T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

İKİLİ OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜ İÇİN YAPAY ALG ALGORİTMASI TABANLI YENİ YAKLAŞIMLAR

Sedat KORKMAZ

DOKTORA TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Temmuz-2019
KONYA

TEZ KABUL VE ONAYI

Sedat KORKMAZ tarafından hazırlanan "İkili Optimizasyon Problemlerinin Çözümü İçin Yapay Alg Algoritması Tabanlı Yeni Yaklaşımlar" adlı tez çalışması 05/07/2019 tarihinde aşağıdaki jüri tarafından oy birliği ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda DOKTORA TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Prof. Dr. Erkan ÜLKER	.....
Danışman Doç. Dr. Mustafa Servet KIRAN	.....
Üye Doç. Dr. Oğuz FINDIK	.....
Üye Dr. Öğr. Üyesi Yüksel ÇELİK	.....
Üye Dr. Öğr. Üyesi Sait Ali UYMAZ	.....

Yukarıdaki sonucu onaylarım.

Prof. Dr. Hakan KARABÖRK
Enstitü Müdürü

Bu tez çalışması Selçuk Üniversitesi Bilimsel Araştırma Projeleri Koordinatörlüğü tarafından 18101002 numaralı proje ile desteklenmiştir.

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Sedat KORKMAZ

05.07.2019

ÖZET

DOKTORA TEZİ

İKİLİ OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜ İÇİN YAPAY ALG ALGORİTMASI TABANLI YENİ YAKLAŞIMLAR

Sedat KORKMAZ

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Mustafa Servet KIRAN

2019, 120 Sayfa

Jüri
Prof. Dr. Erkan ÜLKER
Doç. Dr. Oğuz FINDIK
Doç. Dr. Mustafa Servet KIRAN
Dr. Öğr. Üyesi Yüksel ÇELİK
Dr. Öğr. Üyesi Sait Ali UYMAZ

Son yıllarda optimizasyon problemlerinin çözümü için birçok yeni algoritma önerilmiştir. Bu algoritmalar genellikle doğadaki canlıların bireysel davranış şekillerinden, içgüdüsel hareketlerinden ve birbirleri ile aralarındaki akıllı etkileşimlerinden esinlenilerek geliştirilmişlerdir. Önerilen algoritmalar genellikle belirli bir problem çeşidine veya karakteristiğine sahip problemleri çözecek şekilde tasarlanmaktadır. Daha sonra yapılan çeşitli iyileştirmeler ve geliştirmeler neticesinde yönteme farklı karakteristikteki problemleri de çözebilme yeteneği kazandırılmaktadır. Örneğin, sürekli optimizasyon problemlerini (karar değişkenleri belirli bir aralıktaki tüm değerleri alabilen problemler) çözmek için önerilmiş bir algoritma, ikili optimizasyon problemlerini (karar değişkenleri sadece 0 yada 1 değerlerini alabilen problemler) çözecek şekilde geliştirilebilir. Bu tez kapsamında, karar değişkenleri sürekli değerler alan problemler için, doğada var olan mikroalglerin davranışlarından ilham alınarak geliştirilen Yapay Alg Algoritması (AAA) yeni ve özgün yöntemler geliştirilerek ikili optimizasyon problemlerini çözebilecek şekilde iyileştirilmiştir. Bu bağlamda 3 (üç) adet yeni ve özgün yöntem geliştirilmiştir.

Önerilen ilk yöntemde, popülasyondaki alg kolonileri ikili değerler ile ilklendirilerek yeni aday çözümlerin elde edilebilmesi için helisel hareket fazı, ikili değerler ile çalışabilecek şekilde yeniden uyarlanmıştır. Helisel hareket fazında, seçilen komşu çözümün rastgele belirlenmiş üç adet boyutunun değerleri, belirli bir olasılığa bağlı olarak mantıksal değil (not) işlemine tabi tutularak aday çözüme kopyalanmaktadır. Geliştirilen binAAA yönteminde adaptasyon parametresi hem adaptasyon sürecinin işletilip işletilmemesine karar vermek için, hem de bu süreçte etkilenecek boyutların belirlenmesi için kullanılmıştır. Önerilen binAAA yöntemi Kapasitesiz Tesis Yerleşim Problemleri (Uncapacitated Facility Location Problems-UFLP) üzerinde çalıştırılmış ve elde edilen sonuçlar binABC, BPSO, GA, DisABC, IBPSO ve ABCbin algoritmalarının sonuçlarıyla karşılaştırılmıştır. binAAA yöntemi, ayrık çözüm uzayında çalışması, hem yerel (local) hem de küresel (global) aramada yetenekli bir arama stratejisine sahip olması nedeni ile ikili optimizasyon problemlerini çözme konusunda, karşılaştırılan diğer yöntemler ile eşit veya daha iyi sonuçlar üretmektedir.

Önerilen ikinci yöntem, yeni aday çözümler üretebilmek için iki farklı güncelleme mekanizması barındırmaktadır. Bu mekanizmanın ilki, lojik özel veya (xor) operatörü kullanarak aday çözümler üretirken, ikinci mekanizmada ise ilk mekanizmadan edinilen bilgi kullanılarak stigmerjik (stigmergic) davranış temelinde yeni çözümler üretilmektedir. Önerilen SAAA (Stigmerjik AAA) yönteminde başlangıç çözümleri ikili değerler ile ilklendirilmiş, adaptasyon parametresi hem adaptasyon sürecinin işletilip işletilmemesine karar vermek için, hem de bu süreçte etkilenecek boyutların belirlenmesi için

kullanılmıştır. SAAA yönteminin performansı hem UFLP hem de nümerik fonksiyonlar üzerinde araştırılmıştır. UFLP seti üzerinde, BAAA yönteminin 2 farklı versiyonu, GA yöntemin 3 farklı versiyonu ve BPSO yöntemi ile kıyaslanmıştır. İkinci karşılaştırma için ise CEC2015 (bound constrained single-objective computationally expensive numerical optimization problems) test seti kullanılmış ve önerilen yöntemin performansının değerlendirilebilmesi için SBHS, HS, BLDE, BHTPSO-QI, GBABC, BQIGSA ve SabDE yöntemleri ile kıyaslanmıştır. Bütün sonuçlar genel olarak incelendiğinde, önerilen algoritma sadece düşük boyutlu problemlerde değil, aynı zamanda yüksek boyutlu problemler için de dengeli bir keşif ve sömürü kabiliyeti sunmaktadır. Ayrıca önerilen algoritmanın, çözüm kalitesi, yakınsama özellikleri ve sağlamlık açısından araştırmada ele alınan ikili optimizasyon problemlerini çözmede etkili ve verimli bir algoritma olduğu görülmektedir.

Tez çalışmasında PI-AAA (Popülasyon Etkili AAA) ismi ile önerilen üçüncü yöntemde de ilklendirme ikili değerler ile yapılmaktadır. Yeni ikili aday çözümlerin üretilebilmesi için popülasyon etkisi (population influence) yaklaşımı sunulmuş ve bu yaklaşım AAA'nın çalışmasıyla bütünleştirilmiştir. PI-AAA yönteminde, aday çözümler üretebilmek için, mevcut çözüm, en iyi çözüm ve alg kolonilerinden rastgele seçilen komşu çözümler kullanılmaktadır. Bu çözümler ile yapılan olasılık hesaplamaları neticesinde yeni aday çözümler üretilmektedir. Olasılık değerleri PI-AAA algoritmasına yönteme özgü kontrol parametresi olarak tanımlanmıştır. Önerilen yöntemin performansı için kontrol parametrelerinin değerlerinin önemli olması nedeniyle, parametrelerin etkileri analiz edilmiş ve bu parametreler için en uygun değerin belirlenmesi amaçlanmıştır. Temel AAA'daki adaptasyon aşaması, karar değişkenleri ikili değerler alan bireyler ile çalışabilmesi için yeniden uyarlanmıştır. Önerilen PI-AAA yönteminin performansı, ilk olarak UFLP seti üzerinde ABC, GA, PSO, EDA algoritmaları ile karşılaştırılmıştır. PI-AAA yönteminin etkinliğini kanıtlamak adına yapılan ikinci karşılaştırma CEC2015 problem seti üzerinde gerçekleştirilmiştir. Elde edilen sonuçlar, önerilen PI-AAA'nın karşılaştırmalarda daha iyi veya eşit performans gösterdiğini ve önerilen yaklaşımın rekabetçi bir ikili optimizasyon algoritması olduğunu göstermektedir.

Sonuç olarak bu tez kapsamında, ikili optimizasyon problemlerini çözmek için AAA yöntemi temel alınarak binAAA, SAAA ve PI-AAA isimleri ile 3 (üç) adet yeni ikili optimizasyon algoritması geliştirilmiştir. Deneysel çalışmalardan elde edilen sonuçlar incelendiğinde, önerilen yöntemler çözüm kalitesi, standart sapma ve yakınsama özellikleri açısından ikili optimizasyon problemlerini çözmede alternatif, rekabetçi ve sağlam oldukları görülmektedir. Bu bağlamda bu tez ile literatüre ikili optimizasyon alanında bir katkı sağlanmıştır.

Anahtar Kelimeler: ayrık optimizasyon, doğa esinli algoritmalar, ikili optimizasyon, ikilileştirme yöntemleri, kapasitesiz tesis yerleşim problemleri, sezgisel/meta-sezgisel yöntemler, sürü zekâsı, yapay alg algoritması

ABSTRACT

PhD THESIS

NOVEL APPROACHES BASED ON ARTIFICIAL ALGAE ALGORITHM TO SOLVE BINARY OPTIMIZATION PROBLEMS

Sedat KORKMAZ

Konya Technical University
Institute of Graduate Studies
Department of Computer Engineering

Advisor: Assoc. Prof. Dr. Mustafa Servet KIRAN

2019, 120 Pages

Jury
Prof. Dr. Erkan ÜLKER
Assoc. Prof. Dr. Oğuz FINDIK
Assoc. Prof. Dr. Mustafa Servet KIRAN
Asst. Prof. Dr. Yüksel ÇELİK
Asst. Prof. Dr. Sait Ali UYMAZ

In recent years, many novel algorithms have been proposed for solving optimization problems. These algorithms are usually developed by inspiring from intelligent interactions with each other, behavioral patterns and instinctual movements of living in nature. The proposed algorithms are usually designed to solve problems with a particular type of problem or characteristic. As a consequence of various enhancements and improvements made later on, the method is also able to solve the problems of different characteristics additionally. For instance, an algorithm proposed to solve continuous optimization problems (decision variables that can take all values in a given range) can be developed to solve binary optimization problems (decision variables that can only take values of 0 or 1). In this thesis study, Artificial Algae Algorithm (AAA) that developed for solving continuous optimization problems by inspiring behavior of microalgae that already exists in nature has been improved in order to solve binary optimization problems by establishing novel and unique methods. In this context, 3 (three) novel and unique methods have been developed.

