 2.2'deki τ, soğuma sabiti, OFF ise ünitenin kapalı kaldığı toplam saat olarak tanımlanır.

$$S_x = b_1 \cdot (1 - e^{(-\frac{OFF}{\tau})}) + b_0 \qquad \textbf{(2.3)}$$

Literatürdeki çalışmalarda, bu exponansiyel maliyet eğrisinin, sıcak başlangıç maliyeti ve soğuk başlangıç maliyeti olarak 2 basamaklı bir fonksiyon olacak şekilde modifiye edildiği görülmüştür. Bu fonksiyon **(2.4)** ile tanımlanır. S_h ünitenin sıcak olduğu haldeki başlatma maliyeti, S_c ünitenin soğuk olması durumundaki başlangıç maliyeti, t_{cold} ise ünite kazanının tamamen soğuması için gerekli olan süredir.

$$S_x = \begin{cases} S_h & eğer \quad OFF \le t_{cold} \\ S_c & aksi takdirde \end{cases} \qquad \textbf{(2.4)}$$

Kısa süreli olarak devre dışı bırakılacak üniteler genellikle soğumaya bırakılmaz bunun yerine üretim yapmamalarına rağmen sıcak tutulmaya devam edilirler. Bu durumda, başlangıç maliyeti ünitenin sıcak tutulması için harcanan yakıtın maliyetiyle tanımlanır ve **(2.5)** ile hesaplanır [25].

$$S_x = b_1 \cdot OFF + b_0 \qquad \textbf{(2.5)}$$

2.2.3 Devreden çıkış maliyeti

Ünite kapama maliyeti genellikle sabit bir maliyet olarak tanımlanır.[26]. Üniteyi devreden çıkarılması için harcanan personel maliyeti olarak tanımlanır ve başlangıç maliyetinin yanında ihmal edilebilecek kadar azdır.

2.3 Problem Kısıtları

Ünite yüklenme problemi çok fazla kısıta sahip bir optimizasyon problemidir. Farklı güç sistemlerinin, maruz kaldıkları kısıtlar bütünü birbirinden farklıdır. En genel tanımıyla termal üniteli sistemlerde göz önünde bulundurulması gereken kısıtlar 5 başlık altında açıklanmıştır.

2.3.1 Üretim limiti

Devreye alınmış olan ünite, devredeyken üretebileceği minimum güç olan P_{mini} ile üretebileceği maksimum güç olan P_{maxi} arasında yüklenebilir **(2.6)**.

$$P_{min_i} \leq P_i \leq P_{max_i} \tag{2.6}$$

2.3.2 Güç dengesi

Ünitelerin ürettikleri güçlerin toplamı, talep gücü ve hatlarda meydana gelen kayıp gücü karşılamak zorunda olup. **(2.7)** ile tanımlanır. P_{load} talep gücünü, P_i ünite tarafından üretilen gücü, P_{losses} ise sistemde meydana gelen kayıp gücü tanımlar.

$$P_{load} = \sum_{i=1}^{NG} P_i - P_{losses} \tag{2.7}$$

2.3.3 Güç rezervi

Güç rezervi, devredeki şebeke ile senkronize yani $u_i = 1$ olan ünitelerin, üretebilecekleri maksimum enerji miktarından, şebekenin talep ettiği güç ve kayıp güçler çıkarıldıktan sonra geriye kalan emre amade güç, $P_{reserve}$ olarak **(2.8)** ile tanımlanır [27].

$$\sum_{i=1}^{NG} P_{maxi} \cdot u_i \geq P_{load} + P_{reserve} + P_{losses} \tag{2.8}$$

Enterkonnekte şebekelerde, ünite veya şebeke kaynaklı arızalar vb. sebeplerle ünitelerden herhangi birinin devreden çıkması durumunda talep edilen gücün kesintisiz olarak karşılanmaya devam edilmesini sağlayacak olan güç rezervini hazırda bulundurmak kritik bir öneme sahiptir. Rezervde yeterli gücü bulunan sistemlerde, ünitelerden biri veya birkaçı devreden çıksa bile devrede olan ünitelerle talep karşılanmaya devam edilir.

Güç rezervinin hangi ünitelerden, ne oranda karşılanacağı kararı, sistemin güvenilirliğini gözeterek ekonomik sebeplerle verilir. Kimi işletmelerde rezerv güç miktarı, güç talebinin belirli bir yüzdesi kadar, kimi işletmelerde ise devredeki en büyük kapasiteli ünitenin üretim kapasitesi kadar tahsis edilir.

Ünitelerden herhangi birinin devreden çıkması gibi problemlerin yanı sıra, iletim sisteminden kaynaklanan problemlerde de güç talebinin karşılanmaya devam edilebilmesi için devredeki ünitelerin şebeke içinde homojen olarak yerleştirilmesi de gerekir. Buna ek olarak şebekeden kimi bölgelerin izole edilmesi ve ada çalışma durumlarında da güç rezervinin yeterli olmasına dikkat edilmelidir. Yani rezerv gücün ünite başına ve toplam dağılımı kadar, dağıldığı ünitelerin konumları da önemlidir [4].

2.3.4 Devreye girme ve devreden çıkma süreleri

Devredeki bir ünitenin istenilen herhangi bir anda aniden devreden çıkarılması mümkün değildir, devreye alındıktan sonra ancak belirli bir süre geçti ise devreden çıkarılabilmektedir. Bu süre, ünitenin minimum devrede kalma süresi, t_{up} olarak adlandırılır. Benzer şekilde eğer bir ünite devreden çıkarıldıysa da, ünitenin devreye alınabilmesi ve şebekeye bağlanabilmesi, gerekli koşulların sağlanabilmesi için belirli bir süreye ihtiyaç vardır. Bu süre de, ünitenin minimum devre dışı kalma süresi, t_{down} olarak adlandırılır [4].Ünitenin açık ve kapalı kaldığı toplam süreler göz önünde bulundurularak devrede olma veya devre dışı olma zorunluluğu **(2.9)** ile belirlenir. u_i, ünitenin durumunu tanımlar. u_i'nin 1 olması ünitenin devrede, 0 olması ise devre dışı olduğu anlamına gelir.

$$u_i = \begin{cases} 1 & eğer \quad 1 \le t_{on} \le t_{up} \\ 0 & eğer \quad 1 \le t_{off} \le t_{down} \end{cases} \qquad \textbf{(2.9)}$$

Açma kapama sürelerini belirleyen bir diğer kısıt, termik santrallerdeki üretim ünitelerinin devreye girmesinin ve devreden çıkmasının ancak buhar basıncını ve sıcaklığını ayarlayan personelle mümkün olmasından kaynaklanır. Özellikle birden fazla ünite ihtiva eden santrallerde, devreye alınacak ve devreden çıkarılacak üniteler belirlenirken, mevcut personel sayısı da dikkate alınarak karar verilmelidir [4].

2.3.5 Rampa limitleri

Termal üretim ünitelerinde, belirli bir süre içinde üretimi bir değerden başka bir değere değiştirme limiti mevcuttur. Bu kısıt, ünitenin termal ve mekanik limitlerinden kaynaklanmaktadır. Ünitelerin aşağı rampa ve yukarı rampa limitleri olarak adlandırılan, 1 saat içinde ünitenin gerçekleştirebileceği pozitif ve negatif üretim değişimi limitleri R_i^{up} ve R_i^{down}, **(2.10)** ile tanımlanır.

$$\begin{aligned} P_i^t - P_i^{t-1} &\le R_i^{up} \\ P_i^{t-1} - P_i^t &= R_i^{down} \end{aligned} \qquad \textbf{(2.10)}$$

Ünitenin rampa limitlerinin üretim sürecine ek olarak üniteyi devreye alırken ve devreden çıkarırken de göz önünde bulundurulması gerekir [28].

2.3.6 Devrede olma zorunluluğu

Bazı ünitelerin belirli zamanlarda devrede olma zorunluluğu vardır. Bu zorunluluk, şebekenin gerilim seviyesini desteklemek gibi şebeke kaynaklı bir gereklilik veya santral içindeki buhar ihtiyacını karşılamak gibi ikincil ihtiyaçlardan kaynaklanır.

3. PARÇACIK SÜRÜ OPTİMİZASYONU

3.1 Giriş

Parçacık Sürü Optimizasyonu (PSO) 1995'te Dr. Eberhart ve Dr. Kennedy tarafından geliştirilmiş popülâsyon temelli sezgisel bir optimizasyon tekniğidir. Kuş veya balık sürülerinin sosyal davranışlarından esinlenilerek geliştirilmiştir. Sürüdeki bireylerin basit bir davranış kalıpları vardır; sürünün hareketlerini takip etmek ve yeme ulaşmak. Bu kalıptan meydana çıkan kolektif bilinç, matematiksel olarak, çok boyutlu bir arama uzayındaki optimal alanların keşfi olarak tanımlanabilir [29].

PSO yönteminde, sürüdeki bireyler parçacıklar olarak tanımlanır ve çok boyutlu bir arama uzayında uçurulurlar. Parçacıklar kolektif eğilimleri nedeniyle sürüdeki diğer parçacıkların başarılı hareketlerinden etkilenerek pozisyon değiştirirler. Parçacıkların birbirlerinden etkilenmesi nedeniyle PSO simbiyotik bir dayanışma özelliği gösterir. Bu sosyal davranışın modellenmesinin bir sonucu olarak arama işlemi sırasında parçacıklar, arama uzayındaki daha önce bulunmuş iyi sonuçlara doğru rastgele bir hareket gösterirler [30].

Bu bölümün ilk kısmında, klasik PSO formülleri ve algoritması anlatılmıştır. İkinci kısmında ise çeşitli problemlerin çözümü için geliştirilmiş PSO algoritması varyasyonları, klasik algoritmadan ayrılan yönleri vurgulanarak özetlenmiştir.

3.2 Temel PSO Algoritması

Temel PSO algoritması kuş veya balık sürüsündeki bireylerin, sürünün hareketlerini gözleyerek yiyecek aramalarını taklit eder. Parçacıkların hareketi modellenirken S adet parçacıktan oluşan bir sürünün, D boyutlu bir arama uzayında, $f(x)$ fonksiyonunu minimize etmek için hareket ettiği varsayılır. i parçacığı, x_i parçacığın o andaki pozisyonunu, v_i ise parçacığın o andaki hızını tanımlar. t anı için parçacığın bulunduğu konum, arama uzayının boyutlarına göre değişmekle birlikte **(3.1)** ile tanımlanır.

$$x_i(t) = \left[x_{i1}, x_{i2}, \ldots, x_{iD}\right] \quad \textbf{(3.1)}$$

t anı için parçacığın hızı, konum vektörüyle aynı boyutlarda olup **(3.2)** ile tanımlanır.

$$v_i(t) = \left[v_{i1}, v_{i2}, \ldots, v_{iD}\right] \quad \textbf{(3.2)}$$

Parçacığın $t+1$ anındaki pozisyonu, t anındaki pozisyonuna hız eklenerek **(3.3)** ile bulunur.

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad \textbf{(3.3)}$$

Parçacıkların hareketleri sırasında, $f(x)$ fonksiyonunda en iyi sonuçları vermiş olan x_i pozisyonları, algoritma tarafından güncellenerek kaydedilir. Parçacığın o ana kadar bulunduğu en iyi pozisyon x_{pi} sürüdeki parçacıklar tarafından o ana kadar bulunulmuş en iyi pozisyon ise x_{gi} olarak tanımlanır. t anına kadar elde edilmiş en iyi pozisyonlar **(3.4)**'teki gibidir,

$$\begin{aligned} x_{pi}(t) &= \left[x_{pi1}, x_{pi2}, \ldots, x_{piD}\right] \\ x_{gi}(t) &= \left[x_{gi1}, x_{gi2}, \ldots, x_{giD}\right] \end{aligned} \quad \textbf{(3.4)}$$

Optimizasyon sürecini yöneten dolayısıyla parçacığın kişisel deneyimi ile sürünün deneyimini, parçacığın hareketlerine yansıtan, parçacığın hızıdır ve aşağıdaki **(3.5)** formülüne göre belirlenir,

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (x_{pi}(t) - x_i(t)) + c_2 \cdot r_2 \cdot (x_{gi}(t) - x_i(t)) \quad \textbf{(3.5)}$$

Formüldeki c_1 ve c_2 değerleri, güven katsayıları olarak adlandırılan skaler değerlerdir.

$(x_{pi}(t) - x_i(t))$ işlemi parçacığın, kendi en iyi pozisyonuna doğru ilerlemesini sağlar. Bu işlemin güven katsayısı c_1 büyüdükçe, parçacığın kendi en iyi konumuna doğru ilerlemedeki ısrarcılığı artar.

$(x_{gi}(t) - x_i(t))$ işlemi ise parçacığın, sürünün en iyi pozisyonuna doğru ilerlemesini sağlar. Bu işlemin güven katsayısı c_2 büyüdükçe de parçacığın sürünün en iyi konumuna doğru yönelmedeki ısrarcılığı artar.

Arama sürecindeki rastgelelik r_1 ve r_2 katsayıları ile sağlanır. Bu katsayılar, standart dağılım ile (0,1) aralığında üretilerek, parçacık hareketlerine rastgelelik katarlar [30].

w ise parçacığın bir önceki hareketindeki hızının katsayısı olup atalet ağırlığı olarak adlandırılır. Sönüm parametresi olarak kullanılarak algoritmanın keşfetme ve sömürme fazlarının düzenlenmesini sağlar.

Klasik PSO algoritmasında w sabit olmasına rağmen, literatürde değişken olması önerilmektedir. Parçacıklar global minimuma yaklaştıkça ağırlık katsayısının küçültülmesi, parçacıkların daha kararlı hareket etmesini sağlar. Optimizasyon sürecinde atalet ağırlığının lineer olarak azaltılmasının daha iyi sonuçlar verdiği literatür çalışmalarıyla da kanıtlanmıştır [31]. Lineer azalan w denklemi **(3.6)**'daki gibidir.

$$w = \frac{w_{\max} - w_{\min}}{iter_{\max}} \cdot iter \qquad \textbf{(3.6)}$$

Bu denklemde $w_{\max}$ ve $w_{\min}$ optimizasyona başlamadan tanımlanmış sınırları, $iter_{\max}$ toplam iterasyon sayısını, $iter$ ise söz konusu andaki iterasyonu tanımlar.

Parçacıkların hareketlerinin daha kararlı olması için atalet ağırlığını kontrol etmek kadar parçacık hızı v_i'i kontrol etmek de etkilidir. Büyük v_i değeri, parçacığın arama uzayında büyük hareketler yaparak iyi pozisyonları atlayabilmesine neden olur. Küçük v_i değeri ise parçacığın çok küçük hareketler yaparak arama uzayının keşfini kısıtlar. Hız kararlılığını sağlayacak minimum-maksimum hız sınırlamasını **(3.7)**'deki gibi getirmek parçacığın daha kararlı hareket etmesini sağlar.

$$v_{\min} \le v_i \le v_{\max} \qquad \textbf{(3.7)}$$

Klasik PSO algoritmasının adım adım ilerlemesi aşağıdaki gibidir.

1.Başlangıç

(a) Sabit değerler belirlenir, $v_{\min}, v_{\max}, w_{\min}, w_{\max}, c_1, c_2$

(b) Başlangıç için parçacıklara sınırlar dâhilinde rastgele hızlar atanır, v_i

(c) Başlangıç için parçacıklar arama uzayına rastgele yerleştirilirler, x_i

(d) İterasyon=1

2. Optimizasyon

(a) x_i konumu için amaç fonksiyonu $f(x_i)$ hesaplanır

(b) Eğer $f(x_i) \leq f(x_{pi})$ ise $f(x_{pi}) = f(x_i)$ ve $x_{pi} = x_i$ olarak atanır

(c) Eğer $f(x_i) \leq f(x_{gi})$ ise $f(x_{gi}) = f(x_{pi})$ ve $x_{gi} = x_{pi}$ olarak atanır

(d) Durma kriteri sağlandıysa 3. adıma gidilir

(e) Parçacıkların hızları, v_i **(3.5)**'e göre güncellenir

(f) Parçacıkların konumları, x_i **(3.3)**'e göre güncellenir

(g) İterasyon sayısı arttırılır, iterasyon=iterasyon+1

(h) 2. adıma gidilir

3. Son

Klasik PSO algoritması akış diyagramı Şekil 3.1'te verilmiştir.