In the first proposed method, in order to obtain new candidate solutions, the helical movement phase is re-adapted to work with binary values by initializing algae colonies in the population with binary values. In the helical movement phase, randomly determined three dimension values of selected neighbor solution are copied to the candidate solution and processing logical NOT function by depending on a particular probability. In the developed binAAA method, the adaptation parameter was used to make a decision whether the adaptation process should be operated or not and also to determine the dimensions to be affected in this process. The proposed binAAA method was investigated on Uncapacitated Facility Location Problems (UFLP) and the results were compared with the results of binABC, BPSO, GA, DisABC, IBPSO and ABCbin algorithms. The binAAA method produces equal or better results compared with other methods about solving binary optimization problems. This is because, the binAAA method works in discrete solution space and also it has a capable search strategy on both local and global search.

The second proposed method includes two different update mechanisms to produce new candidate solutions. The first of these mechanisms is to produce candidate solutions by using the logical XOR operator, while in the second mechanism, new solutions are produced based on the stigmergic behavior by using the information obtained from the first mechanism. In the proposed SAAA (Stigmergic AAA) method, initial solutions are initialized with binary values, and the adaptation parameter is used to make a decision whether the adaptation process should be operated or not, and also to determine the dimensions to be affected in this process. The performance of the SAAA method was investigated on both UFLPs and

numeric benchmark problems. The SAAA method was compared on the UFLP set with 2 different versions of the BAAA method, 3 different versions of the GA method and the BPSO method. For the second comparison, the CEC2015 (bound constrained single-objective computationally expensive numerical optimization problems) test set was used and it is compared with SBHS, HS, BLDE, BHTPSO-QI, GBABC, BQIGSA and SabDE methods in order to evaluate the performance of the proposed method. When all the results are examined in general, the proposed algorithm offers a balanced exploration and exploitation capability for not only low-dimensional problems and also for high-dimensional problems. Furthermore, it is seen that the proposed algorithm is an effective and efficient algorithm for solving the binary optimization problems covered in the research in terms of solution quality, convergence characteristics and robustness.

The initialization is been processed with binary values as well in the proposed third method that named PI-AAA (Population Influenced AAA) in the thesis study. A population influence approach was introduced and this approach was integrated into the work of AAA in order to produce new binary candidate solutions. In the PI-AAA method, the randomly selected neighbor solutions from the algae colonies, the current solution and the best solution are used in order to produce candidate solutions. New candidate solutions are produced as a result of the probability calculations made with these solutions. Probability values were defined as peculiar control parameters to PI-AAA algorithm. Due to importance of control parameters' values for the performance of the proposed method, the effects of the parameters are analyzed and it is aimed to determine the most suitable value for these parameters. The adaptation phase in the basic AAA is re-adapted in order to work with binary decision variable values. The proposed PI-AAA method's performance is first compared with the ABC, GA, PSO, EDA algorithms on the UFLP set. The second comparison is performed on the CEC2015 problem set in order to prove the effectiveness of PI-AAA method. The obtained results show that the proposed PI-AAA indicates better or equal performance in comparisons and also those results point out that this proposed approach is a competitive binary optimization algorithm.

In conclusion, in this thesis study, 3 (three) novel binary optimization algorithms have been developed with the name of binAAA, SAAA and PI-AAA based on AAA method in order to solve the binary optimization problems. When the results from experimental studies are examined, it is observed that the proposed methods are alternative, competitive and robust for solving binary optimization problems in terms of solution quality, standard deviation and convergence characteristics. In this context, a contribution has been made to the literature in the area of binary optimization with this thesis.

Keywords: artificial algae algorithm, binary optimization, binarization methods, discrete optimization, heuristic/meta-heuristic methods, nature-inspired algorithms, swarm intelligence, uncapacitated facility location problems

ÖNSÖZ

Doktora tez çalışmam boyunca değerli katkılarını ve desteğini esirgemeyen danışmanım Doç. Dr. Mustafa Servet KIRAN'a;

Emekli olması sebebiyle danışmanlığımdan ayrılan Prof. Dr. Ahmet ARSLAN'a;

Doktora tez izleme komitesinde yer alan ve tavsiyeleriyle çalışmalarıma destek veren Doç. Dr. Oğuz FINDIK'a ve Dr. Öğr. Üyesi Sait Ali UYMAZ'a;

Tez savunmamda yer alan jüri üyeleri Prof. Dr. Erkan ÜLKER'e ve Dr. Öğr. Üyesi Yüksel ÇELİK'e;

Beraber çalışma imkânı bulduğum ve bilgilerinden yararlandığım Dr. Öğr. Üyesi Ahmet BABALIK'a, Dr. Öğr. Üyesi Ersin KAYA'ya ve Bilgisayar Mühendisliği Bölümü'nün tüm öğretim elemanlarına;

Tez projesi kapsamında sağladığı ayni destekten dolayı Selçuk Üniversitesi Bilimsel Araştırma Projeleri Koordinatörlüğü'ne;

Çalışmalarım süresince gösterdikleri sabır, anlayış ve desteklerinden dolayı annem Aysel KORKMAZ'a, babam Bedri KORKMAZ'a, eşim Sibel KORKMAZ'a, oğlum Eymen KORKMAZ'a, kardeşim Hüseyin Bilal KORKMAZ'a ve tüm aileme;

İçtenlikle teşekkür ederim.

Sedat KORKMAZ
KONYA-2019

İÇİNDEKİLER

KISALTMALAR

AAA	: Yapay Alg Algoritması (Artificial Algae Algorithm)
ABC	: Yapay Arı Kolonisi (Artificial Bee Colony) Algoritması
ABCbin	: İkili ABC Algoritması (Binary ABC Algorithm)
ACO	: Karınca Kolonisi Optimizasyon (Ant Colony Optimization) Algoritması
AFSA	: Yapay Balık Sürüsü Algoritması (Artificial Fish Swarm Algorithm)
AM	: Açı Modülasyonu (Angle Modulation)
AMABC	: Açı Modülasyonlu ABC (Angle-Modulated ABC) Algoritması
AMDE	: Açı Modülasyonlu DE (Angle Modulated DE) Algoritması
AMPSO	: Açı Modülasyonlu PSO (Angle Modulated PSO) Algoritması
Ap	: Adaptasyon Parametresi (Adaptation Parameter)
APHS	: Uyarlanabilir Olasılıksal HS (Adaptive Probabilistic HS)
BA	: Yarasa Algoritması (Bat Algorithm)
BAAA-Tanh	: Tanjant Hiperbolik Transfer Fonksiyonu Kullanılmış İkili AAA
BAAA-Sig	: Sigmoid Transfer Fonksiyonu Kullanılmış İkili AAA
BAS	: İkili Karınca Sistemi (Binary Ant System)
BCO	: Arı Kolonisi Optimizasyon (Bee Colony Optimization)
BDE	: İkili Kodlanmış DE (Binary Encoding DE)
BHTPSO-QI	: İkili Hibrit Topolojili PSO Algoritması (Binary Hybrid Topology PSO Algorithm)
binAAA	: İkili AAA (Binary AAA)
binABC	: Lojik Kapı Tabanlı İkili ABC (A Logic Gate-Based Binary ABC) Algoritması
binDE	: İkili DE Algoritması (Binary DE)
BLDE	: İkili Öğrenme DE Algoritması (Binary Learning DE Algorithm)
BPSO	: İkili PSO (Binary PSO) Algoritması
BSO	: Arı Sürüsü Optimizasyon (Bees Swarm Optimization) Algoritması
BQIGSA	: İkili Kuantum-Esinli GSA (Binary Quantum-Inspired GSA)
CEC	: IEEE Evrimsel Hesaplama Kongresi (IEEE Congress on Evolutionary Computation)
CMA-ES	: Kovaryans Matris Adaptasyonu ile Evrim Stratejisi (Evolution Strategy with Covariance Matrix Adaptation)
CPSO	: Modulo-Bazlı İkili Parçacık Sürü Optimizasyonu
CS	: Guguk Kuşu Arama (Cuckoo Search)
D	: Boyut (Dimension)
DBHS	: Ayrık İkili HS (Discrete Binary HS)
DE	: Farksal Gelişim Algoritması (Differential Evolution Algorithm)
DED	: Dinamik Ekonomik Sevkiyat
DisABC	: Farklılık Tabanlı ABC (Dissimilarity Based ABC) Algoritması
DisDE	: Farklılık Tabanlı DE Algoritması (Dissimilarity Based DE Algorithm)
DOS	: Düzeltilmiş Ortalama Sıra
DPSO	: Ayrık PSO (Discrete PSO)
DSP	: Boyut Seçilme Olasılığı (Dimension Selection Probability)
EDA	: Dağılım Tahmin Algoritması (Estimation of Distribution Algorithm)
FA	: Ateşböceği Algoritması (Firefly Algorithm)
FPA	: Çiçek Tozlaşma Algoritması (Flower Pollination Algorithm)
FS	: Özellik Seçimi (Feature Selection)
GA	: Genetik Algoritma (Genetic Algorithm)
GA-SP	: Tek Noktadan Çaprazlamalı GA (GA with Single-Point Crossover)