3.3 Geliştirilmiş PSO Algoritmaları

PSO algoritmasının spesifik problemlere daha iyi çözüm bulmak için modifiye edilmiş birçok varyasyonunu literatürde görmek mümkündür. Optimizasyon problemlerinin birden fazlasına uygulanmış ve başarılı sonuçlar vermiş PSO varyasyonlarının dikkat çeken ve sıklıkla kullanılanlarına bu bölümde özet olarak değinilmektedir.

3.3.1 Ayrık PSO

Klasik PSO algoritması, sürekli arama uzayındaki lineer olmayan optimizasyon problemlerinin çözümünde kullanılmaya uygundur. Buna rağmen, pratik mühendislik problemlerinde ikili sistemler sıkça kullanıldığı gibi bu sistemlerin optimizasyonu için klasik PSO işlevsiz kalmaktadır. Bu tür problemlerin çözümü için Kennedy ve Eberhart ayrık ve ikili sistemde çalışan bir PSO versiyonu geliştirmiştir. Hız güncelleme formülasyonu klasik yöntemdekinin birebir aynısıdır. Bu modifikasyonla parçacıklar, kişisel ve sosyal faktörleri kullanmaya devam ederken, evet-hayır, 1-0 gibi cevaplar verebilir hale gelmiştir [33].

Rastgele bir değerle karşılaştırılan $sig(v_i)$ parametresi, parçacığın 1-0 seçeneklerinden hangisini seçeceğini belirlenmesinde kullanılır. Eğer v_i büyükse, parçacık 1'i seçmeye daha yatkınken, küçük değerlerinde 0'ı seçme eğiliminde

olacaktır. Olasılıksal karşılaştırma değerinin [0, 1] aralığında bulunması gerekir. Bu nedenle üretilmiş v_i parametresini de [0, 1]. aralığına taşımak için yapay sinir ağları(YSA) uygulamalarında sıklıkla kullanılan **(3.8)**'deki sigmoid fonksiyonundan faydalanılır [34].

$$sig(v_i) = \frac{1}{1+\exp(-v_i)} \qquad \textbf{(3.8)}$$

Ayrık PSO algoritmasından tek farkı, v_i'nin hesaplanmasından sonraki sigmoid fonksiyonu ve sigmoid fonksiyonu sonucuna göre yapılan pozisyon güncellemesi **(3.9)**'daki gibi yapılmasıdır.

$$x_i = \begin{Bmatrix} 1 & eğer \quad rand \le sig(v_i) \\ 0 & aksitakdirde \end{Bmatrix} \qquad \textbf{(3.9)}$$

Ayrık PSO için birçok nokta hala araştırılmaya açık olarak beklemektedir. Literatürde, algoritma üzerinde yapılmış bazı modifikasyonlar kuantum yaklaşımı önermektedir [35]. Ayrık PSO algoritmasının geliştirilmesi ve sonuç kalitesinin arttırılması amacıyla yapılan çalışmalar da literatürde mevcuttur [36].

3.3.2 Karmaşık tamsayılı lineer olmayan programlamalı PSO

Mühendislik problemlerinin genellikle lineer olmayan amaç fonksiyonları vardır ve bu fonksiyonlar hem sürekli hem de ayrık değişkenlere sahiptir. Kennedy ve Eberhart PSO'nun ayrık ve sürekli versiyonlarının entegrasyonu konusunda çalışmalar yapmışlardır. Fukuyama da karmaşık tamsayılı lineer olmayan programlamalı (KTLOP) PSO için klasik PSO algoritmasını modifiye etmiştir [37]. Bu çalışmada, ayrık değişkenler **(3.3)** ve **(3.5)** formüllerinin biraz modifiye edilmesiyle algoritmaya dâhil edilmişlerdir.

Parçacıkların pozisyonları ve hızları, sürekli sayılar yerine ayrık sayılarla ifade edilmiştir. **(3.3)**' ün hesaplanmasındaki rastgele değerler için ayrık bir rastgele sayı kullanılmıştır. Yapılan bu değişiklikle herhangi bir karmaşa yaşanmadan hem ayrık hem de sürekli sayılar aynı algoritma için kullanılabilmektedir. Bu algoritma kullanılarak reaktif güç ve gerilim kontrolü problemlerinde oldukça iyi sonuçlar elde edilmiştir. [34].

3.3.3 Kısıtlama faktörlü PSO

Klasik PSO algoritmasındaki denklemler bir nevi fark denklemleri olarak ele alınabilirler. Bu yaklaşımla, sistemdeki dinamik değişkenler, arama prosedürü sırasında fark denklemlerinin özdeğerleriyle analiz edilir. Clerc ve Kennedy, PSO' nun basitleştirilmiş durum denklemini kullanarak PSO'nun kısıtlama faktörlü(KF) yaklaşımını geliştirmiştir. Klasik PSO algoritmasıyla arasındaki fark, v_i formülünün **(3.10)**'daki gibi değiştirilmesinden gelir.

$$v_i(t+1) = K\left[v_i(t) + c_1 \cdot r_1 \cdot (x_{pi}(t) - x_i(t)) + c_2 \cdot r_2 \cdot (x_{gi}(t) - x_i(t))\right] \quad \textbf{(3.10)}$$

Parametrelerin belirlenmesi aşamasında $4 < \varphi$ olacak şekilde $c1$ ve $c2$ katsayıları belirlenir. φ büyüdükçe K değeri **(3.11)** ile küçülür.

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad \textbf{(3.11)}$$

Yani algoritmanın yakınsama karakteristiği φ tarafından **(3.12)** ile kontrol edilebilir.

$$\varphi = c_1 + c_2 \quad \textbf{(3.12)}$$

Literatürdeki çalışmalar göstermektedir ki, kimi problemlerde KF PSO, klasik PSO algoritmasından daha iyi sonuçlar vermektedir. Buna rağmen, algoritma yalnızca bir tek parçacığın dinamik davranışını dikkate alır ve parçacıklar arası etkileşime yoğunlaşır. Parçacığın ve sürünün en iyi pozisyonlarının, sistem dinamikleri üzerindeki etkisinin anlaşılması gelecekteki çalışmalar için önemini korumaktadır [34].

3.3.4 Hibrit PSO

HPSO, Angeline tarafından, evrimsel hesaplama yöntemlerinin seçim mekanizmalarının PSO' ya uyarlanmasıyla geliştirilmiş bir PSO çeşididir [38]. HPSO' da her iterasyonda, iyi sonuçlar veren parçacıkların sayısı arttırılırken, kötü sonuçlar verenlerin sayıları azaltılır. PSO' nun arama yöntemi parçacığın ve sürünün en iyisine yönelme üzerine kurulduğu için arama uzayı da bu iyi pozisyonlar tarafından sınırlandırılmış olur. HPSO algoritmasının seçim tekniğiyle ise en iyi konumların önemi gittikçe azalmakta ve daha geniş bir alanda arama yapmak mümkün olmaktadır. Arama süresince, kötü sonuçlara sahip olan parçacıklar

yerlerini daha başarılı parçacıklara bıraksalar da, elde ettikleri iyi konumlar saklanmaya devam etmekte ve aramanın gidişatına göre aramaya yeniden dâhil olmaları söz konusu olabilmektedir. İyi sonuç veren alana yoğunlaşmak ve beklenen değerleri elde edemeyince daha önce göz ardı edilmiş bir alana parçacıkları geri sıçratmak olarak özetlenebilecek olan bu yöntem bir hayli zaman alıcı olabilmektedir. Bu nedenle HPSO'nun başarısı spesifik problemlere uygulandığında dikkat çekici olmakla birlikte, klasik PSO' ya rakip olabilecek kadar başarılı değildir [34].

3.3.5 Lokal en iyili PSO

Lokal en iyi(Lbest) modeli Eberhart ve Kennedy tarafından geliştirilmiştir. Bu PSO yönteminde, parçacıklar gruplara ayrılır ve sürünün en iyi konumu yerine içinde bulunduğu grubun en iyi konumunu kullanılır. PSO hız denklemi **(3.13)**'teki gibi değiştirilir ve x_{gi}'nin yerini Lbest konumu, x_{li} değeri gelir.

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (x_{pi}(t) - x_i(t)) + c_2 \cdot r_2 \cdot (x_{li}(t) - x_i(t)) \qquad \textbf{(3.13)}$$

Hız denkleminin yukarıdaki gibi modifiye edilmesi dışında Lbest yaklaşımıyla PSO, klasik PSO algoritmasının aynısıdır [34].

3.3.6 Uyum sağlayabilen PSO

Uyum sağlayabilen (US) PSO'da klasik atalet ağırlıklı PSO algoritmasından farklı olarak 2 yeni parametre P_1 ve P_2 tarafından, her iterasyon için parçacığın ve sürünün en iyi pozisyonlarına yaklaşma olasılığı belirlenerek arama yapılır. P_1 ve P_2 parametreleri c_1 ve c_2 değerlerini belirleyerek algoritmaya **(3.14)**'teki gibi dâhil olurlar [34].

$$c_2 = \frac{2}{P_1}, c_1 = \frac{2}{P_2} - c_2 \qquad \textbf{(3.14)}$$

Klasik PSO' da atalet ağırlığı olarak adlandırılan w katsayısı US PSO'da **(3.15)**'deki gibi hesaplanır. Bu yöntemle, parçacıklar x_{gi}, merkez nokta olacak şekilde hareket ederler.

$$w_i = x_{gi}(t) - \left[\frac{c_1 \cdot (x_{pi}(t) - x_i(t)) + c_2 \cdot (x_{gi}(t) - x_i(t))}{2} + x_i(t) \right] \qquad \textbf{(3.15)}$$

Parçacık x_{gi} noktasında geldiğinde, w ağırlığı 0'a eşit olacağı için algoritma takılır. Bu nedenle parçacık x_{gi} noktasına geldiğinde P_1 ve P_2 parametreleri güncellenerek parçacığın bulunduğu noktadan uzaklaşıp aramaya devam etmesi sağlanır. APSO'nun hız güncelleme formülü ise **(3.16)**'daki gibidir.

$$v_i(t+1) = w_i + c_1 \cdot r_1 \cdot (x_{pi}(t) - x_i(t)) + c_2 \cdot r_2 \cdot (x_{gi}(t) - x_i(t)) \quad \textbf{(3.16)}$$

3.3.7 Evrimsel PSO

Evrimsel PSO (EPSO)' da, PSO bir seçim prosedürünü dâhil edilir ve parametrelere adaptasyon yeteneği kazandırılır [39]. Öncelikle parçacıklar kopyalanarak çoğalırlar ve ağırlık bileşenleri **(3.17)**'deki gibi mutasyona uğrar.

$$w_{ik}(t)^* = w_{ik}(t) + \tau \cdot N(0,1) \quad \textbf{(3.17)}$$

Aynı mutasyon, parçacığın hesaba katacağı sürünün en iyi konumu, x_{gi} için de **(3.18)**'le uygulanır.

$$x_{gi}(t)^* = x_{gi}(t) + \tau' \cdot N(0,1) \quad \textbf{(3.18)}$$

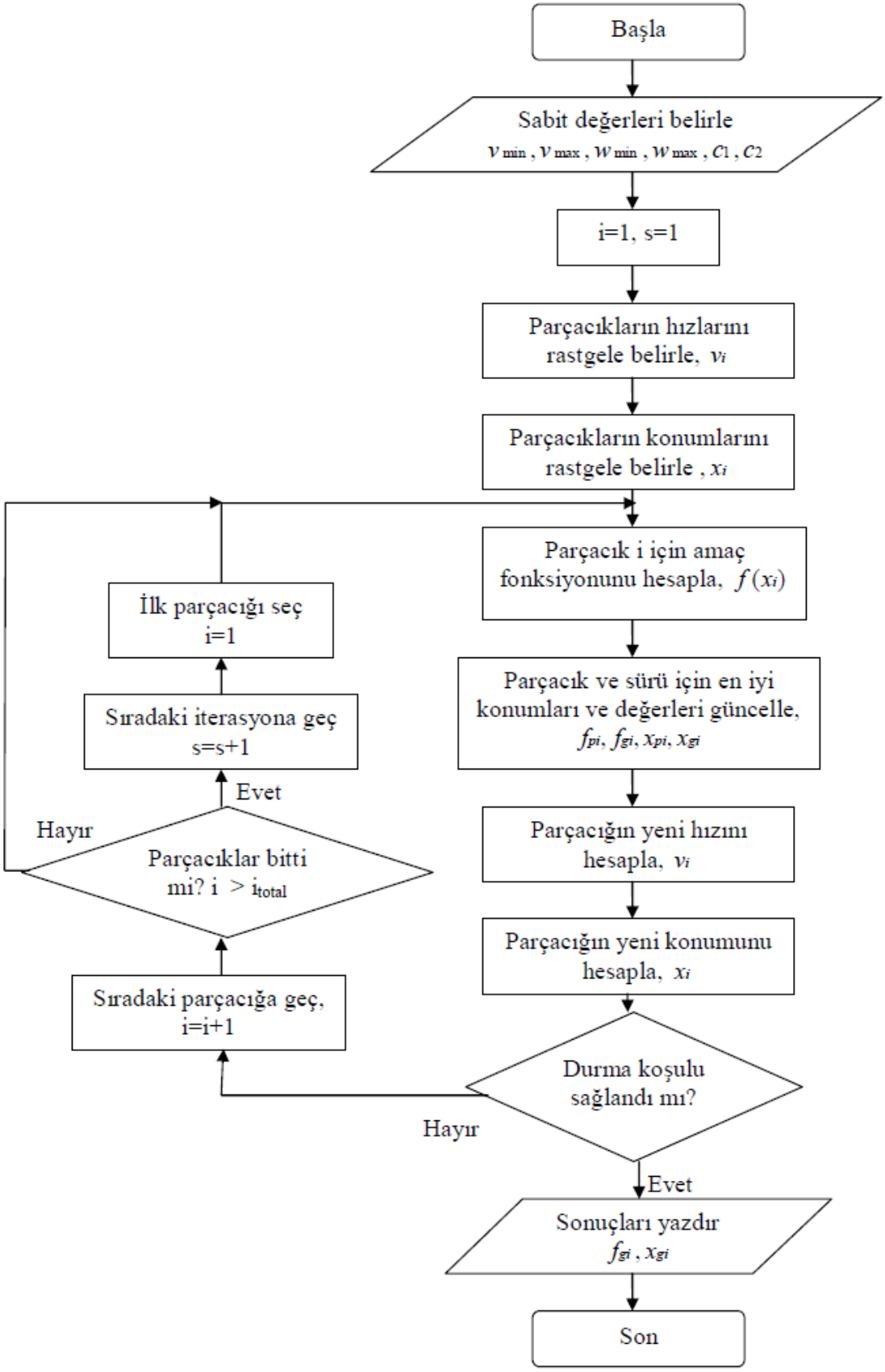
Mutasyon işlemlerinde kullanılan τ ve τ' öğrenme faktörleridir. Sabit değerli olabilecekleri gibi, stratejik olarak değişen değerli de olabilirler. Mutasyon geçirmiş her parçacık **(3.19)** ve **(3.20)**'ye göre yavru parçacığını oluşturur.

$$v_i(t+1) = w_{i0} \cdot v_i(t) + w_{i1} \cdot (x_{pi}(t) - x_i(t)) + w_{i2} \cdot (x_{gi}(t) - x_i(t)) \quad \textbf{(3.19)}$$

Yavru parçacıkların amaç fonksiyonlarının aldıkları değerler karşılaştırılarak, en iyi yavru parçacıklar seçilir ve yeni parçacık sürüsü oluşturulur.

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad \textbf{(3.20)}$$

EPSO, hem evrimsel stratejiden hem de parçacık sürü optimizasyonun ilerleme stratejisinden faydalandığı için hem ES'den hem de klasik PSO'dan daha iyi yakınsama gösterir [34].



Şekil 3.1 : PSO akış diyagramı [32].

4. BİRİM YÜKLENME PROBLEMİNİN PSO İLE ÇÖZÜMÜ

4.1 Giriş

Güç sistemlerinin boyutları büyüdükçe, kısıtlar artmakta ve ünite yüklenme probleminin çözümü de zorlaşmaktadır. Ünite yüklenme probleminin PSO yöntemiyle çözüldüğü çalışmaların çoğunda maliyetlerin hesaplanması, rezerv gücün sağlanması, minimum devrede-devre dışı kalma süreleri gibi problemi karmaşık hale getiren ve çözümü zorlaştıran kısıtlar farklı şekillerde sağlanmıştır. Ünitelerin durumları genellikle üniteleri açık/kapalı olarak gösteren matrislerle tanımlanmıştır.

Yapılan çalışmada da, ünitelerin açık/kapalı durumları, elemanları 0/1 olan matrislerle tanımlanmıştır. Her aşamada parçacıklar hareketlerini yapmadan önce, çevrelerindeki kısıt duvarları belirlenir. Parçacıkların sınırları belli olan bu alanlarda hareket etmeleri beklenir. Ünite devreye alma problemi ayrık PSO yöntemi ile ünitelerin ekonomik yüklenmesi problemi ise klasik PSO yöntemi ile çözülmüştür.

Bu çalışmada geliştirilen maliyet amaçlı ünite yüklenme problemi çözümü 5 aşamada gerçekleştirilir.

1. aşamada, öncelikle probleme ait güç sınırları, maliyet katsayıları vb. bölümün devamında detaylı olarak açıklanacak sabit değerler girilir. APSO ve PSO yöntemlerinin sabit değerleri, parçacık sayısı, parçacıkların uzayda hareke başlayacakları konumlar, başlangıç maliyetleri vb. değerler de bu aşamada belirlenir.

2. aşamada, APSO ile saatlik bazda devreye alınacak ve devre dışı bırakılacak üniteler belirlenir. Bu belirleme, ünitelerin minimum devrede ve devre dışı olma kısıtları ile rezerv güç kısıtları göz önünde bulundurularak yapılır. Ele alınan problemdeki her saat için devrede ve devre dışı bırakılacak üniteler belirlenir.

3. aşamada, PSO ile her bir saat için ünitelerin ne kadar güç üretecekleri belirlenir. 3. aşama, algoritmada 2. aşamanın alt döngüsü olarak yer alır. Bu aşamada, ünitelerin üretim limitleri kısıtı ile güç dengesi kısıtı gözönünde bulundurulur. Her bir ünitenin saatlik bazda üretimleri belirlenir.

4. aşamada, maliyetler hesaplanarak PSO en iyi konumları güncellenir. Her bir ünitenin devreye alındığı ve devre dışı bırakıldığı saatler gözden geçirilerek hesaba katılması gereken başlangıç maliyetleri belirlenir. Ünitelerin üretimlerinden yakıt maliyetleri hesaplanır. Başlangıç maliyetleri ile yakıt maliyetlerinin toplamı, amaç fonksiyonu olan toplam maliyetin saatlik değerini oluşturur.

5. aşamada, her saat için 4. aşamadan alınan maliyet değerleri toplanarak APSO en iyi konumları güncellenir. APSO en iyi konumları tüm saatler için devrede-devre dışı olma durumları ve PSO ile hesaplanan ünite güçleri belirlendikten sonra güncellenir.

Tanımlanmış iterasyonlar tamamlandığında, 5. aşamadan alınan en iyi maliyet, saatlik devrede-devre dışı olma durumları ve ünitelerin saatlik üretimleri problemin çözümünü oluşturur.

Bu bölümün 1. kısmında ele alınan ünite yüklenme probleminin formülasyonu verilmiştir. 2. kısmında ise uygulanan yöntemin algoritması ve çalışma aşamaları detaylı olarak anlatılmıştır.

4.2 Ünite Yüklenme Problem Formülasyonu

Ele alınan problem, literatürde üzerinde en çok çalışma yapılmış olan basitleştirilmiş bir ünite yükleme problemidir. Amaç fonksiyonu, ele alınan süre boyunca söz konusu talep güçlerinin P_{load} ve minimum rezerv miktarı $P_{reserve}$ sağlanarak karşılanırken, bu süreçte meydana gelen toplam işletme maliyeti $F_{totali}(P_i)$'nin **(4.1)** minimize edilmesidir.

$$Minimize \;\; F_{total_s} = \sum_{t=1}^{NT} \sum_{i=1}^{NG} F_i^t(P_i^t) \cdot u_i^t + S_{x_i}^t u_i^t \qquad \textbf{(4.1)}$$

1. Maliyet: Yakıt maliyeti **(4.2)**

$$F_i^t(P_i^t) = a_i + b_i \cdot P_i^t + c_i \cdot P_i^{t2} \qquad \textbf{(4.2)}$$

2. Maliyet: Başlangıç maliyeti **(4.3)**

$$S_{x_i}^t = \begin{cases} S_{h_i} & eğer \quad t_i^t \le t_{cold_i} \\ S_{c_i} & aksi\,takdirde \end{cases} \qquad \textbf{(4.3)}$$

1. Kısıt: Güç dengesi **(4.4)** ile tanımlanır.

$$P_{load}{}^{t} = \sum_{i=1}^{NG} P_i^t \cdot u_i^t \qquad \textbf{(4.4)}$$

2. Kısıt: Güç rezervi **(4.5)** ile tanımlanır.

$$\sum_{i=1}^{NG} P_{maxi} \cdot u_i \geq P_{load} + P_{reserve} \qquad \textbf{(4.5)}$$

3. Kısıt: Ünite üretim limitleri **(4.6)** ile tanımlanır.

$$P_{mini} \leq P_i \leq P_{maxi} \qquad \textbf{(4.6)}$$

4. Kısıt: Ünite minimum açma kapama zamanları **(4.7)** ile tanımlanır.

$$u_i^t = \begin{cases} 1 & eğer \quad u_i^{t-1} = 1 \quad \& \quad 1 \leq t_i^t \leq t_{up} \\ 0 & eğer \quad u_i^{t-1} = 0 \quad \& \quad 1 \leq t_i^t \leq t_{down} \end{cases} \qquad \textbf{(4.7)}$$

4.3 Ünite Yüklenme Problemi PSO Algoritması

Ünite yüklenme problemi, APSO ve PSO yöntemleri ile 5 aşamada çözülmektedir. Algoritmanın adım adım ilerleme detaylarına girilmeden önce, genel yapısı Şekil 4.1'de görülebilir**.**

1. aşama olan başlangıçta, ele alınan problemin değişkenleri tanımlanır.

1.Ünitelere ait değerler girilir,

(a) Ünite üretim limitleri: P_{mini} , P_{maxi}

(b) Ünite minimum devrede ve devre dışı olma süreleri: t_{upi} , t_{downi}

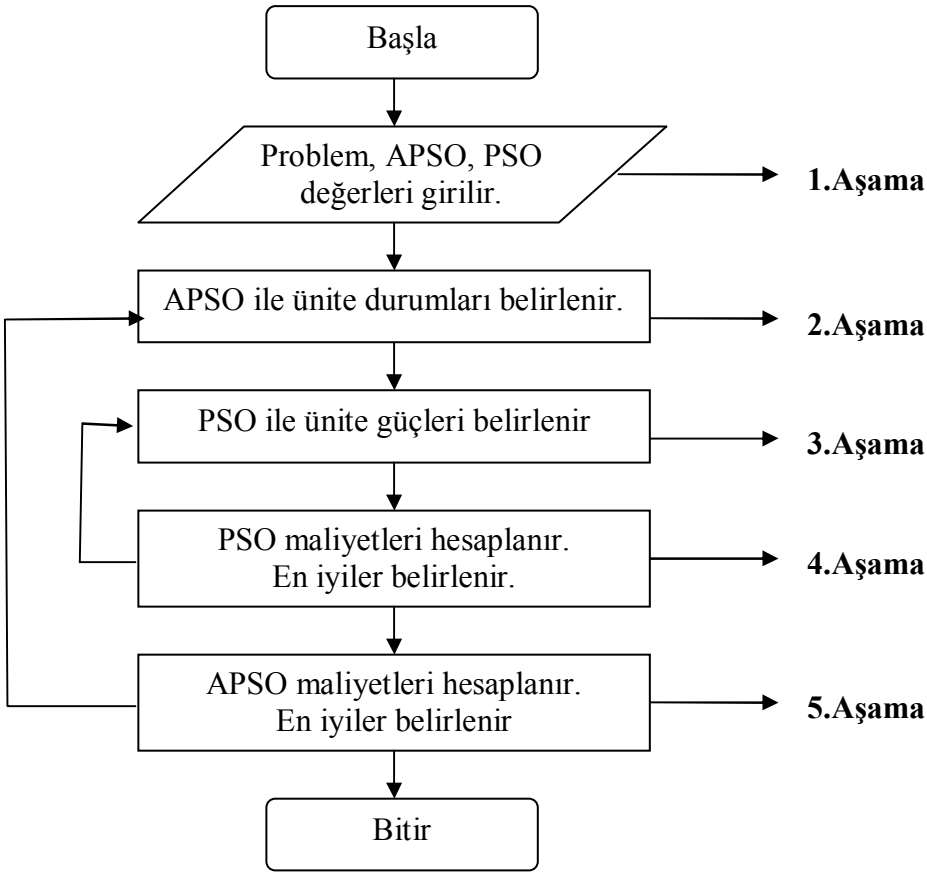
(c) Ünite soğuk başlangıç süresi: t_{coldi}

(d) Ünite soğuk ve sıcak başlangıç maliyetleri: S_{hoti}, S_{coldi}

(e) Ünite üretim maliyeti katsayıları: a_i, b_i, c_i

2. Ünite yüklenme problemine ait değerler girilir,

(a) Probleme başlandığında üniteleri durumu: $State_i$



Şekil 4.1 : Algoritma aşamaları.

(b) Saatlik güç talepleri: P_{demand}

(c) Saatlik rezerv güç talepleri: $P_{reserve}$

3. APSO sabit değerleri girilir,

(a) Parçacık hız limitleri: $f_{pi}, f_{gi}, x_{pi}, x_{gi}, v_{\max}$

(b) Atalet katsayısı limitleri: $w_{\min}, w_{\max}$

(c) Güven katsayıları : c_1, c_2

(d) Parçacık sayısı : s

(e) İterasyon sayısı : i

4. PSO sabit değerleri girilir,

(a) Parçacık hız limitleri: $v2_{max}, v2_{min}$

(b) Atalet katsayısı limitleri: $w2_{min}, w2_{max}$

(c) Güven katsayıları : $c2_1, c2_2$

(d) Parçacık sayısı : $s2$

(e) İterasyon sayısı : $i2$

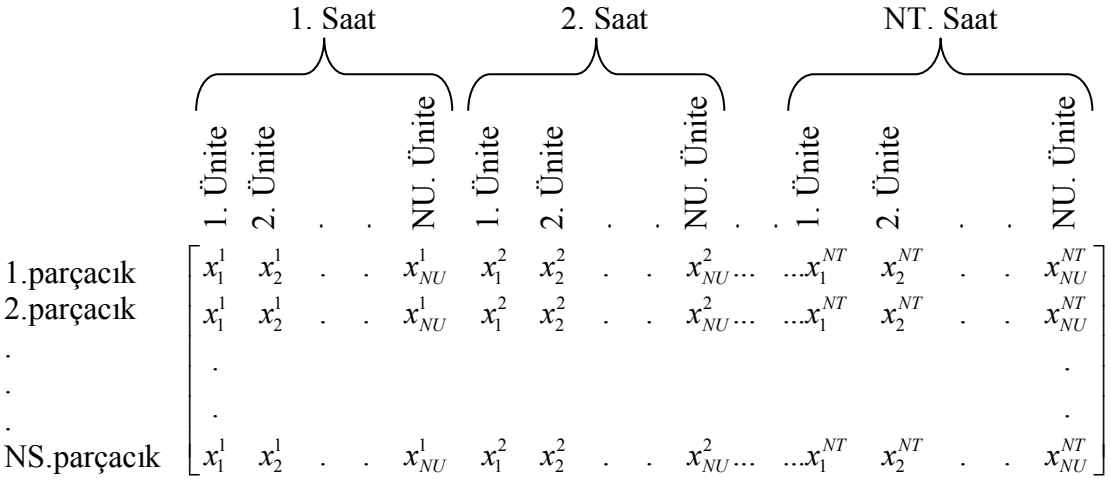
5. APSO başlangıç değerleri girilir,

(a) Başlangıç konumu: x_u^t

Problem tanımındaki her bir saat boyunca devreye alınacak ve devre dışı bırakılacak üniteler **(4.8)**'e göre başlangıçta rastgele belirlenir.

$$x_u^t = round(rand(0,1)) \qquad \textbf{(4.8)}$$

PSO ünite yüklenme problemine uygulanırken, her bir parçacık konumu bir vektörle tanımlanır. Parçacığa ait konum vektörünün 1. elemanı, 1. saatte 1. ünitenin alacağı durumu gösterir. 1. eleman, ünitenin devrede olma durumu için 1, devre dışı olma durumu için 0 değerini alır. NU. eleman ise 1. saatte NU. ünitenin alacağı durumu gösterir. Bu şekilde, t. saatte u. ünitenin alacağı durum, parçacık konum vektörünün $(t \cdot NU) + u$'uncu elemanına bakılarak anlaşılabilir. Her parçacığın konum vektörlerinden oluşturulmuş konum matrisinin yapısı Şekil 4.2'de görülmektedir.

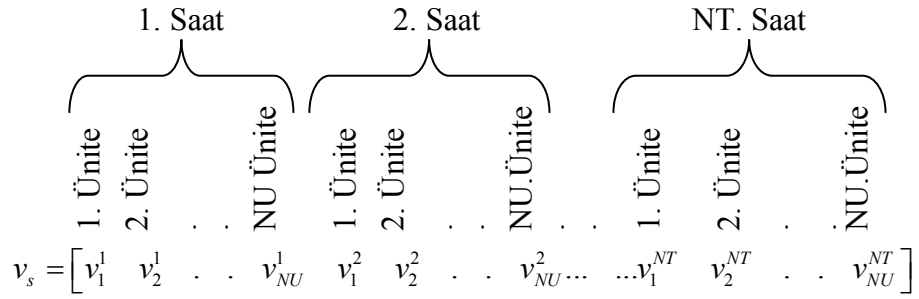


Şekil 4.2 : Konum matrisi.

(b) Başlangıç hızı: v_u^t

Parçacıkların başlangıçtaki hızları da tıpkı başlangıç konumları gibi rastgele belirlenir **(4.9)**. S. Parçacığının hız vektörü Şekil 4.3'te görülmektedir.

$$v_u^t = rand(0,1) \qquad \textbf{(4.9)}$$



Şekil 4.3 : Hız vektörü.

(c) Başlangıç en iyileri: Parçacıkların en iyileri , F_{pbest_s} , x_{pbest_s}

Sürünün en iyileri, F_{gbest}, x_{gbest}

Parçacıkların en iyi konumu da, başlangıçta her parçacık için **(4.10)**'daki gibi rastgele atanır. Parçacık en iyi konum vektörünün yapısı Şekil 4.4'te görülmektedir.

$$x_{pbest_u}^t = round(rand(0,1)) \qquad \textbf{(4.10)}$$

$$x_{pbest_s} = \left[x_{pbest}{}_1^1 \quad x_{pbest}{}_2^1 \quad . \quad . \quad x_{pbest}{}_{NU}^1 \quad . \quad . \quad . \quad . \quad . \quad x_{pbest}{}_1^{NT} \quad x_{pbest}{}_2^{NT} \quad . \quad . \quad x_{pbest}{}_{NU}^{NT} \right]$$

Şekil 4.4 : Parçacığın en iyi konum vektörü.