GA-TP : İki Noktadan Çaprazlamalı GA (GA with Two-Point Crossover)
GA-UP : Bütün Noktalardan Çaprazlamalı GA (GA with Uniform Crossover)
GBABC : Genetik Operatör Tabanlı ABC (Genetic Operators Based ABC) Algoritması
GCBPSO : Garantili Yakınsamalı BPSO (Guaranteed Convergence BPSO)
GSA : Yerçekimsel Arama Algoritması (Gravitational Search Algorithm)
GWO : Gri Kurt Optimizasyon (Grey Wolf Optimization) Algoritması
HMCR : Harmonik Bellek Dikkate Alma Oranı (Harmony Memory Considering Rate)
HS : Harmoni Arama Algoritması (Harmony Search Algorithm)
IBPSO : İyileştirilmiş İkili PSO (Improved Binary PSO) Algoritması
INBPSO : İyileştirilmiş NBPSO (Improved NBPSO) Algoritması
KP : Sırt Çantası Problemleri (Knapsack Problems)
MaxFEs : Maksimum Fonksiyon Çağırma Sayısı (Maximum Number of Function Evaluations)
MBABC : Modifiye Edilmiş İkili ABC (Modified Binary ABC) Algoritması
MBABC-GC : Modifiye Edilmiş İkili ABC-Genetik Çaprazlama (Modified Binary ABC-Genetic Crossover) Algoritması
MBO : Bal Arıları Evliliği Optimizasyon (Marriage in Honey Bees Optimization) Algoritması
MBPSO : Modifiye Edilmiş İkili PSO (Modified Binary PSO) Algoritması
MKP : Çok Boyutlu Sırt Çantası Problemi (Multidimensional Knapsack Problem)
N : Popülasyondaki birey sayısı
NBPSO : Yeni İkili PSO (Novel Binary PSO) Algoritması
NFL : Ücretsiz Öğle Yemeği Yok Teoremi (No Free Lunch Theorem)
NNS : Komşu Çözümlerin Sayısı (Number of Neighbor Solutions)
normDE : Normalizasyon DE (Normalization DE)
NR : mantıksal değil olasılığı (logical not rate)
NS : Komşu Çözümler (Neighbor Solutions)
PAR : Akort Ayarlama Oranı (Pitch Adjustment Rate)
PI-AAA : Popülasyon Etkili AAA (Population Influenced AAA)
PSO : Parçacık Sürü Optimizasyon (Particle Swarm Optimization) Algoritması
S-bAFSA : Basitleştirilmiş İkili AFSA (Simplified Binary AFSA)
SA : Benzetimli Tavlama Algoritması (Simulated Annealing Algorithm)
SAAA : Stigmerjik AAA (Stigmergic AAA)
SabDE : Kendinden Uyarlamalı İkili DE (Self-Adaptive Binary DE) Algoritması
SBHS : Basitleştirilmiş İkili HS (Simplified Binary HS) Algoritması
TR : Toplam Sıra (Total Rank)
TS : Tabu Arama (Tabu Search) Algoritması
TSA : Ağaç-Tohum Algoritması (Tree-Seed Algorithm)
TSP : Gezgin Satıcı Problemi (Traveling Salesman Problem)
UC : Birim Yüklenme Problemi (Unit Commitment Problem)
UFLP : Kapasitesiz Tesis Yerleşim Problemi (Uncapacitated Facility Location Problem)
UM_1 : Güncelleme Mekanizması-1 (Update Mechanism-1)
UM_2 : Güncelleme Mekanizması-2 (Update Mechanism-2)
UMDA : Tek Değişkenli Marjinal Dağılım Algoritması (Univariate Marginal Distribution Algorithm)

UMSP	: Güncelleme Mekanizması Seçme Olasılığı (Update Mechanism Selection Probability)
VBA	: Sanal Arı Algoritması (Virtual Bee Algorithm)
WOA	: Balina Optimizasyon Algoritması (Whale Optimization Algorithm)

1. GİRİŞ

Artan ihtiyaçların kısıtlı kaynaklar ve belirli sınırlar dâhilinde karşılanmaya çalışılması problemi yaşamın her alanında karşımıza çıkmaktadır. Günlük hayatta karşılaştığımız bazı problemler için ürettiğimiz çözümler, farkında olarak veya olmadan yaptığımız bazı davranış şekilleri incelendiğinde, en iyiye ulaşma (optimizasyon) çabası canlıların en temel özelliklerinden birisidir. Yırtıcı bir hayvanın, karşılaştığı avın büyüklüğü ve avı ile arasındaki mesafeye göre ne zaman harekete geçeceğine karar vermesi, kaldırımda yürüyen bir insanın ağaçları, çöp kutusunu ve elektrik direğini göz önünde bulundurarak daha kısa bir yol arayışı (Şekil 1.1), alışveriş yapan bir kişinin sahip olduğu sınırlı maddi kaynaklar ile mümkün olan en kaliteli ve en fazla ürünü alma çabası, yaptığı alışverişten sonra aldığı ürünleri poşetlemek isteyen müşterinin poşetin taşıyabileceği yükü aşmadan en uygun bir şekilde ürünleri yerleştirmeye çalışması günlük hayatta karşılaştığımız, en iyiye ulaşma (optimizasyon) davranışı için verilebilecek örneklerden sadece birkaçıdır.

Şekil 1.1. Kaldırımda yürüyen insanların en kısa yol arayışı (pinkturtleemily, 2013)

Bir problemde var olan olası çözümler arasında en iyi çözümü bulma işlemine *optimizasyon* denilmektedir. 2011 yılında yayımlanan Türk Dil Kurumu Güncel Türkçe Sözlüğü'ne göre "optimizasyon" kelimesi matematik alanında "En uygun duruma getirme" olarak tanımlanmıştır (Türk Dil Kurumu, 2011). Optimizasyon problemleri, belirli sayıda girişleri ve çıkışları olan bir sistem olarak düşünülebilir. Bu sistemin matematiksel olarak modellenmesi gerekmekte ve bu matematiksel modele *amaç fonksiyonu* (objective function) denilmektedir. Amaç fonksiyonunun hesaplanmasında kullanılan giriş değerleri ise *karar değişkeni* (decision variable) olarak isimlendirilmiştir.

Optimizasyon, mühendislik tasarımından bilgisayar bilimlerine, çizelgelemeden (scheduling) finansal piyasalara, endüstriyel uygulamalardan tahmin sistemlerine ve günlük faaliyetlerimizden tatillerimizi planlamaya kadar birçok alanda karşımıza çıkmaktadır. Her zaman kârı en üst düzeye çıkarmak ya da maliyeti en aza indirmek amacındayız. Tatillerimizi planlarken bile, memnuniyetimizi en düşük maliyetle (veya ideal olarak ücretsiz) en üst düzeye çıkarmak isteriz. Aslında, karşılaştığımız her soruna en uygun çözümleri sürekli olarak aramaktayız, ancak bu tür çözümleri her zaman bulamayabiliriz. Karşılaştığımız birçok problemin optimizasyon problemi olduğunun farkına varmak, problem çözmeyi kolaylaştırmamaktadır. Basit gibi görünen birçok sorunun çözümü çok zor olabilmektedir. Satıcının, belirli sayıdaki şehirleri ziyaret etmeyi planladığı, kat ettiği toplam mesafeyi veya toplam seyahat maliyetini en aza indirgemek istediği Gezgin Satıcı Problemi (Traveling Salesman Problem-TSP) en bilinen zor problem örneklerinden birisidir. Bunun gibi zor problemleri çözmek için tam anlamıyla etkili bir algoritma maalesef günümüzde mevcut değildir. Son yirmi yılda geliştirilen modern optimizasyon tekniklerinin büyük bir kısmı genellikle meta-sezgisel yöntemler olmuştur ve neredeyse tüm bilim ve mühendislik alanlarının yanı sıra endüstriyel uygulamalarda da uygulanmaktadır (Yang, 2014).

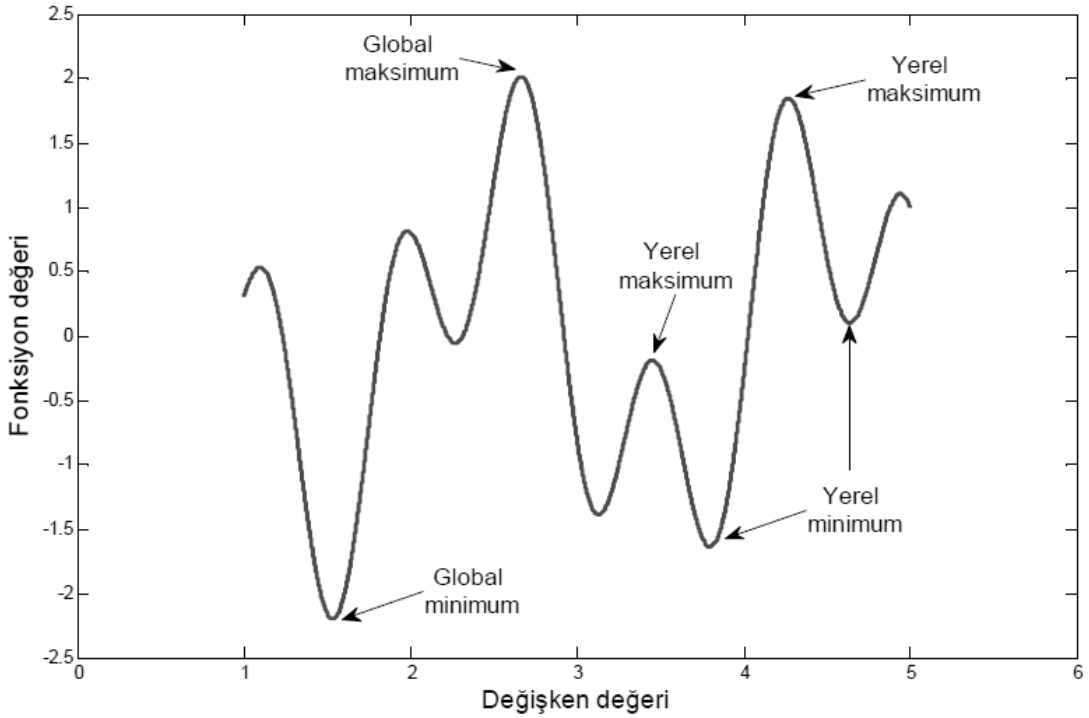
Optimizasyon farklı biçimlerde olabilmektedir. Birçok matematiksel ve istatistiksel yöntem aslında optimizasyonun farklı bir formudur. Örneğin, veri işlemede kullanılan en küçük kareler yöntemi, öngörülen değerler (matematiksel modeller ile) ve gözlenen değerler arasındaki farkı en aza indirmeye çalışır. Sonlu farklar yöntemleri gibi tüm büyük sayısal yöntemler, gerçek çözümler ve tahmini çözümler arasındaki farkı en aza indiren bazı yaklaşımlar bulmayı amaçlamaktadır (Yang, 2010b).