Parçacıkların en iyi maliyeti **(4.11)** ile başlangıç için çok büyük bir değer olarak atanır. Bu sayede, parçacıklar arama uzayına ilk çıktıklarında ne kadar kötü bir sonuç elde ederlerse etsinler, elde edilmiş en iyi konum ve maliyet vektörlerinin güncellenmesini sağlanır. Aksi takdirde parçacıklar ilk başta rastgele olarak üretilmiş konumları, en iyi konumlar sandıkları için arama performansları düşük olur.

$$F_{pbest_s} = 10^{24} \qquad \textbf{(4.11)}$$

Sürünün en iyi konumu, başlangıç için **(4.12)**'yle rastgele atanır. Sürünün en iyi konum vektörü Şekil 4.5'te görülmektedir. En iyi konumları tamamen 0'lardan veya 1'lerden oluşan vektörler olarak atanmamalarının sebebi, uzayın keşfini teşvik etmektir. İterasyon ilerledikçe, parçacıklar elde edilmiş en iyi konumlara ilerledikler, program başlarken ne kadar rastgelelik yaratılırsa arama o kadar iyi olur.

$$x_{gbest_u}^t = round(rand(0,1)) \qquad \textbf{(4.12)}$$

$$x_{gbest} = \left[x_{gbest}{}_1^1 \quad x_{gbest}{}_2^1 \quad . \quad . \quad x_{gbest}{}_{NU}^1 \quad . \quad . \quad . \quad . \quad . \quad x_{gbest}{}_1^{NT} \quad x_{gbest}{}_2^{NT} \quad . \quad . \quad x_{gbest}{}_{NU}^{NT} \right]$$

Şekil 4.5 : Sürünün en iyi konum vektörü.

Sürünün en iyi maliyeti de, parçacıkların ilk hareketleriyle güncellenebilecek şekilde **(4.13)**' teki gibi atanır. Görüldüğü gibi sürünün en iyi maliyeti, parçacığından en iyi maliyetinden daha küçük seçilmiştir. Bunun sebebi yine parçacıkların arama sonuçları ne kadar kötü olursa olsun, parçacıkların kendi en iyi konumlarını güncellemesini sağlamaktır.

$$F_{gbest} = 10^{23} \qquad \textbf{(4.13)}$$

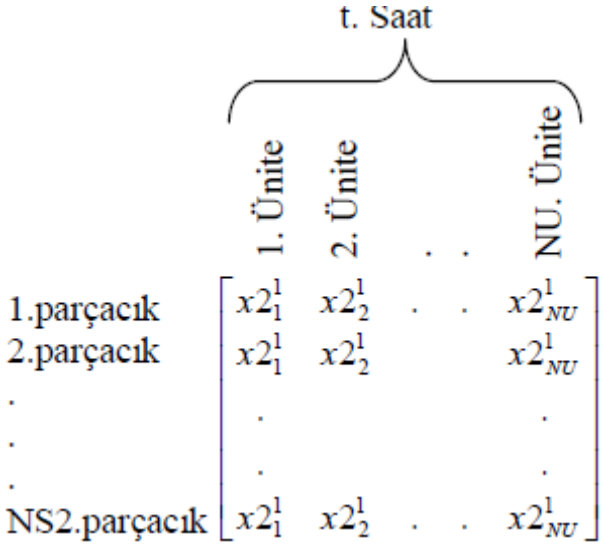
6. PSO başlangıç değerleri girilir,

(a) Başlangıç konumu: $x2_u^t$

Problem tanımındaki her bir saat boyunca devreye alınan ünitelerin ne kadar güç üretecekleri başlangıçta rastgele belirlenir. Bu rastgeleliğin makul bir aralıkta olması, parçacıkların daha kısa sürede sonuca ulaşmak için konum **(4.14)**'e göre belirlenir.

$$x2_u^t = rand(0,1) \cdot (P_{max_u} - P_{min_u}) \qquad \textbf{(4.14)}$$

Konum matrisi APSO yönteminde kullanılan matrisle aynı yapıdadır. Şekil 4.6'da görülen konum matrisi $x2_u$'nin satır sayısını, APSO için tanımlanmış parçacık sayısı s değil, PSO için tanımlanmış parçacık sayısı $s2$ belirler.



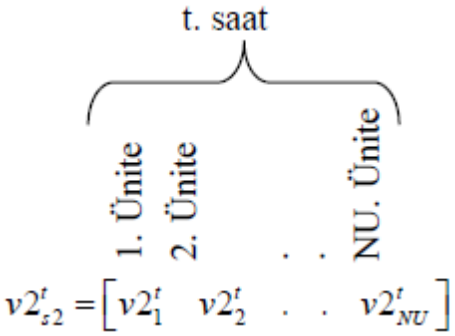
Şekil 4.6 : Konum matrisi.

(b) Başlangıç hızı: $v2_u^t$

Parçacıkların başlangıçtaki hızları da tıpkı başlangıç konumları gibi rastgele atanır. Parçacıkların konum değişikliklerini kararlı olarak yapmaları için hızları **(4.15)**'e göre belirlenir.

$$v2_u^t = \frac{rand(0,1)\cdot(P_{max_u} - P_{min_u})}{10} \qquad \textbf{(4.15)}$$

PSO hız vektörü ise Şekil 4.7' deki gibidir.



Şekil 4.7 : Hız vektörü.

(c) Başlangıç en iyileri: Parçacıkların en iyileri , $F2_{pbest_{s2}^t}, x2_{pbest_{s2}^t}$

Sürünün en iyileri, $F2_{gbest^t}, x2_{gbest^t}$

Parçacıkların en iyi konumu da, başlangıçta her parçacık için rastgele **(4.16)**'daki gibi atanır.

$$x2_{pbest_u^t} = rand(0,1)\cdot(P_{max_u} - P_{min_u}) \qquad \textbf{(4.16)}$$

$$x2_{pbest_{s2}^t} = \left[x2_{pbest_1^t} \quad x2_{pbest_2^t} \quad . \quad . \quad x2_{pbest_{NU}^t} \right]$$

Şekil 4.8 : Parçacığın en iyi konum vektörü.

PSO parçacıklarının en iyi maliyeti de, başlangıç için çok büyük bir değer olarak **(4.17)** ile atanır. Amaç, parçacıklar arama uzayına çıktıktan sonra hareketlerinin devamlılığını sağlamaktır.

$$F2_{pbest_{s2}^t} = 10^{22} \qquad \textbf{(4.17)}$$

Sürünün en iyi konumu da başlangıç için **(4.18)** ile rastgele atanır. Amaç başlangıçta parçacığı uygun bir noktadan harekete başlatmaktır.

$$x2_{gbest}{}^{t} = rand(0,1)\cdot(P_{max_u} - P_{min_u}) \qquad \textbf{(4.18)}$$

En iyi konum vektörü Şekil 4.9'da görülebilir.

$$x2_{gbest}{}^{t} = \left[x2_{gbest}{}_{1}^{t} \quad x2_{gbest}{}_{2}^{t} \quad . \quad . \quad x2_{gbest}{}_{NU}^{t} \right]$$

Şekil 4.9 : Sürünün en iyi konum vektörü.

Sürünün en iyi maliyeti de, parçacıkların ilk hareketleriyle güncellenebilecek şekilde **(4.19)**' daki gibi atanır.

$$F2_{gbest}{}^{t} = 10^{21} \qquad \textbf{(4.19)}$$

2. aşamada, ele alınan saat için ünitelerin açık kapalı durumları APSO ile belirlenir. Kısıtlar dâhilinde, PSO algoritmasındaki parçacıkların hareketine göre üniteler devreye alınır veya çıkarılır.

1. Her bir iterasyonda (k), her bir parçacık için (s), her bir saatte (t), her bir ünitenin (i) durumunun belirleneceği (0-1) döngü oluşturulur.

2. Minimum devrede ve devre dışı olma zaman kısıtına takılan üniteler **(4.20)**'ye göre belirlenerek durumları atanır.

$$u_u^t = \begin{cases} 1 & eğer \quad u_u^{t-1} = 1 \quad \& \quad 1 \le t_u^t \le t_{up} \\ 0 & eğer \quad u_u^{t-1} = 0 \quad \& \quad 1 \le t_u^t \le t_{down} \end{cases} \qquad \textbf{(4.20)}$$

Bu döngüde, ele alınan saatteki tüm üniteler kontrol edilir ve kısıta takılanların durumları değişmemek üzere atanır.

3. 2. adımdaki kısıta göre devreye alınmış ünitelerin, üretebilecekleri maksimum güçlerin toplamının rezerv güç kısıtını sağlayıp sağlamadığı **(4.21)**'le kontrol edilir.

$$\sum_{u=1}^{NU} P_{max_u}{}^{t} \cdot u_u^t \le P_{reserve}{}^{t} \qquad \textbf{(4.21)}$$

Devreye alınmış olan ünitelerin üretebilecekleri maksimum güçlerin toplamı, rezerv güç değerinden çıkarılır. Bu sayede $P_{reserve}{}^{t}$ değeri güncellenmiş olur ve bu noktadan

sonra güncel rezerv güç değeri göz önüne alınır. $P_{reserve}^{t}$ değeri kontrol edilir, 0'dan küçük veya 0 ise 6. aşamaya geçilir.

4. Rezerv güç kısıtı $P_{reserve}^{t}$, sağlanana kadar, açık-kapalı olma kısıtına takılmamış ama devreye de alınmamış üniteler için APSO yöntemi uygulanır.

(a) Parçacık hızı **(4.22)** ile hesaplanır,

$$v_u^t = w_i \cdot v_u^t + c_1 \cdot r_1 \cdot (x_{pbest_u}^t - x_u^t) + c_2 \cdot r_2 \cdot (x_{gbest}{}^t - x_u^t) \qquad \textbf{(4.22)}$$

(b) Sigmoid fonksiyonu **(4.23)** ile hesaplanır,

$$sig(v_u^t) = \frac{1}{1+\exp(-v_u^t)} \qquad \textbf{(4.23)}$$

(c) Sigmoid fonksiyonunun değerine göre parçacığın durumu **(4.24)**'le güncellenir

$$x_i^t = \begin{cases} 1 & eğer \quad rand \le sig(v_i^t) \\ 0 & aksitakdirde \end{cases} \quad sig(v_i^t) = \frac{1}{1+\exp(-v_i^t)} \qquad \textbf{(4.24)}$$

Yapılan algoritma testlerinde, ünitelerin hızlarını sırayla hesaplamanın, PSO'nun bu üniteleri sırayla devreye almasına neden olabildiği görülmüştür. Bu durum parçacıkların lokal minimumda takılı kalmalarına neden olmuştur. Takılmayı aşmak için APSO algoritmasına farklı bir bakış açısı getirilmiştir. Her ünite için hesaplanmış olan Sigmoid fonksiyonu değerleri bir vektör içinde tutulmuştur. Bu vektörün içindeki en büyük $sig(v_u^t)$ değerine sahip olan ünite devreye alınmıştır. Devreye alınan her ünite ile $P_{reserve}^{t}$ değeri güncellenmiştir. $P_{reserve}^{t}$ değeri sağlandığında ünitelerin devreye alınması durdurulmuştur.

Kısıta takılmamış ünitelerin hepsi devreye alındığı halde $P_{reserve}^{t}$ sağlanamamışsa, ilgili döngüden çıkılır. Sağlanamayan kıstasın bedeli olarak, çok büyük bir maliyet değeri penaltı değeri olarak parçacık maliyetine eklenir. Bu sayede parçacık, kısıtları sağlayamadığı başarısız bir duruma girmiş olsa da hareketine devam eder.

3. aşamada, tüm ünitelerin devrede-devre dışı konumları belirlenmiş durumdadır. Bu aşamada ünite güçleri P_u^t, ünite üretim kısıtları ve güç talebi kısıtı dâhilinde belirlenir. Bu belirleme klasik PSO yöntemi ile yapılır.

2. aşama sürecinde gerçekleşen fakat bahsedilmeyen bir husus, güç talebinin güncellenmesi $P_{demand}{}^{t}$, bu aşamada anlatılacaktır. 2. aşamada devreye alınan her ünite ile $P_{reserve}{}^{t}$ değerinin güncellendiği hatırlanmalıdır. $P_{reserve}{}^{t}$ güncellenirken, $P_{demand}{}^{t}$ de güncellenmiştir. Devreye alınan her ünitenin, üretebileceği minimum gücü P_{min_u} üreteceği varsayılmıştır, buna göre $P_{demand}{}^{t}$ güncellenmiştir. Sonuç olarak 3. aşamaya gelindiğinde belirlenmesi gereken aslında, devreye alınmış olan ünitelerin minimum güçlerinin üzerine kaç MW daha yüklenecekleridir.

1. Her bir ünite için hız değeri **(4.25)** hesaplanır

$$v2_u^t = w2_{i2} \cdot v2_u^t + c2_1 \cdot r_1 \cdot (x2_{pbest_u}^t - x2_u^t) + c2_2 \cdot r_2 \cdot (x2_{gbest}{}^t - x2_u^t) \quad \textbf{(4.25)}$$

Her bir ünite için konum değeri **(4.26)** hesaplanır.

$$x2_u^t = x2_u^t + v2_u^t \quad \textbf{(4.26)}$$

2. Devreye alınmış ünitelere üretim limitleri **(4.27)** dâhilinde güç ataması yapılır.

$$0 \le x2_u^t \le P_{max_u} - P_{min_u} \quad \textbf{(4.27)}$$

(4.26)'daki konum $x2_u^t$, ünitenin üretebileceği minimum gücün üzerine yüklenebileceği MW değeri olarak tanımlandığı için **(4.28)**'e göre limitler içinde kalması sağlanır.

$$0 \le x2_u^t \le P_{demand}{}^{t} \quad \textbf{(4.28)}$$

$P_{demand}{}^{t} = 0$ sağlandıktan sonra, sıradaki parçacıkların $x2_u^t$ değerleri 0'a atanır.

3. Güç dengesi sağlanamadıysa, güç dengesi sağlanır.

Tüm parçacıklar aldıkları $x2_u^t$ konumlarına göre yüklendikten sonra hala $P_{demand}{}^{t} = 0$ koşulu sağlanamadıysa. APSO aşamasında hesaplanmış olan $sig(v_u^t)$ değeri yüksek olan ünitenin devreye alınmasına benzer bir yaklaşıma gidilir. Algoritma tarafından tercih edilen parçacıklara daha fazla yüklenmeleri konusunda öncelik tanınır. Her ünite için **(4.29)**'daki formüle göre bir PSO yüklenme oranı belirlenir. Bu aşamada $xig(x_u^t)$ değeri maksimum olan ünite belirlenerek, üretilebileceği maksimum gücü üretmeye zorlanır.

$$xig(x2_u^t) = \frac{x2_u^t}{Pmax_u - Pmin_u} \qquad \textbf{(4.29)}$$

Üniteler, problemde tanımlandıkları sırayla, maksimum güç üretmeye zorlanabilecekken $xig(x_u^t)$ değeri büyük olanın buna zorlanması PSO'nun kararlı hareketlerini kuvvetlendirmek için tercih edilmiştir. Aksi takdirde, ilk sırada bulunan üniteler güç dengesinin sağlanmadığı her durumda maksimum güçlerini üretmeye zorlanırlar, bu da algoritmanın rastgeleliğini engeller. $xig(x_u^t)$ ve $sig(v_u^t)$'nin kullanıldığı ve kullanılmadığı, yani ünitelerin problemde veriliş sırasıyla PSO'ya dâhil olduğu yöntemler test çalışmalarında da incelenerek sonuçlar sıradaki bölümde verilmiştir

Böylelikle güç dengesi kısıtı **(4.30)** sağlanmış olur.

$$P_{load}{}^t = \sum_{u=1}^{NU} P2_u^t \cdot u_u^t \qquad \textbf{(4.30)}$$

4. aşamada problem maliyetleri hesaplanır. Bu aşamaya gelindiğinde, ünitelerin ilgili saatte ne kadar güç üretecekleri de belirlenmiş durumdadır. Öncelikle 3. aşamada hesaplanmış olan $x2_u^t$ değerleriyle, ünitelerin yüklenecekleri güçler **(4.31)** ile hesaplanır.

$$P2_u^t = Pmin_u + x2_u^t \qquad \textbf{(4.31)}$$

Maliyet hesaplamalarına başlamadan önce talep güç. kısıtının sağlanıp sağlanmadığının kontrolü **(4.32)**'ye göre yapılır. Devreye alınmış olan ünitelerin üretimlerinin toplamı toplam talep güce eşit değilse, parçacık maliyetine penaltı maliyeti olarak çok büyük bir değer atanır.

$$P_{demand}{}^t - \sum_{i=u}^{NU} P2_u^t \cdot u_u^t \neq 0 \quad ise \quad F2^t = 10^{19} \qquad \textbf{(4.32)}$$

1. Ünitenin ürettiği güç için yakıt maliyeti **(4.33)** ile hesaplanır.

$$F2_u^t(P2_u^t) = a_u + b_u \cdot P2_u^t + c_u \cdot P2_u^{t\,2} \qquad \textbf{(4.33)}$$

2. Ünitenin açılıp kapanma saatlerine göre sıcak veya soğuk başlangıç maliyetleri **(4.34)**'e göre belirlenir.

$$S_{x_u}^t = \begin{cases} S_{h_u} & eğer \quad t_u^t \le t_{cold_u} \\ S_{c_u} & aksi\,takdirde \end{cases} \qquad \textbf{(4.34)}$$

3. Yakıt maliyeti ile başlangıç maliyetleri **(4.35)**'teki gibi toplanarak, parçacık için saatlik toplam maliyet hesaplanır.

$$F2_{total}{}_{s2}^t = \sum_{u=1}^{NU} F2_u(P2_u^t) \cdot u_u^t + S_{x_u} \cdot u_u^t \qquad \textbf{(4.35)}$$

4. Parçacığın maliyeti daha önceki maliyetleriyle karşılaştırılır. Önceki maliyetten düşük bir maliyetse, parçacık en iyi maliyeti olarak **(4.36)**'daki gibi atanır. Konum vektörü, parçacığın en iyi konum vektörü olarak, güç vektörü de parçacığın en iyi güç vektörü olarak atanır.

$$\begin{aligned} F2_{pbest}{}_{s2}^t &= \begin{cases} F2_{total}{}_{s2}^t & eğer \quad F2_{total}{}_{s2}^t \le F2_{pbest}{}_{s2}^t \end{cases} \\ P2_{pbest}{}_{s2}^t &= \begin{cases} P2_{s2}^t & eğer \quad F2_{total}{}_{s2}^t \le F2_{pbest}{}_{s2}^t \end{cases} \\ x2_{pbest}{}_{s2}^t &= \begin{cases} X2_{s2}^t & eğer \quad F2_{total}{}_{s2}^t \le F2_{pbest}{}_{s2}^t \end{cases} \end{aligned} \qquad \textbf{(4.36)}$$

5. Parçacığın en iyi maliyeti sürünün en iyi maliyetiyle karşılaştırılır. Önceki en iyi maliyetten daha düşükse, sürü en iyi maliyeti olarak **(4.37)**'deki gibi atanır. Konum vektörü sürünün en iyi konum vektörü olarak, güç vektörü de sürünün en iyi güç vektörü olarak atanır.

$$\begin{aligned} F2_{gbest}{}^t &= \begin{cases} F2_{pbest}{}_{s2}^t & eğer \quad F2_{pbest}{}_{s2}^t \le F2_{gbest}{}^t \end{cases} \\ P2_{gbest}{}^t &= \begin{cases} P2_{pbest}{}_{s2}^t & eğer \quad F2_{pbest}{}_{s2}^t \le F2_{gbest}{}^t \end{cases} \\ X2_{gbest}{}^t &= \begin{cases} X2_{pbest}{}_{s2}^t & eğer \quad F2_{pbest}{}_{s2}^t \le F2_{gbest}{}^t \end{cases} \end{aligned} \qquad \textbf{(4.37)}$$

5. aşamada, APSO döngüsü içinde devrede olma durumları belirlenmiş olan parçacık ünitelerinin güç atamaları yapılarak maliyet karşılaştırması yapılır. PSO ile bulunmuş olan saatlik en iyi maliyet ve üretim değerleri APSO'ya saatlik maliyet ve saatlik üretimler olarak Şekil 4.10'daki gibi dâhil olur.

$$\begin{aligned} P_{gbest}{}^t &= \underbrace{\begin{bmatrix} P_1^t & P_2^t & . & . & P_{NU}^t \end{bmatrix}} \\ P2_{gbest}{}^t &= \begin{bmatrix} P2_1^t & P2_2^t & . & . & P2_{NU}^t \end{bmatrix} \end{aligned}$$

Şekil 4.10 : APSO parçacığı güç vektörü.

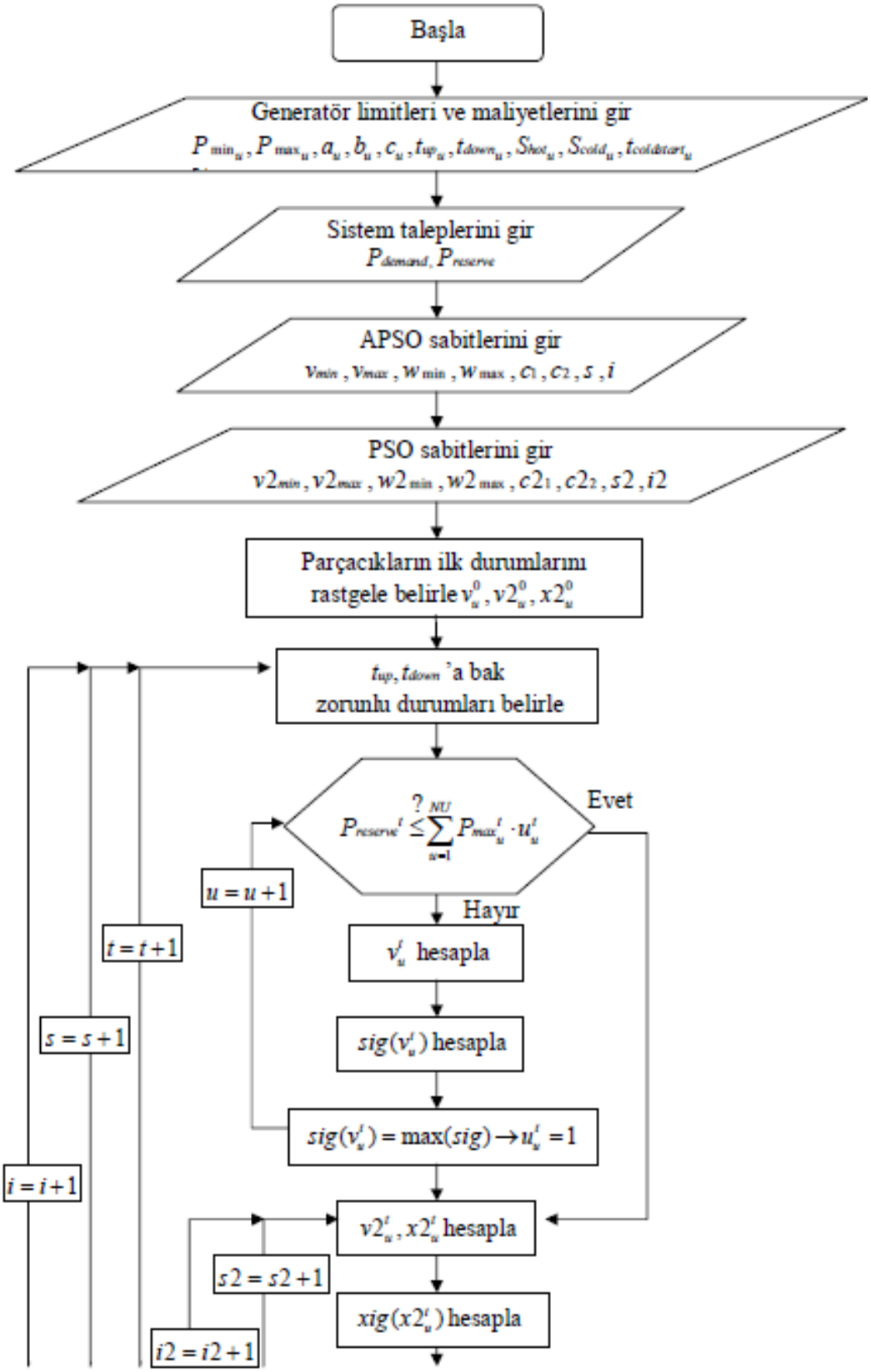
APSO algoritması her saat için döngüsünü tamamladığında, PSO algoritmasından gelen en iyi güç ve maliyetler, APSO parçacığının ilgili kısmına atanır. APSO algoritmasının içinde herhangi bir güç hesaplaması yapılmamakta, güç değerleri APSO içindeki PSO algoritmasının kullanması için alınmaktadır. Alınan maliyet değeri ise saatlik toplam maliyet olarak kullanılır. APSO içinde herhangi bir maliyet hesaplaması yapılmadığı halde, bu bilgi de PSO'ya bildirilmek ve toplam maliyet hesaplaması yapılırken kullanılmak üzere APSO içinde kaydedilir. İlgili parçacığın her saat için atamaları tamamlandığında parçacık en iyi maliyeti, en iyi güç ve en iyi konum vektörleri atamaları **(4.38)** ile yapılır.

$$\begin{aligned} F_{pbest_s} &= \begin{cases} F_{total_s} & eğer \quad F_{total_s} \le F_{pbest_s} \end{cases} \\ P_{pbest_s} &= \begin{cases} P_s & eğer \quad F_{total_s} \le F_{pbest_s} \end{cases} \\ X_{pbest_s} &= \begin{cases} X_s & eğer \quad F_{total_s} \le F_{pbest_s} \end{cases} \end{aligned} \qquad \textbf{(4.38)}$$

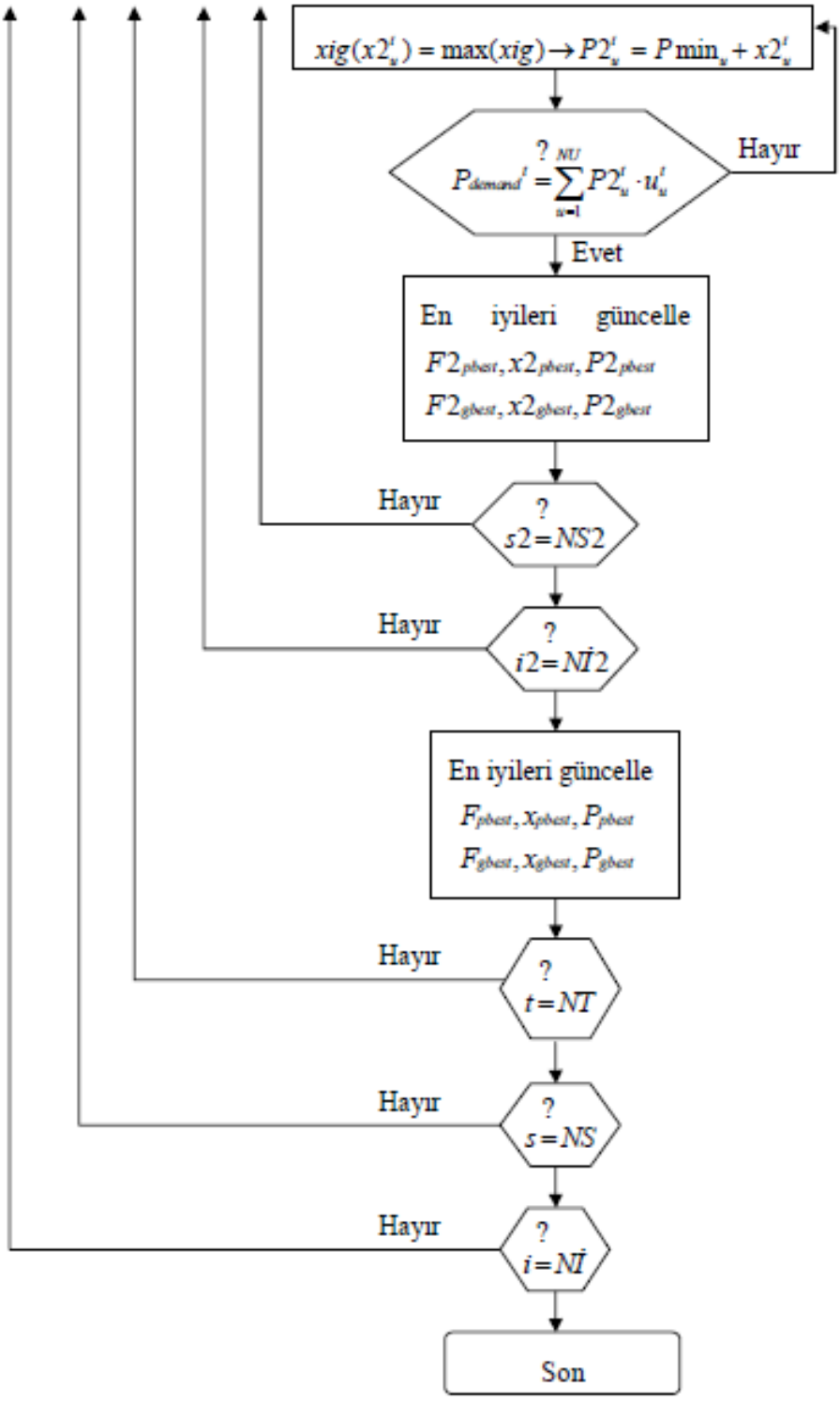
APSO, her parçacığın tüm saatleri için tamamlandığında ve toplam maliyetler karşılaştırılarak sürünün en iyi atamaları **(4.39)**'daki gibi yapılır.

$$\begin{aligned} F_{gbest} &= \begin{cases} F_{pbest_s} & eğer \quad F_{pbest_s} \le F_{gbest} \end{cases} \\ P_{gbest} &= \begin{cases} P_{pbest_s} & eğer \quad F_{pbest_s} \le F_{gbest} \end{cases} \\ X_{gbest} &= \begin{cases} X_{pbest_s} & eğer \quad F_{pbest_s} \le F_{gbest} \end{cases} \end{aligned} \qquad \textbf{(4.39)}$$

En iyi maliyet, güç ve konumların atanması ile ilgili iterasyon sona ermiş olur. Bir sonraki iterasyon başladığında, APSO parçacıklarının başlangıç konumları en son bulundukları konumdur. Parçacıkların değerlendirmeye alacakları en iyi değerler ise **(4.38)** ve **(4.39)**'daki güncellenmiş değerlerdir. Akış diyagramı Şekil 4.11'dedir.



Şekil 4.11 : Algoritma akış diyagramı.



Şekil 4. 11(devam): Algoritma akış diyagramı.

5. TEST SONUÇLARI

Ünite yüklenme problemi için geliştirilen PSO algoritması, Türkiye enterkonnekte şebekesinden alınmış küçük boyutlu bir sistemde ve literatürde sıklıkla kullanılan 1 test sisteminin genişletilmesiyle oluşturulmuş 3 sistem üzerinde test edilmiştir. Algoritmalarda parametre ayarlamalarının sistematik olarak ilerlemesi mümkün olmayıp, test sistemine ve diğer parametrelere bağlı olduğu için yalnızca en iyi sonucu veren algoritmanın parametreleri tablo halinde verilmiştir. Minimum maliyeti veren parametre kombinasyonu tablosunda, minimum maliyetin hesaplanma süresi de verilmiştir. Bu süre, algoritmanın çalışmaya başlamasından, minimum maliyetin hesaplandığı ilk ana kadar geçen süredir.

Test sistemlerinin çözümünde algoritmanın yakınsama becerisi, her iterasyon için bulunmuş olan maliyet değerleri grafiği oluşturularak incelenmiştir. Literatürde aynı sistemler üzerinde test edilmiş algoritmaların, buldukları en düşük maliyetlerle, PSO algoritmasıyla bulunan sonuçlar tek tablo içinde verilmiştir.

Literatürdeki çalışmalarda, optimizasyon sonuçlarından en iyi, en kötü ve ortalama maliyet değerlerinin sonuç olarak verildiği görülmüştür. Test sonuçlarının karşılaştırılmasında literatürden alınan maliyet değerleri herhangi bir değişiklik yapılmaksızın alınarak en iyi, en kötü ve ortalama değerler üzerinden karşılaştırmalar yapılmıştır.

Karşılaştırma tablolarında yöntem isimleri kısaltmaları ile verilmiş olan maliyet değerlerinin alındığı çalışmalar özet olarak aşağıda verilmiştir.

- (LR1) GLY1, gevşetilmiş Lagrange yöntemi uygulamasıdır[12].
- (LR2) GLY2, gevşetilmiş Lagrange yöntemi uygulamasıdır[40]
- GA1, klasik genetik algoritma uygulamasıdır[40].
- GA2, özel operatörlerin kullanıldığı bir genetik algoritma uygulamasıdır.[40].
- (GRA1) ARAA1 ve (GRA2) ARAA2, açgözlü rastgele adapte olabilen arama yöntemi uygulamasıdır [12].