Optimizasyon problemlerinde hedef, amaç fonksiyonunu en büyük (global maksimum) veya en küçük (global minimum) yapacak karar değişkenlerinin değerlerinin bulunmasıdır. En büyük amaç fonksiyonu değerinin arandığı optimizasyon problemine

maksimizasyon problemi, en küçük amaç fonksiyonu değerinin arandığı optimizasyon problemine ise *minimizasyon problemi* denilmektedir.

Örnek bir optimizasyon probleminin matematiksel ifadesi Denklem 1.1'de ve grafiği Şekil 1.2' de verilmiştir (Yaman, 2014).

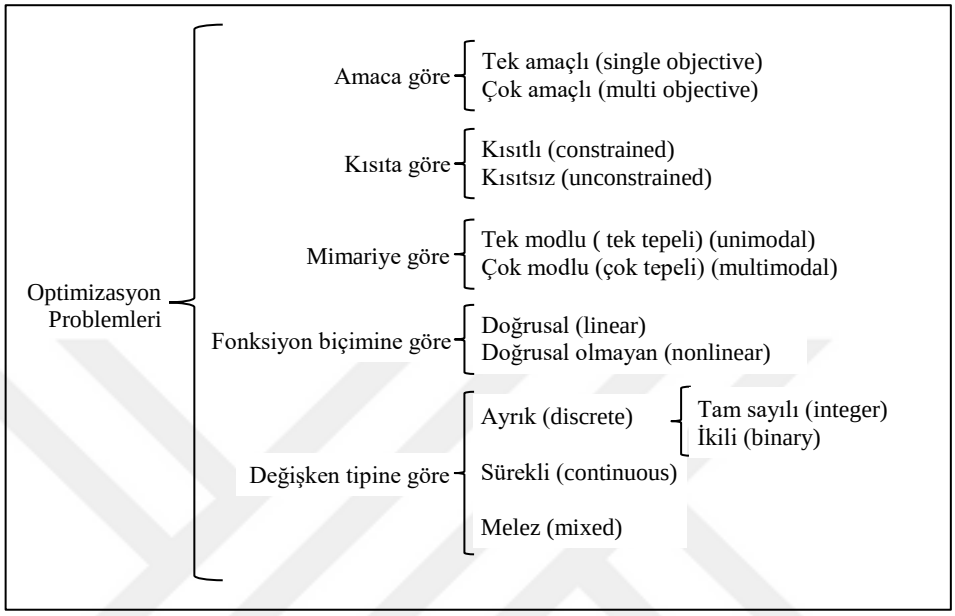
$$f(x) = 1.2\sin(\pi x) - \cos(2.6\pi x)\,, 1 \le x \le 5 \qquad (1.1)$$



Şekil 1.2. Örnek bir optimizasyon problemi grafiği (Yaman, 2014)

Denklem 1.1'de, $f(x)$ amaç fonksiyonu ve x ise [1,5] aralığında tanımlanmış probleme ait karar değişkenidir. Denklem 1.1' de verilen amaç fonksiyonunun grafiği Şekil 1.2' de verilmiştir. Karar değişkeninin tanım aralığı içerisindeki en yüksek amaç fonksiyon değerine *global maksimum*, en düşük amaç fonksiyon değerine ise *global minimum* denilmektedir. Karar değişkeninin tanım aralığı içerisinde belirli bir aralıktaki en yüksek amaç fonksiyonu değerine *yerel (lokal) maksimum* ve en düşük amaç fonksiyonu değerine ise *yerel (lokal) minimum* denilmektedir. Örnekte de görüldüğü üzere, herhangi bir optimizasyon probleminde, birden fazla yerel maksimum ve yerel minimum noktaları bulunabilmektedir.

Yapılan literatür araştırması sonucunda, optimizasyon problemleri; amaca göre, kısıta göre, mimariye göre, fonksiyon biçimine göre ve değişken tipine göre olmak üzere beş ana başlıkta sınıflandırılmıştır. Sınıflandırma şeması Şekil 1.3'de verilmiştir (Yang, 2010b).



Şekil 1.3. Optimizasyon problemlerinin sınıflandırılması (Yang, 2010b)

Optimizasyon problemlerini amaç sayısı açısından değerlendirdiğimizde *Tek Amaçlı* (single objective) ve *Çok Amaçlı* (multi objective) olmak üzere iki gruba ayırabiliriz. Denklem 1.1'de verilen örnekte amaç fonksiyonu tek olmakla birlikte farklı problemlerde amaç fonksiyonu birden fazla olabilmektedir. Örneğin bir aracın hedef noktasına en kısa yoldan, en kısa zamanda gitmesi gerekirken aynı zamanda yakıt tüketiminin de minimum olması istenebilir. Bazı durumlarda amaçlar birbirleri ile çelişebilmektedir. Amaçlardan birinin maksimizasyonu istenirken diğer amacın minimize edilmesi beklenebilmektedir (Kıran, 2014). Mühendislik problemleri başta olmak üzere pek çok gerçek dünya problemi birden fazla amaca sahiptir (Özkış, 2018).

Karar değişkenlerine ait herhangi bir sınırlamanın bulunmadığı problemler *Kısıtsız Optimizasyon Problemleri* sınıfına, bir veya daha fazla sınırlamanın bulunduğu problemler ise *Kısıtlı Optimizasyon Problemleri* sınıfına girmektedir. Karar değişkenlerinin sınırlandırılmaları eşitlik veya eşitsizlik şeklinde olabilmektedir.

Kısıtlı optimizasyon problemlerinde, arama uzayında uygun ve uygun olmayan bölgeler bulunabilmektedir. Bu nedenle kısıtlı optimizasyon problemlerini çözebilmek

için optimizasyon algoritmasının uygun olmayan bölgeleri de dikkate alarak yeni çözümler üretebilmesi gerekmektedir (Yaman, 2014).

Optimizasyon problemleri mimari yapısına göre incelendiğinde, *Tek Modlu* (tek tepeli) (unimodal) ve *Çok Modlu* (çok tepeli) (multimodal) olarak sınıflandırılmaktadır. Benzersiz bir global optimum değeri için sadece bir adet çukur veya bir adet tepe noktası mevcut ise bu optimizasyon problemi tek modludur. Bunun aksine, problem grafiksel olarak incelendiğinde birden fazla çukur veya tepe içeriyorsa çok modlu optimizasyon problemleri sınıfındandır. En derin çukur veya en yüksek tepe noktaları problemin global optimum noktasını, bunların dışındaki tepe veya çukurlar ise yerel optimum noktalarını ifade etmektedir (Çelik, 2013). Optimizasyon probleminin çok modlu olması çözümünü zorlaştıran bir etkendir.

Optimizasyon problemlerinin bir başka sınıflandırması ise sınırlama fonksiyonlarının ve amaç fonksiyonlarının doğrusallığına göre yapılmaktadır. Optimizasyon problemi, doğrusal sınırlama ve amaç fonksiyonlarına sahip ise bu problem *Doğrusal* (linear), sınırlama veya amaç fonksiyonlarından herhangi biri doğrusal değil ise problem *Doğrusal Olmayan* (nonlinear) problem sınıfındandır (Karaboğa, 2014).

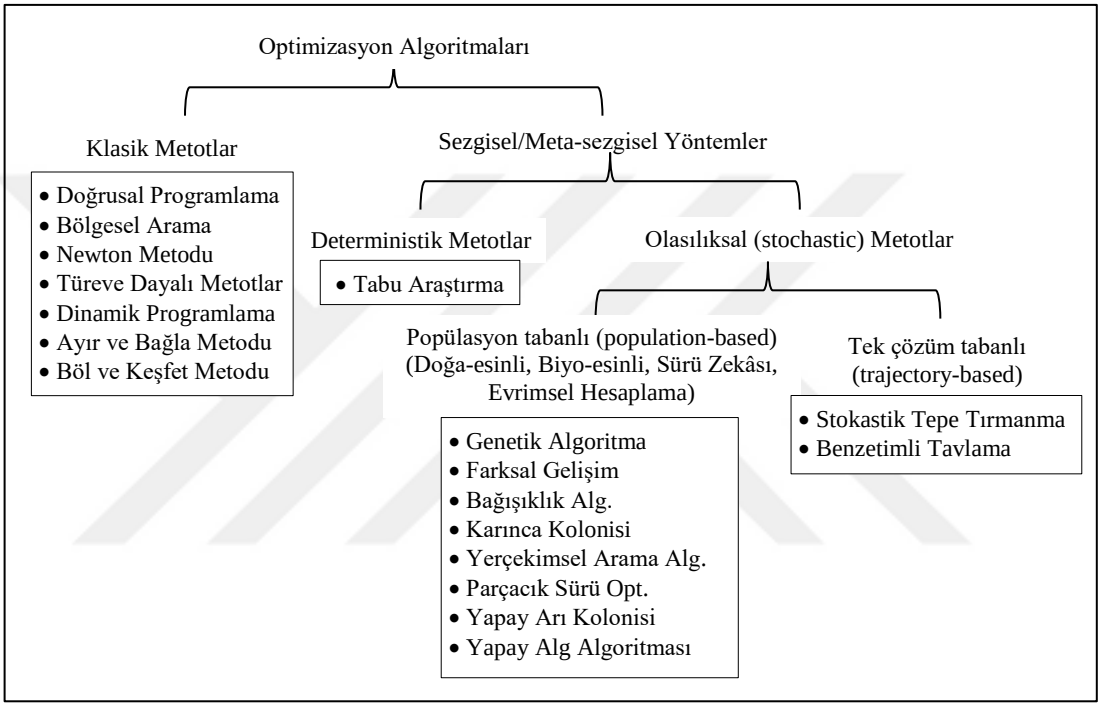
Optimizasyon problemleri, karar değişkenlerinin tiplerine göre de sınıflandırılmaktadır. Probleme ait karar değişkenleri verilen bir set içerisinden belli değerler alıyorsa problem *Ayrık* (discrete) sınıfındadır (Akay, 2009). Ayrık niceliklerin optimum şekilde gruplanması, düzenlenmesi, sıraya konulması veya seçilmesi problemi *ayrık optimizasyon* problemi olarak adlandırılır (Karaboğa, 2014). Karar değişkenlerinin alabileceği değerler sürekli (gerçek) değerler ise bu tür problemlere *Sürekli* (continuous) optimizasyon problemi denir. Bununla birlikte, değişken tipine göre *Melez* (mixed) sınıfına dahil olan bazı optimizasyon problemlerinin karar değişkenlerinin bir kısmı ayrık değerler alırken, diğer bir kısmı ise sürekli değerler alabilmektedir.