- MA ve SMA, memetik algoritma uygulamalarıdır[40].
- (BDE1) AFEA1, ayrık farksal evrim algoritması uygulamasıdır [41]
- (BDE2) AFEA2, ayrık farksal evrim algoritması uygulamasıdır [42].
- ES, evrimsel stratejiler algoritması uygulamasıdır [41].
- (SSGA) DDGA, durgun durum genetik algoritma uygulamasıdır.[41]
- (ICGA) TKGA, tamsayı kodlanmış genetik algoritma uygulamasıdır [43].
- PSO, PSO ve APSO'nun entegre edildiği tez kapsamında geliştirilmiş algoritmanın uygulamasıdır.

5.1 Türkiye Şebekesi Test Sistemi

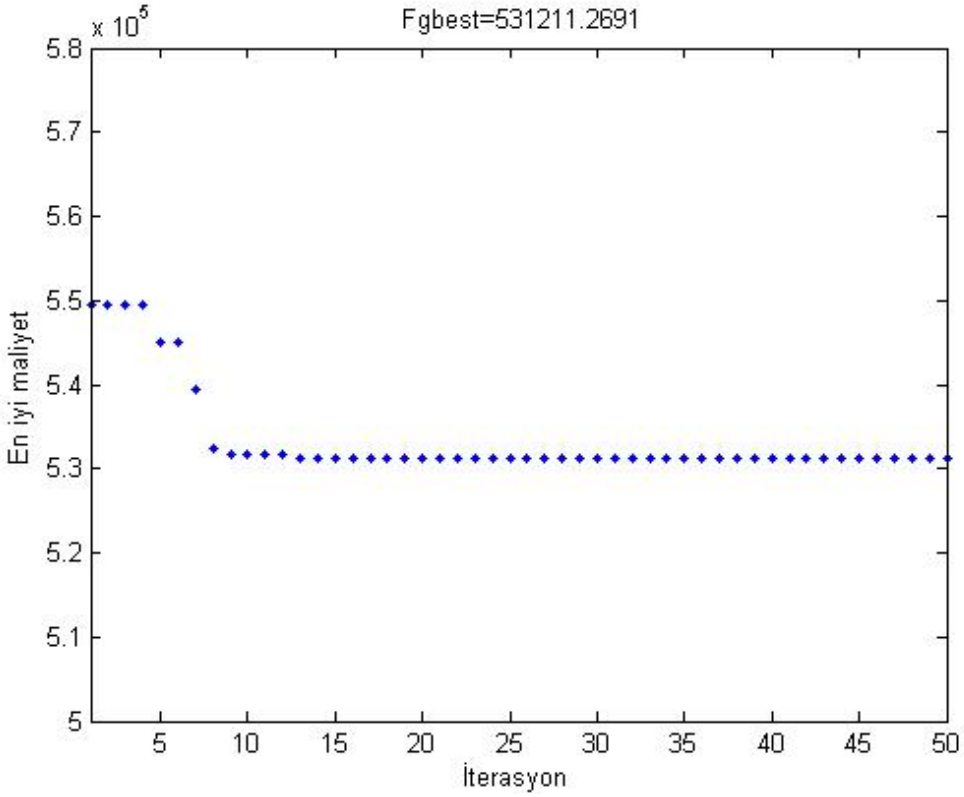
Algoritmanın test edildiği ilk sistem, Türkiye şebekesinin küçük bir parçasıdır. Sistem bilgileri [42]'den alınmış, bilgiler Çizelge A.1'de verilmiştir. 8 üretim ünitesinden oluşan sistem için 8 saatlik ünite yüklenme problemi çözülmüştür. PSO algoritmasının en büyük avantajı sistemlere kolay uygulanabilmesiyken, en büyük dezavantajı ise algoritma parametrelerinin belirlenmesinin zorluğudur. 8 üniteli sistem için hesaplanmış en iyi maliyeti veren parametreler Çizelge 5.1'de verilmiştir. PSO güven katsayıları, 1.1 olarak seçilmiştir. Literatürdeki tavsiyelerin aksine de olsa tamamen problemin yapısından ve tanımlanmış kısıtlardan kaynaklanan şimdiye kadar bulunmuş optimum değerlerdir.

Çizelge 5.1 : Türkiye test sistemi parametreleri.

t=32 sn	s	i	c1	c2	wmin	wmax
APSO	40	50	2	2	0.1	1
PSO	10	40	1.1	1.1	0.1	0.92

Parçacıkların lokal optimumlara takılmasının önüne geçmek için wmax, oldukça küçük seçilmiştir. İterasyon sayısının parçacık sayısından daha yüksek seçilmesinin parçacıkların daha kararlı hareket etmesini sağladığı görülmüştür.

Her iterasyonda güncellenen en iyi maliyetin değişim grafiği Şekil 5.1' de görülmektedir.



Şekil 5.1 : Türkiye test sistemi iterasyon sayısı-en iyi maliyet grafiği.

Parçacıklar, oldukça kararlı hareket etmiş, kısıtların fazlalığı sayesinde iterasyona başlar başlamaz oldukça iyi değerler üretmeye başlamış ve 12. iterasyonda en iyi maliyet değerine oturmuşlardır. Ünitelere atanmış olan güçler (MW) Çizelge 5.2' de görülebilir.

Çizelge 5.2 : Türkiye test sistemi ünite güçleri.

Saat	1. Ünite	2. Ünite	3. Ünite	4. Ünite	5. Ünite	6. Ünite	7. Ünite	8. Ünite
1.	0	0	0	549,4146	0	359,283	583,464	583,464
2.	0	0	740,7115	600	0	420	630	630
3.	844,4435	1107,051	1401,085	600	867,4213	420	630	630
4.	190	0	0	304,6653	210	197,25	321,7129	321,7129
5.	511,7254	0	873,5041	600	534,7705	420	630	630
6.	762,9347	0	1271,455	600	785,6099	420	630	630
7.	338,675	0	0	548,7038	364,1112	358,6021	582,774	582,774
8.	190	0	0	371,4747	214,21	241,5435	393,5047	393,5047

Çizelge 5.2'de ünitelere PSO yöntemi atanmış güçler verilmiştir. Güç değerlerinin ünitelerin minimum-maksimum değerlerinde farklı olmaları algoritmanın çalışmasının umut vaat ettiğinin göstergesidir. Parçacık güçleri güç talebi tamir mekanizmasına dâhil olduklarında, güçleri algoritmanın detaylı olarak anlatıldığı 4. bölümde bahsedilen sırayla, maksimum güçlerini üretmeye zorlanırlar. Oysaki PSO

parçacıkları kararlı hareketler yaparlarsa güçleri PSO formülasyonuna göre hesaplanır ve bu formülden çıkan değerler de Çizelge 5.2'de görüldüğü gibi ünite sınır güçlerinden farklı olur. Çizelge 5.3'de PSO' nun literatürdeki değerlerle karşılaştırıldığı tablo verilmiştir.

Çizelge 5.3 : Türkiye test sistemi maliyet karşılaştırması.

Algoritma	En İyi Maliyet	En Kötü Maliyet	Ortalama Maliyet
BDE2	530346	530346	530346
ES	530392	530392	530392
SSGA	530392	530392	530392
PSO	531211	531858	531340,4
BDE1	532142	-	-

5.2 10 Üniteli Test Sistemi

10 üniteden oluşan ikinci test sistemi literatürde, algoritmaların performansını test etmek için yaygın olarak kullanılan bir sistemdir. Sisteme ait bilgiler Çizelge A.2'de verilmiştir. Çok sayıda araştırmacının farklı yöntemleri test etmek için kullanmış olması, geniş kapsamlı bir karşılaştırma imkânı sağlamaktadır. Her ne kadar geliştirilmiş PSO yöntemi henüz, literatürde sıkça alıntılanan tablodaki değerlerle yarışacak seviyede olmasa da, hesaplama süresi ve sondan da olsa tabloya girmiş olması ile gelecekteki çalışmalar için olumlu bir öngörü yaratmaktadır.

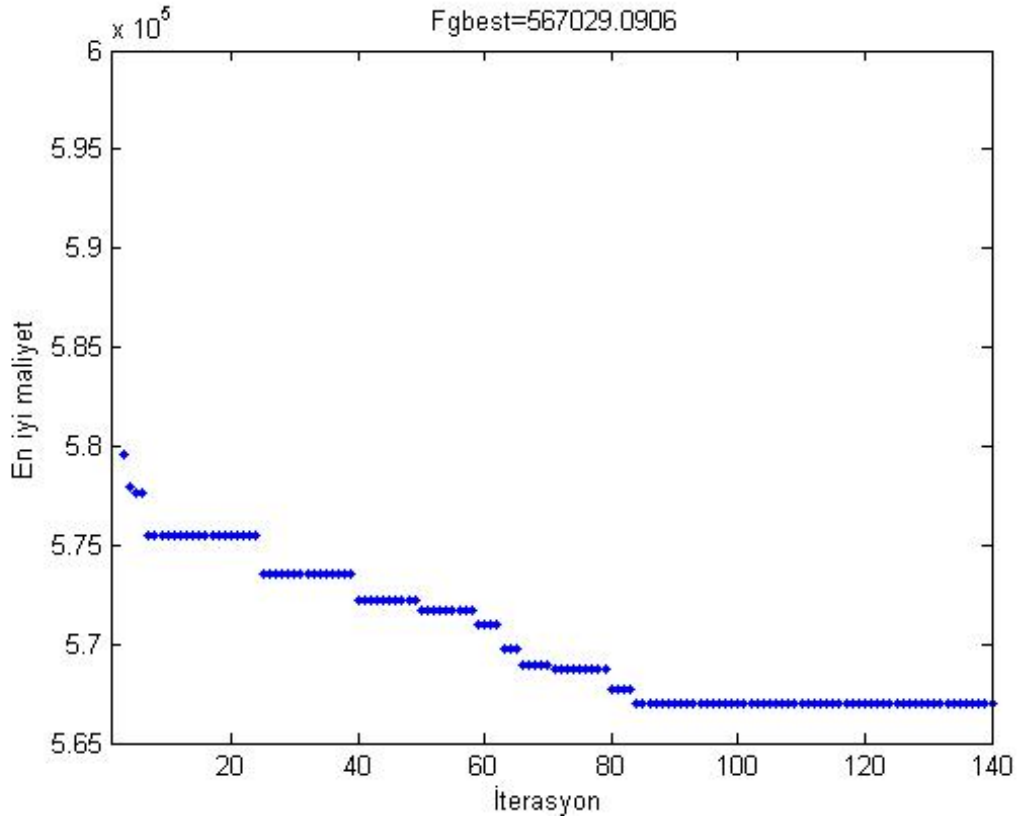
Çizelge 5.4'te 10 üniteli sistem için bulunan en iyi maliyeti veren algoritmada kullanılan APSO ve PSO parametreleri verilmiştir.

Çizelge 5.4 : 10 üniteli test sistemi parametreleri.

t=41sn	s	i	c1	c2	wmin	wmax
APSO	20	140	2.4	2.82	0.1	2.81
PSO	2	2	2.5	2	0.1	4.835

PSO döngüsü, APSO döngüsünün içinde yer aldığı için çifte iterasyonda hesaplanır. Özellikle ünite atanma durumlarının oturduğu belli bir iterasyon sayısından sonra işleyen her APSO iterasyon ve parçacık döngüsü, PSO' nun optimumu yakalaması için birer fırsattır. Geliştirilen algoritmanın belli bir iterasyon sayısından sonra ünite değişimini durdurduğu fark edildikten sonra bahsedilen çifte iterasyondan PSO'nun faydalanacağı hâlihazırda öngörülmüştür. Bu sebeple PSO'nun iterasyon ve parçacık sayıları, hesaplama süresini uzatmamak için olabildiğince küçük tutulmuştur.

10 üniteli test sistemi için iterasyon sayına göre hesaplanmış en iyi maliyetlerin yer aldığı grafik Şekil 5.2'de görülmektedir. 80. iterasyondan sonra birim durumları ve üretim değerleri neredeyse hiç değişmemiştir.



Şekil 5.2 : 10 üniteli test sistemi iterasyon sayısı-en iyi maliyet grafiği.

Çizelge 5.6'da test sistemi için hesaplanmış maliyetler verilmiştir. PSO'nun güvenilir bir deterministik yöntem olan GLY2(LR2)' yi minimum maliyet hesaplamasında geçmiş olması algoritma açısından iyi bir sonuçtur. Daha detaylı parametre belirleme çalışmaları yapıldığı takdirde, üst sıralara çıkması mümkün görünmektedir. Ayrıca PSO'nun en kötü maliyetinin, en iyi maliyetine çok yakın olması da dikkat çekmektedir.

Çizelge 5.5'te ünitelerin, yukarıda grafiği verilen algoritma sonrasında atanmış olan saatlik yükleri görülmektedir. Algoritma oluşturulurken, ünitelerin sıraylar devreye alınmasını önlemek için yapılan modifikasyonların başarılı olduğu çizelgede açıkça görülmektedir. Sıralı atama probleminin önüne geçilemediği takdirde, 3. saat gibi yeni bir ünitenin devreye alınma zorunluluğunun doğduğu saatlerde algoritma daha iyi olana yakınsama imkanı bulamadan 3. üniteyi devreye alma eğilimi gösterir. Sig ve xig değerleri ile bu durumun önüne geçilmiştir.

Çizelge 5.5 : 10 üniteli test sistemi ünite güçleri.

	1. Ünite	2. Ünite	3. Ünite	4. Ünite	5. Ünite	6. Ünite	7. Ünite	8. Ünite	9. Ünite	10. Ünite
1. Saat	455	245	0	0	0	0	0	0	0	0
2. Saat	455	295	0	0	0	0	0	0	0	0
3. Saat	455	370	0	0	25	0	0	0	0	0
4. Saat	455	455	0	0	40	0	0	0	0	0
5. Saat	455	455	65	0	25	0	0	0	0	0
6. Saat	455	455	130	35	25	0	0	0	0	0
7. Saat	455	455	130	85	25	0	0	0	0	0
8. Saat	455	455	130	130	30	0	0	0	0	0
9. Saat	455	455	130	130	85	20	25	0	0	0
10. Saat	455	455	130	130	162	33	25	10	0	0
11. Saat	455	455	130	130	162	73	25	10	10	0
12. Saat	455	455	130	130	162	80	58	10	10	10
13. Saat	455	455	130	130	162	33	25	0	0	10
14. Saat	455	455	130	130	85	20	25	0	0	0
15. Saat	455	455	130	130	30	0	0	0	0	0
16. Saat	455	455	95	20	25	0	0	0	0	0
17. Saat	455	455	45	20	25	0	0	0	0	0
18. Saat	455	455	130	35	25	0	0	0	0	0
19. Saat	455	455	130	115	25	20	0	0	0	0
20. Saat	455	455	130	130	162	33	25	10	0	0
21. Saat	455	455	130	130	49,28	55,71	25	0	0	0
22. Saat	455	455	130	35	0	0	25	0	0	0
23. Saat	455	425	0	20	0	0	0	0	0	0
24. Saat	455	345	0	0	0	0	0	0	0	0

Çizelge 5.6 : 10 üniteli test sistemi maliyet karşılaştırması

Algoritma	En İyi Maliyet	En Kötü Maliyet	Ortalama Maliyet
LR1	565825	N/A	N/A
GRA1	565825	-	-
GA2	565825	570032	-
BDE2	565827	566650	565965
MA	565827	566861	566453
ES	565827	571312	569199
GA1	565866	571366	567329
BDE1	566166	-	-
ICGA	566404	-	-
SMA	566686	567822	566787
PSO	567029	567436	567191.8
LR2	567663	N/A	N/A

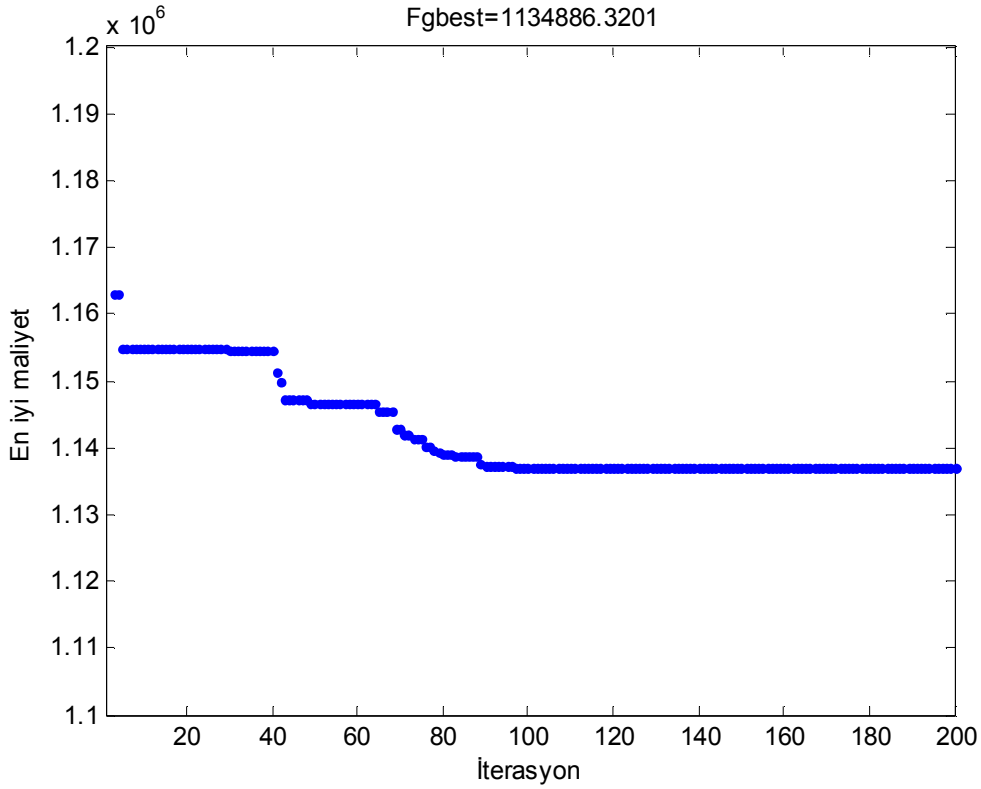
5.3 20 Üniteli Test Sistemi

20 üniteli test sistemi, 10 üniteli sistem datalarının, yeni bir 10 ünite sisteme dâhil olmuş gibi kullanılmasıyla elde edilmiştir. Literatürdeki yöntemlerde, 20 üniteli sistem için ünite ataması ve ünite yüklenmelerinin, ikiz olan üniteler için aynı olduğu görülmüştür. Bu PSO gibi rastgele hareket eden bir yöntem için ancak iterasyon sayısı çok büyük seçilirse mümkündür. Yapılan testler de bu durumu tasdiklemiştir. 20 üniteli sistem için Çizelge 5.7'de görülen parametrelerle algoritma çalıştırılmıştır

Çizelge 5.7 : 20 üniteli test sistemi parametreleri.

t=75sn	s	i	c1	c2	wmin	wmax
APSO	50	200	2.2	2.6	0.1	2.43
PSO	4	4	2.5	2	0.1	4.41

PSO parçacık ve iterasyon sayılarının küçük seçilmiş olması, 10 üniteli sistemde bahsedilmiş olan bir süre sonra APSO iterasyonlarının PSO iterasyonu gibi işlemeye başlamasından kaynaklanır. Hesaplama süresini uzatmamak için algoritmanın ilk iterasyonlarda bulması muhtemel olan iyi sonuçlardan feragat edilmiş ve optimum sonuçlara yaklaşmaktansa optimuma maliyet olarak yakın çözümün daha kısa süre içerisinde hesaplanması amaçlanmıştır. 75 sn'lik süre, bu açıdan hedeflenen sonuca ulaşıldığını göstermektedir. Maliyeti düşürmek için algoritma parametreleriyle daha detaylı çalışmalar yapılması gerekmektedir. 20 üniteli test sistemi için iterasyon sayısına göre hesaplanmış en iyi maliyetlerin yer aldığı grafik Şekil 5.3'te görülmektedir.



Şekil 5.3 : 20 ünite iterasyon sayısı-maliyet minimizasyonu grafiği

20 üniteli test sistemi için maliyet karşılaştırması Çizelge 5.8'de verilmiştir. Yöntemin, karşılaştırma tablosunda yer alan literatürde uygulanmış diğer yöntemlerle yarışabilmesi için parametre ayarlamasının gelecek çalışmalarda sistematik olarak yapılması şarttır. Buna rağmen bulunmuş olan maliyet hesaplama süresi de göz önüne alındığında uygun bir maliyettir.

Çizelge 5.8 : 20 üniteli test sistemi maliyet karşılaştırması

Algoritma	En İyi Maliyet	En Kötü Maliyet	Ortalama Maliyet
GA2	1126243	1132059	-
GRA2	1126805	-	-
ICGA	1127244	-	-
MA	1127254	1130916	1128824
GRA1	1128160	-	-
SMA	1128192	1128403	1128213
GA1	1128876	1131565	1130160
LR2	1129633	N/A	N/A
LR1	1130660	N/A	N/A
PSO	1134886	119163	114453

20 üniteli test sistemin ünite güçleri büyük bir tablo olduğu için Şekil A.3'te verilmiştir.

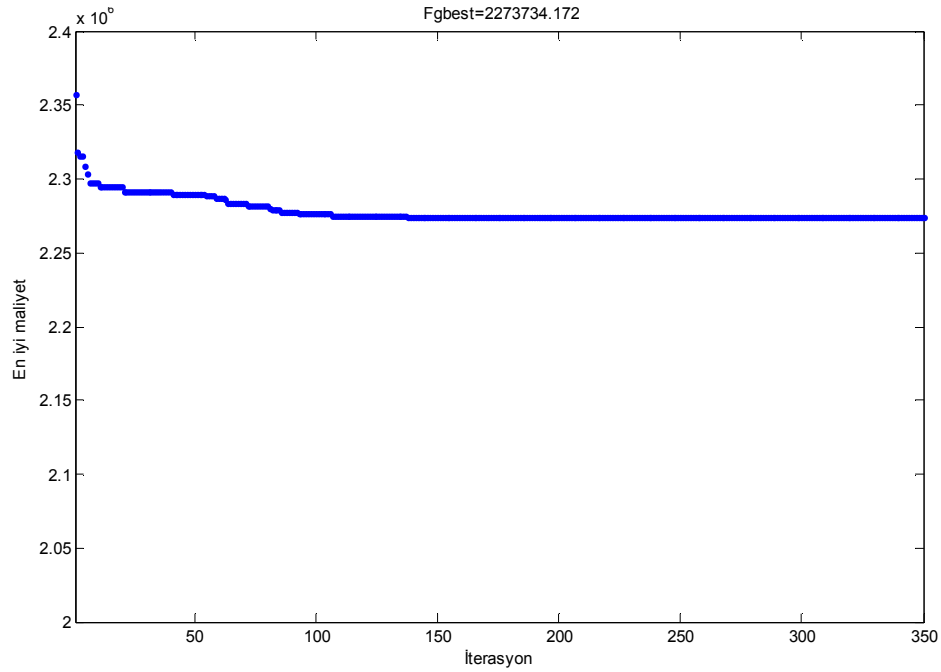
5.4 40 Üniteli Test Sistemi

40 üniteli test sistemi, 20 üniteli sistem bilgilerinin, aynı 20 ünite sisteme ek olarak dâhil olmuş gibi kullanılmasıyla elde edilmiştir. 40 üniteli sistem için kullanılan parametreler Çizelge 5.9'da görülmektedir.

Çizelge 5.9 : 40 üniteli test sistemi parametreleri.

t=153sn	s	i	c1	c2	wmin	wmax
APSO	100	350	2.1	2.81	0.1	2.37
PSO	6	6	2.6	2.1	0.1	4.7

Artan ünite sayısı nedeniyle ekonomik yük dağıtımını PSO'nun çok az sayıdaki parçacıklarıyla yapmak bu test sistemi için imkânsızdır. Bu nedenle zaman kaybı yaratsa da parçacık ve iterasyon sayıları arttırılmıştır. PSO'nun c1 katsayısının c2'den büyük seçilmesinin sebebi, algoritmanın böyle büyük sistemlerde her türlü lokal minimuma takılmasının önüne geçmektir. Bu sayede parçacıklar sürüden ziyade kendi hareketlerini daha çok önemseyip, daha başına buyruk hareketler yapacaklardır. Algoritmanın sonuçlanma süresinin çok önem teşkil etmediği problemlerde c1 katsayısını arttırmak, algoritmanın uzayın her yerinde daha detaylı bir arama yapacağının garantisidir. 40 üniteli test sistemi için iterasyon sayına göre hesaplanmış en iyi maliyetlerin yer aldığı grafik Şekil 5.4'te görülmektedir.



Şekil 5.4 : 40 ünite iterasyon sayısı-maliyet minimizasyonu grafiği

Çizelge 5.10'de sistemin, literatürdeki diğer çalışmalarda hesaplanmış maliyetlerle karşılaştırılması verilmiştir. Çok üniteli sistemlerde parametre çalışmalarının yapılması zorlaştığı için, tez kapsamında küçük sistemlerdeki kadar iyi sonuçlar elde etmek mümkün olmamıştır. Buna rağmen PSO yine hesaplama süresiyle dikkat çekmektedir.

Çizelge 5.10 : 40 üniteli test sistemi maliyet karşılaştırması

Algoritma	En İyi Maliyet	En Kötü Maliyet	Ortalama Maliyet
SMA	2249589	2249589	2249589
LR2	2250223	N/A	N/A
GA2	2251911	2259706	-
GA1	2252909	2269282	2262585
MA	2252937	2270361	2262477
ICGA	2254123	-	-
GRA2	2255416	-	-
LR1	2258503	N/A	N/A
GRA1	2259340	-	-
PSO	2273734	2276056	227893

6. SONUÇ

Tez çalışması kapsamında, termal ünite yüklenme problemi geliştirilen zincirleme PSO yöntemi ile çözülmüştür. Problemin ilk ve çözülmesi daha karmaşık olan alt problemi, ünite durumlarının belirlenmesinde geliştirilen ayrık PSO yöntemi kullanılmıştır. Saatlik bazda durumlar belirlendikten sonra ikinci alt problem olan ünite güç tahsisine geçilmiştir. Saatlik ekonomik güç dağıtımı alt problemi klasik PSO yöntemiyle çözülmüştür. Elde edilen maliyet ve hesaplama süreleri bazında literatürdeki yöntemlerle performans karşılaştırılması yapılmıştır.

6.1 Sonuç Değerlendirmesi

Ünite yüklenme problemine PSO yöntemi ile çözüm ararken ilk hedef klasik APSO yöntemini kullanmaktır. Tez kapsamında bahsedilen çalışmalar göstermiştir ki, ünite durumlarını belirlemek için kullanılan APSO yönteminin uygun sonuçlar üretebilmesi için altında çalışan güvenilir bir ekonomik yük dağıtımı algoritmasına ihtiyacı vardır. Bu ihtiyacı karşılamak için öncelikle rastgelelik temelli çeşitli arama prosedürleri geliştirilmiştir. Fakat bu prosedürlerin APSO'nun başarısını güçlendirmekten ziyade hesaplama süresini uzatarak, tatmin edici olmayan lokal optimumlarda takılı kaldığı görülmüştür. Bir sonraki denemede, yük dağıtımı algoritmasının da APSO'dan beslenmesi prosedürü denenmiştir. 4. Bölüm'de anlatılan ve APSO'da üretilen sigmoid fonksiyonu sonucu daha yüksek olan ünitenin daha çok yüklenmesi esas alınmıştır. Test çalışmaları göstermiştir ki, bu yöntemle elde edilen sonuçlar da tatmin edici değildir. Tezin amacı hesaplama süresini kısaltmak olarak belirlendiği için yük dağıtımı alt probleminin klasik PSO yöntemiyle yapılmasına karar verilmiştir.

Zincirleme APSO ve PSO algoritması ünite yüklenme problemine uygulanırken çeşitli yöntemler denenmiştir. İlk başta rastgeleliğe daha çok önem verilirken 4. Bölümde de bahsedildiği gibi algoritma geliştikçe, rastgelelikten ve problem veriliş sırasından tamamen uzaklaşarak, PSO yöntemlerinin daha çok güvendiği üniteye hesaplamada öncelik tanımak olarak özetlenebilecek bir prosedür benimsenmiştir.

Algoritma geliştirilmeye başlandığında öncelikli hedef kısa hesaplama süresi olarak belirlenmiştir. Arama uzayında rastgele ama kolektif yaklaşımla arama yapan parçacıkların sağladığı avantaj birçok uygulamada da görüldüğü gibi optimum sonucu elde etmekten ziyade, uygulanabilir bir sonucu minimum sürede sağlanmasıdır. Hâlihazırda geliştirilmiş evrimsel strateji temelli yöntemlerin çoğu birim yüklenme problemine uygulanmış durumdadır. Bu yöntemlerin ulaştığı maliyet sonuçları test sistemlerinde karşılaştırmaların yapıldığı tablolardan da görülebileceği gibi PSO'nun henüz rekabet edebileceği seviyelerin üstündedir. Buna rağmen PSO'nun uygulama kolaylığı ve hesaplama süresinde sağladığı avantaj da umut vaat etmektedir.

Yapılan literatür taramalarında görülmektedir ki, birim yüklenme problemine uygulanmış çeşitli hibrit PSO yöntemleri olmasına rağmen, bu yöntemlerin güvenilirliği tartışmaya açıktır. Maliyet sonucu açısından, yazarları tarafından GA sonuçlarıyla kıyaslanan bu yöntemlerin hemen hemen hepsinde hesaplama hatalarının yapıldığı görülmüştür. Bir kısmında maliyet değeri, başlangıç maliyetlerinden bağımsız olarak yalnızca yakıt maliyeti olarak tanımlanmıştır. Buna rağmen program sonuçları, başlangıç maliyetlerini de hesaba katan yöntemlerle karşılaştırılmıştır. Sonuç bazında incelendiğinde PSO ünite yüklenme problemi için avantajlı bir yöntem gibi görünse de eğer hedef optimum maliyete ulaşmaksa bu yalnızca bir yanılgıdan ibarettir.

Literatürde evrimsel temelli olmayan güvenilir bir PSO sonucu mevcut olmadığından, karşılaştırmalar evrimsel temelli yöntemlerle yapılmıştır. Bu aşamada PSO'nun tek avantajı hesaplama süresini kısaltmak olmuştur. Maliyet değerleri henüz GA ve ES temelli yöntemlerle yarışabilecek seviyede değildir.

6.2 Gelecek Çalışmalar için Öneriler

Algoritmanın test çalışmaları esnasında PSO sonuçlarının, atalet katsayısı, güven katsayıları, sürüdeki parçacık sayısı ve iterasyon değerlerine göre çok geniş bir aralıkta yer aldığı görülmüştür. Mevcut süre içinde özellikle küçük sistemler için bu parametreler optimize edilmeye çalışılmış fakat büyük sistemlerde hesaplama süresinin uzun olması sebebiyle yeterince inceleme yapılamamıştır. Bu bağlamda, gelecek çalışmalar için öneriler aşağıdaki gibi özetlenebilir,

- Büyük sistemlerde parametrelerin iyileştirilmesinin sonuç üzerindeki etkisinin araştırılması şiddetle önerilmektedir.
- Zincirleme PSO algoritması hesaplama süresi hesaba katılmadan tamamen maliyet minimizasyonu olarak çalıştırıldığında elde edilen sonuçların GA ile yarışabileceği düşülmektedir.
- PSO yönteminin büyük arama uzaylarında diğer yöntemlerden daha başarılı olduğu görülmüştür. Ünite yüklenme problemi için de parametre iyileştirilmesi testlerinin büyük sistemler üzerinde yapılmasının sonuçları iyileştireceği tahmin edilmektedir.
- Geliştirilen algoritmada PSO yöntemlerindeki parçacıkların hareketleri kısıtlarla sınırlandırılmıştır. Bunun yerine, parçacıkların hareketlerini ilk aşamada serbest bırakarak arama tamamlandıktan sonra tamir prosedürlerinin gerçekleştirilmesinin de denenmesi önerilmektedir.
- Çalışma kapsamında, problem spesifik parametre iyileştirilmesi yapılmasının çok zaman aldığı görülmüştür. Gelecek çalışmalarda PSO ile tümleşik bir parametre iyileştirmesi yönteminin de algoritmaya dâhil edilmesi tavsiye edilmektedir.
- Birim devreye alma alt probleminin APSO ile ekonomik yük dağıtımı probleminin ise deterministik bir yöntemle yapılması maliyetin iyileştirilmesi hedeflenen çalışmalar için önerilmektedir.