Ayrık optimizasyon problemleri *Tam Sayılı* (integer) ve *İkili* (binary) olmak üzere iki gruba ayrılmaktadır. Tam sayılı optimizasyon problemlerinde karar değişkenleri tam sayılı değerler alırken, ikili optimizasyon problemlerinde karar değişkenleri sadece 0 (sıfır) veya 1 (bir) değerini alabilmektedir.

Ayrık optimizasyon problemlerinde sıralamanın (permütasyon) önemli olduğu problemlere kombinasyonel (combinatorial) optimizasyon problemleri denilmektedir. Kombinasyonel tipindeki problemlere verilebilecek örneklerin en başında iş planlama problemi ve gezgin satıcı problemi gelmektedir. Bir diğer ayrık optimizasyon problemi tipi ise sıranın önemli olduğu seçim tabanlı problemlerdir. Bu tip problemlere örnek

olarak araç rotalama problemi, en kısa yol problemi ve minimum kapsayan ağaç problemi gösterilebilir. Sıranın önemli olmadığı seçim (alt küme) tabanlı problemlere örnek olarak ise maksimum klik problemi, sırt çantası problemi ve kenar kapsama problemi verilebilir (Cebeci, 2012).

Yapılan literatür araştırması sonucunda, optimizasyon algoritmaları Şekil 1.4'de verilen şemadaki gibi sınıflandırılmıştır (Akay, 2009; Yang, 2010b; Kıran, 2014; Köse, 2017).



Şekil 1.4. Optimizasyon algoritmalarının sınıflandırılması (Akay, 2009; Yang, 2010b; Kıran, 2014; Köse, 2017)

Klasik teknikler, hızlı olmaları, deterministik olmaları ve tam sonucu vermelerinden dolayı optimizasyonda bazı durumlarda avantajlı olmaktadırlar (Akay, 2009). Türevlenebilir ve sürekli fonksiyonların çözümünde klasik teknikler kullanışlı yöntemlerdir (Kıran, 2014). Ancak gerçek dünya problemleri genellikle türevlenemeyen ve/veya doğrusal olmayan amaç fonksiyonlarına sahip olduklarından dolayı klasik teknikler ile çözülmelerinde zorluklarla karşılaşılmaktadır. Optimizasyon probleminin boyutsallığı arttıkça klasik tekniklerin hesaplama süreleri kabul edilebilir sınırları aşmaktadır. Bu yüzden klasik metotlar, genellikle çok az boyuta sahip problemlerin çözümünde kullanılabilmektedir. Ayrıca klasik teknikler genellikle problem bağımlıdırlar.

Doğrusal olmayan (nonlinearity) ve çok modlu (multimodality) problemler, tepe tırmanma (Hill-Climbing) yöntemi gibi çoğu geleneksel yöntemleri verimsiz kılmaktadır. Daha da zor olan bir başka sorun da karar değişkenlerinin sayısı arttığında yani n değerinin çok büyük olduğunda ortaya çıkmaktadır. Ayrıca, büyük ölçekli karmaşıklıkla doğrusal olmamanın birleşimi işleri daha da zorlaştırmaktadır. Gezgin satıcı probleminde olası kombinasyonların sayısı n! (n faktöriyel) olmaktadır. Birbirinden farklı olası tur kombinasyonlarının sayısı ise n!/2n'dir (Özkan, 2010). Gezilecek şehirlerin sayısı 100 (n=100) olduğunda ise, bu kombinasyonların sayısı $n!/2n \approx 4{,}6x10^{155}$ gibi astronomik bir ihtimali ortaya çıkmaktadır. Günümüzde dünyanın en iyi süper bilgisayarları bile saniyede yaklaşık $3x10^{15}$ kayan noktalı sayıları işleyebilmektedir. Dünyadaki bütün bilgisayarları kullanabilsek bile 100 şehirli gezgin satıcı probleminin tüm kombinasyonlarını aramak evrenin ömründen çok daha uzun zaman alacaktır. Tüm olası kombinasyonları aramanın pratik bir yol olmadığı açıkça görülmektedir. Bu nedenle, yeterli derecede verimli alternatif yöntemlerin kullanımı zorunlu hale gelmektedir (Yang, 2010b).

Klasik algoritmalar deterministiktir, yani aynı ilklendirme (başlangıç noktası) ile hep aynı yolu izleyerek aynı sonuca ulaşırlar. Olasılıksal sezgisel/meta-sezgisel optimizasyon algoritmaları ise rastgeleliğe sahiptirler. Her zaman optimum sonucu bulma garantisi vermezler ama kabul edilebilir süreler içerisinde kaliteli çözümler sunabilmektedirler. Daha kısa süreler içerisinde daha yüksek oranda kaliteli çözümler bulabilmek için meta-sezgisel algoritmalar üzerine yapılan çalışmalar hâlâ devam etmektedir (Uymaz, 2015).

Sezgisel ve meta-sezgisel algoritmalar, klasik teknikler ile çözülemeyen veya çözümü çok uzun süreler alan problemlerle başa çıkmak için tasarlanmışlardır. Bu algoritmaların çoğu, doğadan (nature-inspired), biyolojik sistemlerden (bio-inspired), evrimsel süreçlerden (Evolutionary Computation) ve sürü zekâsından (swarm intelligence) esinlenilerek geliştirilmiştir. Doğa, milyonlarca hatta milyarlarca yıl boyunca çeşitli zor problemleri çözmektedir. Sadece en uyumlu olanlar hayatta kalmaktadır. Benzer şekilde, sezgisel algoritmalar problemleri çözmek için deneme-yanılmayı, öğrenme ve uyarlamayı kullanmaktadır. Her zaman en iyi çözümü bulmalarını bekleyemeyiz, ancak kabul edilebilir sürelerde yeterince iyi çözümler bulmalarını bekleyebiliriz. Modern meta-sezgisel algoritmalar, çözümü zor optimizasyon problemlerinin geniş bir yelpazesi için iyi sonucu garanti edebilmektedir. Ancak, optimizasyon biliminde, "no free lunch" ismiyle bilinen bir teorem karşımıza çıkmaktadır

(Wolpert ve Macready, 1997). “no free lunch” teoremine göre, eğer A algoritması bazı problemler için B algoritmasından daha iyi ise, B'nin diğer problemler için A'dan daha iyi performans gösterebileceği belirtilmektedir. Yani, evrensel olarak tam anlamıyla verimli bir algoritma mevcut değildir. Optimizasyon ve algoritma geliştirme araştırmalarının temel amacı, verilen bir optimizasyon görevi için en uygun ve en verimli algoritmaları tasarlamak ve/veya seçmektir (Yang, 2010b).

Sezgisellik, karmaşık bir probleme makul bir zamanda kabul edilebilir çözümler üretmek için deneme-yanılma yöntemini kullanan çözüm üretme stratejisidir. Problemin karmaşıklığı, mümkün olan her çözümü veya kombinasyonu aramayı imkânsız kılmakta, kabul edilebilir bir zaman diliminde iyi ve uygulanabilir çözümler bulmak amaçlanmaktadır. En iyi çözümlerin bulunabileceğinin garantisi yoktur ve bir algoritmanın problemi çözmede işe yarayıp yaramayacağını önceden bilememekteyiz. Asıl amaç, çoğu zaman işe yarayacak, kaliteli çözümler üretebilecek, verimli ve pratik bir algoritma arayışıdır (Yang, 2010b).

Yaklaşık çözümler üreten olasılıksal yöntemler, sezgisel ve meta-sezgisel olmak üzere iki sınıfa ayrılabilirler. Sezgisel yöntemler, geliştirildiği problemin bilgilerini kullandığı için bu probleme bağımlıdır ve sadece geliştirildiği problem için iyi sonuçlar vermesi beklenir. Meta-sezgisel yöntemler ise, problem bağımlılığı olmayıp, ufak uyarlamalar ile birçok optimizasyon problemlerinin çözümünde kullanılabilen genel yöntemlerdir (Cebeci, 2012). Ayrıca meta-sezgisel yöntemler, geçmişteki davranışlardan elde edilen bilgi ve tecrübeleri kullanarak, sonraki davranışları şekillendirebilme yeteneğine sahip olabilmektedir.

Literatürde var olan optimizasyon algoritmaların bir çoğu ilk olarak sürekli optimizasyon problemlerini çözmek için önerilmişleridir. Bunlardan bazıları daha sonra, ayrık optimizasyon problemlerini çözebilmek için uyarlanmışlardır. Sürekli değerler alan optimizasyon problemlerinde karar değişkenleri, eğer bir kısıt yok ise, limitler içerisinde sonsuz sayıda değeri kabul edebilmektedir. Fakat ayrık optimizasyon problemlerinde karar değişkenleri tam sayılar veya kategorik değişkenler olabilmekte, hatta ikili optimizasyon problemlerinde karar değişkenlerinin alabileceği değerler sadece 0 (sıfır) yada 1 (bir) olabilmektedir.

Sürü zekâsı temelli algoritmalarının birçoğunda olası çözüm, bir dizi gerçek değişken olarak kodlanır. Vektördeki herhangi bir pozisyonun (boyutun) güncellenmesi diğerlerinden bağımsız olarak yapılabilmektedir. Güncellenen değerler bu boyut için önceden tanımlanmış aralık içinde ise aday çözüm geçerlidir. Bununla birlikte,

kombinasyonel (combinatorial) problemler için, çözüm tamsayı değerlerinin bir permütasyonu olması gerektiğinden bu yaklaşım uygun olmamaktadır. Bu nedenle, kombinasyonel problemler için bir çözümü temsil eden vektörde gerçekleştirilecek güncellemeler permütasyon koşuluna uygun olmak zorundadır (Krause ve ark., 2013).