KAYNAKLAR

[1] **Miller R.H., Malinowski J.H.**, 1994. *Power System Operation*, 3rd Edition, Boston, MA: McGraw Hill, p. 63.

[2] **Shahidehpour M., Yamin H., Li Z.Y.,** 2002. *Market Operations in Electric Power Systems*, New York: Wiley, p. 115.

[3] **Hobbs B.F., Rothhopf M.H., Oneill R.P., Chao H.,** 1999. *The Next Generation of Electric Power Unit Commitment Models*, Kluwer Academic Publishers, p.5.

[4] **Wadhwa C.L.**, 2006. *Electrical Power Systems*, Fourth Edition, New Age. International, pp.179-185.

[5] **Senjyu T., Shimabukuro K., Uezato K., Funabashi T.**, 2003. "A fast technique for unit commitment problem by extended priority list", *IEEE Trans. Power Syst.,* vol. **18**, pp.882-888.

[6] **Cohen A.I., Yoshimura M.,** 1983. "A branch and bound algorithm for unit commitment", *IEEE Trans. Power Appl. Syst*., vol. **12**, pp.444-451.

[7] **Ouyang Z., Shahidehpour S.M.,** 1991. "An intelligent dynamic programming for unit commitment application", *IEEE Trans. Power Syst.,* vol. **6**, pp. 1203-1209.

[8] **Juste K.A., Kita H., Tanaka E., Hasegawa J.,** 1999. "An evolutionary programming solution to the. unit commitment problem", *IEEE Trans.Power Syst.,* vol. **14**, pp. 1452-1459.

[9] **Mantawy A.H., Abdel-Magid Y.L., Shokri Z.S.**, 1998. "A simulated annealing algorithm for unit commitment", *IEEE Trans. Power Syst.*, vol. **13**, pp. 197-204.

[10] **Sum-im. T., Ongsakul W.**, 2003. "Ant colony search algorithm for unit commitment", *Industrial Technology, IEEE International Conference*, vol. **1**, pp.72 -77.

[11] **Maifeld T.T., Sheble G.B.,** 1996. "Genetic-based unit commitment algorithm", *IEEE Trans. Power Syst.,* vol. **11**, pp. 1359-1370.

[12] **Kazarlis S.A., Bakirtzis A.G., Petridis. V.,** 1996. "A Genetic algorithm solution to the unit commitment problem", *IEEE Trans. Power Syst.,* vol. **11**, pp. 83-92.

[13] **Swarp K. S., Yamashiro S.,** 2002. "Unit commitment solution methodology using genetic algorithm", *IEEE Trans. Power Syst.,* vol. **17**, pp.87-91.

[14] **Sriyanyong P., Song, Y.H.,** 1995. "Unit commitment using particle swarm optimization combined with Lagrange relaxation", *IEEE Power Engineering Society General Meeting,* vol. **3**, pp. 2752-2759.

[15] **Gaing Z.L.,** 2003. "Discrete particle swarm optimization algorithm for unit commitment", *IEEE Power Engineering Society General Meeting,* vol. **1**, pp. 418-424.

[16] **Zhao B., Guo C.X., Bai B.R., Cao Y.J**., 2006. "An improved particle swarm optimization algorithm for unit commitment", *International Journal of Electrical Power and Energy Systems*, vol. **28**, pp. 482-490.

[17] **Ting T.O., Rao M.V.C., Loo C.K.**, 2006. "A novel approach for unit commitment problem via an effective hybrid particle swarm optimization", *IEEE Trans. Power Syst.,* vol. **21**, pp. 411-418.

[18] **Victoire T.A.A., Jeyakumar A.E.,** 2005."Unit commitment by a tabu search based hybrid optimization technique", *IEEE Proc. -Generation, Transmission and Distribution*, vol. **152**, pp. 563-574.

[19] **Michalewicz Z.,** 1990. *Genetic Algorithm + Data Structure = Evolution Program*, Springer Verberg Heidelberg, New York, pp.121-126.

[20] **Tong S.K, Shahidehpour S.M, Ouyang Z.,** 1991. "Short term unit commitment", *IEEE Trans. Power Syst*., vol. **6**, pp.1210 - 1216.

[21] **Momoh J.A.**, 2001. *Electric power system applications of optimization*, New York: Wiley, pp. 401-405.

[22] **Wood A.J., Wollenberg B. F.,** 1996. *Power generation, operation, and control*, New York: Wiley, 2nd ed., pp. 130-136.

[23] **Elgerd O.I.,** 1971. *Electric energy systems theory: an introduction*, McGraw-Hill Publishing, New York, pp. 294-296.

[24] **Nagrath I.J., Kothari D.P.,** 2001. *Modern power system analysis*, Tata McGraw-Hill Publishing, 2nd ed., pp. 333-334.

[25] **Baldick R.,** 1995. "The generalized unit commitment problem", *IEEE Trans. Power Syst*., vol. **10**, pp. 465-475.

[26] **Orero S., Irving M.,** 1997. "Large scale unit commitment using hybrid genetic algorithm", *Int. J. Electr. Power Energy Syst.,* vol. **19**, pp. 45-55.

[27] **Chakrabarti A., Hadler S.,** 2008. *Power system analysis: operation and control,* PHI Learning Private Ltd, 2nd ed., pp.154-156.

[28] **Wang C., Shahidehpour S. M.,** 1993. "Effects of ramp-rate limits on unit commitment and economic dispatch", *IEEE Trans. Power Syst.,* vol. **8**, pp. 1341-1350.

[29] **Kennedy J., Eberhart R.C.** 1995. "Particles swarm optimization", *Proc. IEEE International Conference on Neural Networks*, Piscataway, NJ, pp. 177-183.

[30] **Engelbrecht A.,** 2007. *Computational intelligence: an introduction*, John Wiley & Sons, England, pp. 289-299.

[31] **Shi Y.H., Eberhart R.C.,** 1999. "Empirical study of particle swarm optimization", *In: Proc. of Congress on Evolution Computation*, pp. 1945-1950.

[32] **Url-1** <*http://www.mae.ufl.edu/haftka/stropt/Lectures/PSO_introduction.pdf*>, alındığı tarih 02.05.2011.

[33] **Kennedy J., Eberhart R.C.**, 1997. "A discrete binary version of the particle swarm algorithm", *Proceedings of the Conference on Systems, Man, and Cybernetics*, Piscataway, N.J, pp. 4104–4108

[34] **Lee K.Y., Mohamed A.,** 2007. "Fundamentals of particle swarm optimization techniques", *Modern heuristic optimization techniques: theory and application to power systems*, pp. 76-82.

[35] **Yang S., Wang M., Jiao L.**, 2004. "A quantum particle swarm optimization", *In Proceedings of IEEE Congress on Evolutionary Computation 2004 (CEC 2004)*, pp 320-331.

[36] **Clerc M**., "Binary particle swarm optimisers: toolbox, derivations and mathematical insights",<*http://clerc.maurice.free.fr/pso/binary_pso*>, alındığı tarih Şub. 2005.

[37] **Yiqing L., Xigang Y., Yongjian L.**, 2007. "An improved PSO algorithm for solving nonconvex NLP/MINLP problems with equality constraints", *Computers and Chemical Engineering*, pp:153-162.

[38] **Angeline P.J.,** 1998. "Using selection to improve particle swarm optimization", *In Proc. IEEE Int. Conf. Computational Intelligence*, pp. 84-89.

[39] **Miranda V., Fonseca N.,** 2001. "EPSO Evolutionary self-adapting particle swarm optimization", *Internal report INESC Porto.*

[40] **Valenzuela J., Smith A.E.,** 2008. "A seeded memetic algorithm for large unit commitment problems," *Journal of Heuristics*, vol. **8**, pp.173-195.

[41] **Uyar A.S., Turkay B.,** 2008. "Evolutionary algorithms for the unit commitment problem", *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. **16**, pp. 239-255.

[42] **Keles A., Uyar A.S., Turkay B.**, 2007. "A differential evolution approach for the unit commitment problem", *ELECO: 5th International Conference on Electrical and Electronics Engineering*.

[43] **Damousis S., Bakirtzis A.G.**, 1996. "Genetic algorithm solution to the economic dispatch problem", *IEE Proceedings-C*, vol. **141**, pp. 377-382.

EKLER

EK A.1 : Türkiye test sistemi bilgileri
EK A.2 : 10 üniteli test sistemi bilgileri
EK A.3 : 20 üniteli test sistemi ünite güçleri

EK A.1

Çizelge A.1 : Türkiye test sistemi bilgileri

	Ünit 1	Ünit 2	Unit 3	Unit 4
Pmax (MW)	1120	1350	1432	600
Pmin (MW)	190	245	318	150
a ($/MWH)	6995.5	7290.6	6780.5	1564.4
b ($/MWH)	70063	72.592	5682	31.288
c ($/MWH)	0.0168	0.0127	0.0106	0.0139
min up (h)	8	1	1	10
min down (h)	2	0.5	0.5	3
hot start cost ($)	800	800	600	400
cold start cost ($)	1600	1600	1200	800
cold start hrs (h)	8	1	1	10
initial status (h)	4	-4	-4	-4

	Unit 5	Unit 6	Unit 7	Unit 8
Pmax (MW)	990	420	630	630
Pmin (MW)	210	110	140	140
a ($/MWH)	5134.1	1159.5	1697	1822.8
b ($/MWH)	6.232	33.128	32.324	3.472
c ($/MWH)	0.0168	0.021	0.013	0.0147
min up (h)	10	10	10	10
min down (h)	3	3	3	3
hot start cost ($)	500	400	400	400
cold start cost ($)	1000	800	800	800
cold start hrs (h)	10	10	10	10
initial status (h)	-4	-4	-4	-4

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

EK A.2

Çizelge A.2 : 10 üniteli test sistemi bilgileri

	Ünit 1	Ünit 2	Unit 3	Unit 4	Unit 5
Pmax (MW)	455	455	130	130	162
Pmin (MW)	150	150	20	20	25
a ($/MWH)	1000	970	700	680	450
b ($/MWH)	16.19	17.26	16.60	16.50	19.70
c ($/MWH)	0.00048	0.00031	0.002	0.00211	0.00398
min up (h)	8	8	5	5	6
min down (h)	8	8	5	5	6
hot start cost ($)	4500	5000	550	560	900
cold start cost ($)	9000	10000	1100	1120	1800
cold start hrs (h)	5	5	4	4	4
initial status (h)	8	8	-5	-5	-6

	Unit 6	Unit 7	Unit 8	Unit 9	Unit 10
Pmax (MW)	80	85	55	55	55
Pmin (MW)	20	25	10	10	10
a ($/MWH)	370	480	660	665	670
b ($/MWH)	22.26	27.74	25.92	27.27	27.79
c ($/MWH)	0.00712	0.00079	0.00413	0.00222	0.00173
min up (h)	3	3	1	1	1
min down (h)	3	3	1	1	1
hot start cost ($)	170	260	30	30	30
cold start cost ($)	340	520	60	60	60
cold start hrs (h)	2	2	0	0	0
initial status (h)	-3	-3	-1	-1	-1

Hour	Demand(MW)	Reserve(MW)	Hour	Demand(MW)	Reserve(MW)
1	700	70	1	1400	140
2	750	75	2	1300	130
3	850	85	3	1200	120
4	950	95	4	1050	105
5	1000	100	5	1000	100
6	1100	110	6	1100	110
7	1150	115	7	1200	120
8	1200	120	8	1400	140
9	1300	130	7	1300	130
10	1400	140	8	1100	110
11	1450	145	7	900	90
12	1500	150	8	800	80

EK A.3

Çizelge A.3 : 20 üniteli test sistemi ünite güçleri

	1. Ünite	2. Ünite	3. Ünite	4. Ünite	5. Ünite	6. Ünite	7. Ünite	8. Ünite	9. Ünite	10. Ünite
1. Saat	455	295	0	0	0	0	0	0	0	0
2. Saat	455	385	0	0	0	0	0	0	10	0
3. Saat	455	455	0	0	40	0	0	0	0	0
4. Saat	455	455	65	0	25	0	0	0	0	0
5. Saat	455	455	130	35	25	0	0	0	0	0
6. Saat	455	455	130	85	25	0	0	0	0	0
7. Saat	455	455	130	130	30	0	0	0	0	0
8. Saat	455	455	130	130	85	20	25	0	0	0
9. Saat	455	455	130	130	162	33	25	10	0	0
10. Saat	455	455	130	130	162	73	25	10	10	0
11. Saat	455	455	130	130	162	80	25	43	10	10
12. Saat	455	455	130	130	162	33	25	0	10	0
13. Saat	455	455	130	130	85	20	25	0	0	0
14. Saat	455	455	56,60	130	103,39	0	0	0	0	0
15. Saat	455	455	63,82	51,17	25	0	0	0	0	0
16. Saat	455	455	20	45	25	0	0	0	0	0
17. Saat	455	455	114,45	50,54	25	0	0	0	0	0
18. Saat	455	455	65,71	130	94,28	0	0	0	0	0
19. Saat	455	455	130	130	162	33	25	10	0	0
20. Saat	455	455	130	130	85	20	25	0	0	0
21. Saat	455	455	125	20	0	20	25	0	0	0
22. Saat	455	425	0	0	0	20	0	0	0	0
23. Saat	455	345	0	0	0	0	0	0	0	0
24. Saat	455	345	0	0	0	0	0	0	0	0

Çizelge A.3 (devam): 20 üniteli test sistemi ünite güçleri

	11. Ünite	12. Ünite	13. Ünite	14. Ünite	15. Ünite	16. Ünite	17. Ünite	18. Ünite	19. Ünite	20. Ünite
1. Saat	455	295	0	0	0	0	0	0	0	0
2. Saat	455	385	0	0	0	0	0	0	10	0
3. Saat	455	455	0	0	40	0	0	0	0	0
4. Saat	455	455	65	0	25	0	0	0	0	0
5. Saat	455	455	130	35	25	0	0	0	0	0
6. Saat	455	455	130	85	25	0	0	0	0	0
7. Saat	455	455	130	130	30	0	0	0	0	0
8. Saat	455	455	130	130	85	20	25	0	0	0
9. Saat	455	455	130	130	162	33	25	10	0	0
10. Saat	455	455	130	130	162	73	25	10	10	0
11. Saat	455	455	130	130	162	80	25	43	10	10
12. Saat	455	455	130	130	162	33	25	0	10	0
13. Saat	455	455	130	130	85	20	25	0	0	0
14. Saat	455	455	56,60	130	103,39	0	0	0	0	0
15. Saat	455	455	63,82	51,17	25	0	0	0	0	0
16. Saat	455	455	20	45	25	0	0	0	0	0
17. Saat	455	455	114,45	50,54	25	0	0	0	0	0
18. Saat	455	455	65,71	130	94,28	0	0	0	0	0
19. Saat	455	455	130	130	162	33	25	10	0	0
20. Saat	455	455	130	130	85	20	25	0	0	0
21. Saat	455	455	125	20	0	20	25	0	0	0
22. Saat	455	425	0	0	0	20	0	0	0	0
23. Saat	455	345	0	0	0	0	0	0	0	0
24. Saat	455	345	0	0	0	0	0	0	0	0

ÖZGEÇMİŞ

Ad Soyad:	Yasemin Zeybekoğlu
Doğum Yeri ve Tarihi:	Afyon 1986
Adres:	Küçükyalı Merkez Mah. Mektep Cad. No:64 D:4 Maltepe/İstanbul
Lisans Üniversite:	İstanbul Teknik Üniversitesi, Elektrik Mühendisliği
	2004-2008
Lise:	Çanakkale Fen Lisesi
	2001-2004