İkilileştirme yöntemlerine genel olarak bakıldığında temel problemin aday çözüm üretme stratejisinde ortaya çıktığı görülmektedir. Genetik algoritma gibi yöntemler çözüm gösterimi için bit dizilerini kullandığından dolayı ikili problemlere uygulanması kolaydır. Çaprazlama ve mutasyon operatörleri ile bit dizileri üzerinde yapılan işlemler sayesinde aday çözüm üretebilmektedir. Genetik algoritma ve benzeri yöntemler haricindeki algoritmalar karar değişkenleri sürekli değerler alan problemleri çözmek için tasarlandığından dolayı, ikili problemleri çözebilmeleri için yöntemlerin yeniden ele alınması ve ikili değerler ile çalışacak şekilde uyarlanmaları gerekmektedir. Yapılması gereken değişiklikler yöntemin karakteristiğine ve çalışma şekline göre farklılık gösterebilmektedir. Örneğin Yapay Alg Algoritmasındaki (AAA) ana adımlardan biri olan adaptasyon sürecinde, ikili değerler ile çalışabilmesi için bazı değişikliklerin yapılması gerekmekte, helisel hareket fazında kullanılan kesme kuvveti parametresinin ise sürekli değerler üretmek için kullanıldığından dolayı belki de yöntemden çıkarılması gerekebilmektedir. Ama genel olarak bakıldığında bütün doğa-esinli veya meta-sezgisel yöntemlerin ortak noktalarının aday çözümlerin üretilmeye çalışıldığı bölümler olduğunu söyleyebiliriz. İkilileştirme yöntemlerinin bazıları yöntemin çalışma karakteristiğinde değişiklik yapma esasına göre çalışırken, bazı yöntemler ise sadece probleme ait amaç fonksiyonun hesaplanması sırasında devreye girmektedir. Yani yöntem sürekli değerlere sahip karar değişkenleri ile çalışırken, probleme ait amaç fonksiyonu hesaplanmasından hemen önce bu sürekli değerler transfer fonksiyonları gibi metotlar vasıtası ile ikili değerlere çevrilerek kullanılmakta ve yöntemin ikili problemleri çözebilmesi sağlanmaktadır.

İkilileştirme yöntemleri açısından literatür incelendiğinde, yaygın olarak kullanılan yöntemler aşağıdaki gibidir (Banitalebi ve ark., 2016):

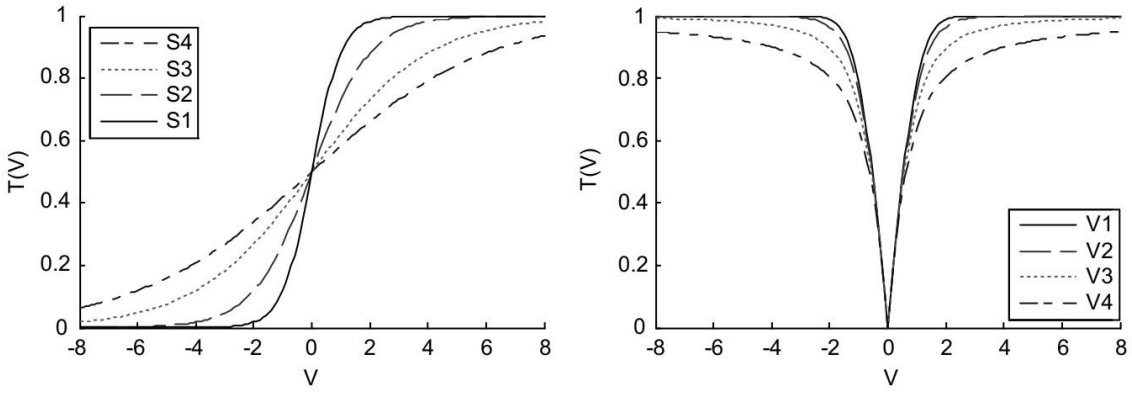
Transfer fonksiyonu: Kennedy ve Eberhart (1997) sürekli değerler alan karar değişkenlerini ikili değerlere dönüştürebilmek için sigmoid fonksiyonundan yararlanmışlardır. Bu teknik ilk olarak Parçacık Sürü Optimizasyon (Particle Swarm Optimization-PSO) algoritmasında kullanılmıştır. PSO algoritmasındaki hız vektöründeki değerler sigmoid fonksiyonu vasıtası ile 0 ya da 1 değerlerine

dönüştürülmüştür. İlerleyen zamanlarda farklı transfer fonksiyonları önerilmiş ve farklı algoritmaların ikilileştirmeleri için kullanılmıştır (Lee ve ark., 2008; Nezamabadi-pour ve ark., 2008; Beheshti ve ark., 2015). Sürekli değerleri ikili değerlere dönüştürmek için kullanılan transfer fonksiyonlarından bazıları Çizelge 1.1'de verilmiştir (Mirjalili ve Lewis, 2013).

Çizelge 1.1. Transfer fonksiyonları (Mirjalili ve Lewis, 2013)

S-Şekilli Transfer Fonksiyonları		V-Şekilli Transfer Fonksiyonları	
İsmi	Fonksiyonu	İsmi	Fonksiyonu
S1	$T(x) = \frac{1}{1+e^{-2x}}$	V1	$T(x) = \lvert \mathrm{erf}(\frac{\sqrt{\pi}}{2}x) \rvert = \lvert \frac{\sqrt{2}}{\pi} \int_0^{(\sqrt{\pi}/2)x} e^{-t^2}\, dt \rvert$
S2 (Geem ve ark., 2001)	$T(x) = \frac{1}{1+e^{-x}}$	V2 (Wang ve ark., 2010a)	$T(x) = \lvert \tanh(x) \rvert$
S3	$T(x) = \frac{1}{1+e^{(-x/2)}}$	V3	$T(x) = \lvert (x)/\sqrt{1+x^2} \rvert$
S4	$T(x) = \frac{1}{1+e^{(-x/3)}}$	V4	$T(x) = \lvert \frac{2}{\pi} \arctan(\frac{\pi}{2}x) \rvert$

Yukarıda Çizelge 1.1'de verilen transfer fonksiyonlarının grafiksel gösterimleri aşağıda Şekil 1.5'de verilmiştir.



Şekil 1.5. Transfer fonksiyonlarının grafiksel gösterimleri (Mirjalili ve Lewis, 2013)

Açı modülasyonu (Angle Modulation-AM): Telekomünikasyon endüstrisinde sinyal işleme alanında kullanılan açı modülasyonu tekniğinin temeli, ikili dizileri oluşturmak için birleşik sinüs ve kosinüs fonksiyonlarını kullanmaya dayanmaktadır. Fonksiyonun 4 (dört) adet sürekli değerler alan parametresi olmakla birlikte her bir parametre seti yalnızca bir ikili diziyi temsil etmektedir. Yöntem, ikili arama alanına

eşdeğer 4 (dört) boyutlu sürekli bir alana dönüştürmeyi esas almaktadır (Pampara ve ark., 2005; Pampara ve ark., 2006; Pampará ve Engelbrecht, 2011)

Kuantum ilhamlı bitler: Bu teknik kuantum hesaplama bilim dalından ilham alınarak geliştirilmiştir. Bu tekniğe göre, her bir ajan bir çift sayıyla tanımlanmaktadır. Aday çözümler üretmek için dönme açısının sürekli olarak güncellendiği bir rotasyon matrisi kullanılmaktadır (Nezamabadi-pour, 2015).

Genetik operatörler: Arama uzayının ikili yapıda olmasından dolayı, ikili çaprazlama ve takas (swap) operatörü gibi genetik operatörler evrimsel algoritmalarda da kullanılabilmektedir. Öztürk ve ark. tarafından yapılan çalışmada, genetik operatörler özel arama stratejileri tasarlamak için kullanılmış ve Yapay Arı Kolonisi (Artificial Bee Colony-ABC) algoritmasının etkin bir ikili varyasyonu önerilmiştir (Ozturk ve ark., 2015).

İkili (binary) operatörler: Lojik alanında sıklıkla kullanılan, ikili değerler ile çalışan özel veya (xor), ve (and), veya (or) ve değil (not) operatörleri, ikili optimizasyon problemlerinin çözümünde de kullanılabilmektedir. İlgili çalışmalarda, lojik kapı-tabanlı operatörlerden bir veya birkaçı farklı kombinasyonlarda kullanılmış ve bu sayede yeni aday çözümler üretilebilmiştir (Marandi ve ark., 2006; Deng ve ark., 2011; Kiran ve Gunduz, 2013; Jia ve ark., 2014; Cinar ve Kiran, 2018).

Benzerlik tabanlı yaklaşımlar: Bu yaklaşımlarda, iki adet ikili değerler içeren bit dizilerinin (vektörün) benzerliğini (similarity) hesaplamak için farklı benzerlik indeksleri kullanılmakta ve hesaplanan bu benzerlik değerlerine göre yeni aday çözümler üretilmektedir. Literatürde genellikle Jaccard'ın benzerlik indeksi ve Hamming mesafesi teknikleri benzerlik ölçüsü olarak kullanılmaktadır (Cha ve ark., 2005; Choi ve ark., 2010; Kashan ve ark., 2012; Cinar ve Kiran, 2018).

Sezgisel metotlar: Literatürde, karar değişkenleri sürekli değerler alan problemleri çözmek için tasarlanmış yöntemleri, karar değişkenleri ikili değerler alan problemleri de çözebilmesi için özel olarak tasarlanmış farklı deneysel yöntemler de kullanılmaktadır (Chen ve ark., 2015).

Bu tez kapsamında, karar değişkenleri sürekli değerler alan problemlerin çözümü için yakın zamanda önerilmiş Yapay Alg Algoritmasının (AAA), literatürde var olan ikilileştirme yöntemlerine alternatif olarak yeni ve özgün yöntemler geliştirilerek, karar değişkenleri ikili değerler alan problemleri de çözebilmesinin sağlanması amaçlanmıştır.

2. KAYNAK ARAŞTIRMASI

Optimizasyon problemlerinin incelenmesi, bilinçli veya bilinçaltı davranışı olarak çözüm aranması bilimin kendisi kadar eskidir. Eski Yunan matematikçilerinin birçok optimizasyon problemi çözdükleri bilinmektedir. Örneğin, milattan önce 300 yıllarında yaşayan Euclid, bir karenin, dört kenarının toplam uzunluğu aynı olan tüm olası dikdörtgenler arasında en geniş alanı kapladığını kanıtlamıştır (Yang, 2010b).

1906'da Danimarkalı matematikçi J. Jensen, dışbükeylik kavramını ortaya koymuş ve şu anda Jensen'in eşitsizliği olarak adlandırılan ve dışbükey optimizasyonu (convex optimization) ve ekonomi gibi diğer alanlarda önemli bir rol oynayan bir eşitsizlik türetmiştir. Dışbükey optimizasyonu çok önemli bir matematiksel optimizasyon sınıfıdır çünkü bulunan herhangi bir optimal çözüm aynı zamanda global optimal olarak garanti edilmektedir. Dışbükey optimizasyon yöntemi kontrol sistemleri, veri uydurma ve modelleme, optimal tasarım, sinyal işleme, matematiksel finans olmak üzere birçok alanda uygulanmıştır (Yang, 2010b).

İkinci Dünya Savaşı sırasında Alman Enigma şifrelerini kıran Alan Turing sezgisel algoritmaları kullanan ilk kişi olmuştur. İngiliz matematikçi Gordon Welchman ile birlikte 1940 yılında kod çözme çalışmalarına kullanmak üzere Bombe adında kripto analitik bir elektromekanik makine tasarlamışlardır. Bombe makinesi, bir Enigma mesajında kodlanmış yaklaşık 10^{22} potansiyel kombinasyon içerisinden doğru olanı arayabilmek için sezgisel bir algoritma kullanmaktadır. Turing, beklendiği gibi çoğu zaman çalışan ancak doğru çözümü bulma garantisi olmayan arama metoduna sezgisel arama (heuristic search) adını vermiştir. (Yang, 2010a).

Bir sonraki önemli adım, 1960'larda ve 1970'lerde evrimsel algoritmaların geliştirilmesidir. John Holland, Michigan Üniversitesi'ndeki arkadaşları ile birlikte Genetik Algoritmaları (Genetic Algorithm-GA) geliştirdi, adaptif sistem üzerinde çalıştı ve bu sistemleri modellemek için çaprazlama ve rekombinasyon işlemlerini kullanan ilk kişi oldu. Genetik algoritmaların gelişimini özetleyen kitabını 1975 yılında yayınlandı. GA, Darwin'in evrimi ve biyolojik sistemlerin doğal seçimine dayanan ve onları matematiksel operatörler ile temsil eden bir araştırma yöntemidir. Önerildiği zamandan günümüze kadar çeşitli optimizasyon problemlerini çözmede çok başarılı olmuş, GA ile ilgili binlerce araştırma makalesi ve yüzlerce kitap yazılmıştır (Holland, 1975; Holland, 1992b; Yang, 2010b).

GA ile aynı dönemde, Berlin Teknik Üniversitesi'nden Ingo Rechenberg ve Hans-Paul Schwefel, 1963 yılında, havacılık mühendisliğindeki optimizasyon problemlerini çözmek için evrim stratejisi olarak adlandırılan bir arama tekniği geliştirdi. Bir sonraki büyük adım Kirkpatrick ve ark. (1983) tarafından metallerin tavlama işleminden esinlenerek Benzetimli Tavlama (Simulated Annealing-SA) optimizasyon tekniğinin geliştirilmesidir. SA, yüksek sıcaklıktaki ilk tahmin çözümüyle başlayan ve kademeli olarak sistemi soğutan bir arama algoritmasıdır. Yöntemde, aday çözüm daha iyi ise kabul edilmekte, aksi takdirde, sistemin herhangi bir yerel optimumdan çıkmasını sağlamak adına belirli bir olasılıkla da kabul edilmektedir. Daha sonra sistem yeterince yavaş bir şekilde soğutularak, küresel optimal çözüme ulaşılması beklenmektedir (Yang, 2014).

Dorigo (1992), optimizasyon ve doğal algoritmalar üzerine yazdığı doktora tezinde Karınca Kolonisi Optimizasyonu (Ant Colony Optimization-ACO) isimli yenilikçi çalışmasını sunmuştur. Bu arama tekniği, feromonu kimyasal bir iletişim aracı olarak kullanan sosyal karıncaların sürü zekâsından ilham alınarak geliştirilmiştir. Daha sonra, 1992 yılında, Stanford Üniversitesi'nden bilgisayar bilimcisi John R. Koza, bilgisayar programında çığır açan yepyeni bir makine öğrenme alanının temelini atarak Genetik Programlama (Genetic Programming) üzerine bir tez yayınlamıştır (John, 1992). Bu yaklaşım, bilgisayar programlarının üretilebilmesi için genetik operatörlerin kullanılması ve böylece belirli bir problem için en iyi programların kademeli olarak üretilebilmesi temeline dayanmaktadır (Yang, 2010b).

1995 yılında, sosyal psikoloji uzmanı James Kennedy ve mühendis Russell C. Eberhart tarafından PSO algoritmasının geliştirilmesiyle doğa-esinli yöntemler adına önemli bir ilerleme kaydedildi (Kennedy ve Eberhart, 1995) (Eberhart ve Kennedy, 1995). PSO, balık ve kuş sürülerinin zekasından ilham alan bir optimizasyon algoritmasıdır. Parçacık adı verilen ajanlar, başlangıçtaki rastgele çözümlerden başlayarak, arama alanında hareket ederler. Sürü, bireysel en iyi ve küresel en iyi çözümleri kullanarak daha kaliteli çözümleri bulmaya çalışır. İlk önerilmesinden bu yana, optimizasyon problemlerinin hemen hemen tüm alanlarına uygulanan yaklaşık 20 farklı PSO çeşidi geliştirilmiştir (Yang, 2010a).

Storn ve Price (1997), birçok uygulamada Genetik Algoritmadan daha verimli olduklarını deneysel olarak kanıtlayarak, Farksal Gelişim (Differential Evolution-DE) isimli vektör tabanlı evrimsel algoritmalarını geliştirdiler (Yang, 2014).

David H. Wolpert ve William G. Macready tarafından 1997 yılında yapılan çalışmada önerilen Ücretsiz Öğle Yemeği Yok Teoremi (No Free Lunch Theorem-NFL)

optimizasyon kavramına farklı bir bakış açısı getirdi (Wolpert ve Macready, 1997; 2005). Bu teoreme göre, eğer A algoritmasının performansı bazı optimizasyon fonksiyonları için B algoritmasından daha iyi ise, bunun tersine B algoritması da diğer fonksiyonlar için A algoritmasından daha iyi performans gösterebilmekteydi. Yani NFL teoremi, araştırmacılara evrensel olarak en iyi algoritmanın mevcut olmadığını bildirmekteydi. Ancak deneyimlerimizden elde ettiğimiz bilgilere göre, bazı algoritmaların verilen bazı optimizasyon problemleri için diğerlerinden daha iyi performans gösterdiklerini biliyoruz. Bu nedenle araştırmacılar bütün problemler için değil ama çoğu problem tipi için daha iyi algoritmalar tasarlamaya çalışmaktadırlar. (Yang, 2010b).

Modern meta-sezgisellerde bellek kullanımı ilk olarak Glover ve Laguna (1998) tarafından önerilen Tabu Arama (Tabu Search-TS) yönteminde gerçekleştirilmiştir. TS algoritmasında önceki çözümlerin tekrarlanmasının önüne geçmek için tabu belleği kullanılmaktadır. 2000'li yıllar, literatürde itibar gören optimizasyon algoritmaları için verimli yıllar olmuştur. Geem ve ark. (2001), Harmoni Arama (Harmony Search-HS) olarak isimlendirdikleri, müzisyenlerin doğaçlama davranışlarını taklit eden yeni bir sezgisel algoritma geliştirmişlerdir. Yine aynı yıl içerisinde, Abbass (2001) tarafından Bal Arıları Evliliği Optimizasyon Algoritması (Marriage in Honey Bees Optimization Algorithm-MBO) önerilmiştir. Birkaç yıl sonra, Kovaryans Matris Adaptasyonu ile Evrim Stratejisi (Evolution Strategy with Covariance Matrix Adaptation- CMA-ES) isimli yeni bir evrimsel tabanlı algoritma Hansen ve ark. (2003) tarafından literatüre kazandırılmıştır. 2004-2006 yılları arasında, araştırmacılar arıların davranışlarını modellemeye çalışmışlar ve bu yıllar optimizasyon yöntemlerinin gelişiminde bir anlamda arıların yılı olmuştur. Wedde ve ark. (2004), Arı Kovanı (BeeHive) isimli algoritmalarını, bal arılarının iletişimsel ve değerlendirmeci yöntemlerinden ilham alarak geliştirmişlerdir. Yine aynı yıl içerisinde Nakrani ve Tovey (2004), Bal Arısı (Honeybee) algoritmasını önermişler ve sunucu barındırma merkezlerinde dinamik sunucu tahsisinin optimizasyonunda kullanmışlardır. 2005 yılında, Teodorovic ve Dell'Orco (2005) Arı Kolonisi Optimizasyon (Bee Colony Optimization–BCO) algoritmasını, Drias ve ark. (2005) Arı Sürüsü Optimizasyon (Bees Swarm Optimization-BSO) algoritmasını, Yang (2005) mühendislik problemlerinin çözümünde kullandığı Sanal Arı Algoritmasını (Virtual Bee Algorithm-VBA) ve Karaboga (2005) Yapay Arı Kolonisi (Artificial Bee Colony-ABC) algoritmasını önermişlerdir. Bu algoritmalar içerisinde ABC algoritması araştırmacılar tarafından fazlasıyla itibar görmüş ve optimizasyon alanındaki çalışmalarda sıklıkla kullanılmıştır. Popülasyon tabanlı olan ve bal arılarının yiyecek

arama ve sürü davranışlarından ilham alınarak geliştirilen bir diğer yöntem olan Arılar Algoritması (Bees Algorithm), Pham ve ark. (2006) tarafından önerilmiştir. Xin-She Yang, Ateşböceği Algoritması (Firefly Algorithm-FA) (Yang, 2009) ve Guguk Kuşu Arama (Cuckoo Search-CS) algoritması (Yang ve Deb, 2009) olmak üzere iki yeni yöntemi aynı yıl içerisinde literatüre kazandırmıştır. Bu tarihten bir yıl sonra Yang (2010d), yarasaların ekolokasyon davranışlarından esinlenerek Yarasa Algoritmasını (Bat Algorithm-BA) geliştirmiştir. Yarasa Algoritmasından iki yıl sonra 2012 yılında, doğadaki çiçeklerin tozlaşma yönteminden esinlenen Xin-She Yang, Çiçek Tozlaşma Algoritmasını (Flower Pollination Algorithm-FPA) önermiştir (Yang, 2012).

Yeni optimizasyon tekniklerinin geliştirilmesi dünyada olduğu gibi ülkemizde de araştırmacılar tarafından ilgi duyulan bir konu olmuştur. 2005 yılında Derviş Karaboğa'nın geliştirdiği ABC algoritması (Karaboga, 2005) dünyada yaygın olarak kullanılan ilk algoritmalarımızdan birisi olmuştur. ABC algoritmasının literatüre kazandırılmasından günümüze kadar geçen zamanda araştırmacılarımız tarafından onlarca optimizasyon algoritması önerilmiştir. Doğadaki ağaç ve tohumları arasındaki ilişkiden ilham alınarak geliştirilen Ağaç-Tohum algoritması (Tree-Seed Algorithm-TSA) (Kiran, 2015b) ve su yosunlarının yaşam döngüsünden ilham alınarak geliştirilen Yapay Alg Algoritması (Artificial Algae Algorithm-AAA) (Uymaz, 2015; Uymaz ve ark., 2015a) yakın zamanda önerilen (state of the art) ve yaygın kullanımları açısından ümit vaat eden algoritmalar arasında yer almaktadır.

İkili optimizasyon, ayrık optimizasyonun bir alt alanıdır ve ikili optimizasyondaki karar değişkenleri sadece 0 (sıfır) veya 1 (bir) olabilmektedir. Bu değerlerin (0 yada 1) anlamı probleme özgüdür ama genel olarak 0 değerinin anlamı, niteliğin seçilmediği, 1 değerinin anlamı ise niteliğin seçildiğidir. Kapasitesiz Tesis Yerleşim Problemleri (Uncapacitated Facility Location Problems-UFLP) için bakacak olursak, 1 değeri tesisin açıldığını, 0 değeri ise tesisin kapalı olduğunu belirtir. Sırt Çantası Problemlerinde (Knapsack Problems-KP) ise 0 değeri o nesnenin çantaya koyulmadığını, 1 değeri ise nesnenin çantaya yerleştirildiğini belirtmek için kullanılır. Literatürde, Özellik Seçimi (Feature Selection - FS) (Tan ve ark., 2008; Taormina ve Chau, 2015; Shang ve ark., 2016; Shunmugapriya ve Kanmani, 2017), UFLP (Beasley, 1990a; Kashan ve ark., 2012; Kiran ve Gunduz, 2013; Kiran, 2015a; Atkinson ve Campos, 2016; Gunasundari ve ark., 2016), Birim Yüklenme Problemleri (Unit Commitment Problems-UC) (Pavez-Lazo ve Soto-Cartes, 2011; Trivedi ve ark., 2015; Kamboj, 2016; Khazaei ve ark., 2016; Li ve ark., 2016; Saravanan ve ark., 2016; Sun ve ark., 2017) ve Sırt Çantası Problemleri

(Knapsack Problems-KP) (Haddar ve ark., 2013; Haddar ve ark., 2015; Moosavian, 2015; Patvardhan ve ark., 2016; Pavithr ve Gursaran, 2016) gibi önemli ikili optimizasyon problemleri mevcuttur. Problemin boyutsallığına bağlı olarak muhtemel çözümlerin sayısı katlanarak artmaktadır ve muhtemel çözümlerin tümünün değerlendirilmesi zaman alıcı bir süreçtir. Bu tarz problemlerin olası çözümlerinin sayısı boyuta bağlı olarak üstel olarak arttığından dolayı *Np-Hard* problem kategorisindedirler (Krarup ve Pruzan, 1983; Monabbati ve Kakhki, 2015). Literatürde, bu tür problemleri çözmek için, enumeration (Yanasse ve Soma, 1987; James ve Nakagawa, 2005), branch and bound (Lalami ve El-Baz, 2012), azaltma şemaları (reduction schemes) (Freville ve Plateau, 1994), Beasley'in sezgisel yöntemi (Beasley's heuristic) (Beasley, 1990b) gibi bazı klasik teknikler uygulanmıştır. Fakat problemin boyutsallığı arttığında, bu yöntemler hesaplama maliyeti açısından verimli değildir ve ayrıca bu yöntemler genellikle problem bağımlıdır. Bu eksikliklerin üstesinden gelebilmek için, ikili optimizasyon problemlerinin çözümünde, literatürde yakın zamanda önerilmiş yöntemler arasında sürü zekası ve evrimsel hesaplama algoritmaları büyük yer işgal etmektedir. Bu bağlamda, son yıllarda, sürüler veya koloniler halinde yaşayan canlıların akıllı davranışlarını modelleyerek, optimizasyon problemlerini çözmek için birçok sürü zekası algoritması önerilmiştir.

Ayrık (discrete) problemlerin çözümü için Dorigo ve ark. (1996) tarafından önerilen ACO algoritması, gerçek hayattaki karıncaların yuva ve besin kaynağı arasındaki hareket ve davranışlarını modellemektedir. Karar değişkenleri sürekli değerler alan problemlerin çözümü için Kennedy ve Eberhart (1995) tarafından önerilen, PSO algoritması ise balık veya kuş sürülerinin davranışlarını taklit etmektedir. ABC algoritması, bal arılarının yiyecek arama davranışlarını ve bilgi paylaşımı için kullandıkları bir dans türünü (waggle dance) modellemektedir (Karaboga ve Basturk, 2007). Gri Kurt Optimizasyon (Grey Wolf Optimization-GWO) algoritması, gri kurtların liderlik hiyerarşisini ve avlanma mekanizmalarını taklit etmektedir (Mirjalili ve ark., 2014a). Mikro yarasaların ekolokasyon (ses dalgaları ile yer belirleme) yeteneğine dayanarak, sürekli optimizasyon problemini çözmek için Yarasa Algoritması (Bat Algorithm-BA) önerilmiştir (Yang, 2010d). Dünyanın yerçekimi yasasına ve kütle etkileşimlerine dayanan optimizasyon yöntemi Yerçekimsel Arama Algoritmasıdır (Gravitational Search Algorithm-GSA) (Rashedi ve ark., 2009). TSA algoritması ise doğadaki ağaç ve tohumları arasındaki ilişkiden ilham alınarak geliştirilmiştir (Kiran, 2015b). Mirjalili ve Lewis (2016) kambur balinalarının sosyal davranışlarını modelleyerek Balina Optimizasyon Algoritmasını (Whale Optimization Algorithm-

WOA) literatüre kazandırmışlardır. Ateşböceklerinin yanıp sönme davranışlarını modelleyen Yang (2010c), Ateşböceği Algoritmasını (Firefly Algorithm-FA) önermiştir. Başka bir çalışmada, optimizasyon problemlerinin çözümü için bakterilerin yiyecek arama davranışları modellenmiştir (Passino, 2002). Su yosunlarının yaşam döngüsünden ilham alınarak geliştirilen optimizasyon algoritması (Uymaz ve ark., 2015a) Yapay Alg Algoritmasıdır (Artificial Algae Algorithm-AAA). Bu algoritmaların ortak noktası, popülasyon kullanmaları ve bu popülasyon arasındaki etkileşimleri değerlendirmeleridir. ACO algoritması haricindeki diğer yöntemlerin bir başka ortak noktası ise sürekli optimizasyon problemlerini çözmek için önerilmiş olmalarıdır. İkili optimizasyon problemlerini çözebilmeleri için ise ayrık olarak yapılandırılmış çözüm uzayında çalışacak şekilde dönüştürülmeleri gerekmektedir. Literatürde, bu algoritmaların ikilileştirilmesi için önerilmiş farklı yaklaşımlar mevcut olup önceki bölümde anlatılmıştır. İkilileştirme çalışmaları için yapılan kaynak araştırması ise algoritma bazında gruplandırılmış ve aşağıdaki gibi özetlenmiştir.

Parçacık Sürü Optimizasyonu (PSO) Algoritması kullanılarak yapılan çalışmalar: Kennedy ve Eberhart (1997), PSO algoritmasını ikili değerler alan karar değişkenleri ile çalışacak şekilde yeniden yapılandırarak İkili PSO Algoritmasını (Binary PSO-BPSO) önermişlerdir. Bu çalışmada, sürekli değerler alan karar değişkenlerini ikili değerlere dönüştürmek için sigmoid limit dönüşüm (sigmoid limiting transformation) fonksiyonu kullanılmıştır. Sevkli ve Guner (2006) araştırma makalesinde, sürekli bir Parçacık Sürüsü Optimizasyonu (PSO) algoritmasının ikili uzaya dönüştürülmesinde mod_2 ve kırpma (floor) lojistik fonksiyonlarını uygulamış ve UFLP setini çözmek için kullanmışlardır. Çözüm kalitesini artırmak için, yerel arama (local search) mekanizması PSO algoritmasına entegre edilmiştir. Nezamabadi-pour ve ark. (2008) tarafından yapılan çalışmada Yeni İkili PSO (Novel Binary PSO- NBPSO) Algoritması önerilmiştir. NBPSO algoritmasında, sürekli değerli değişkenleri ikili değerlere dönüştürmek için hız tabanlı (velocity-based) bir sigmoid transfer fonksiyonu kullanılmıştır. NBPSO algoritmasında bazı uyarlamalar yapıldıktan sonra, Garantili Yakınsamalı BPSO (Guaranteed Convergence BPSO-GCBPSO) ve İyileştirilmiş NBPSO (Improved NBPSO- INBPSO) önerilmiştir. GCBPSO yönteminde farklı bir hız güncelleme denklemi önerilmiştir. INBPSO yönteminde ise algoritmanın durağanlığı (stagnation) kontrol edilmek istenmiş ve yönteme eklenen durağanlık kontrol parametresi ile sigmoid transfer fonksiyonu değiştirilmiştir. Guner ve Sevkli (2008) tarafından yapılan çalışmada, ayrık optimizasyonda yaygın olarak kullanılan UFLP setini çözmek için Ayrık Parçacık

Sürüsü Optimizasyonu (Discrete Particle Swarm Optimization-DPSO) yöntemi önerilmiştir. Sonuçları iyileştirmek için, yönteme yerel bir arama mekanizması entegre edilmiştir. Saha ve ark. (2011), ikili optimizasyon problemlerinin çözümü için önerilmiş olan Modulo-Bazlı İkili Parçacık Sürü Optimizasyonu (CPSO) algoritmasında uyarlamalar yaparak UFLP setini çözmüşlerdir. Beheshti ve ark. (2015) çalışmalarında, PSO algoritmasının memetik ikili varyasyonlarını önermişlerdir. Bansal ve Deep (2012) çalışmalarında, özellikle 0-1 Sırt Çantası Problemi ve Çok Boyutlu Sırt Çantası Probleminin (Multidimensional Knapsack Problem-MKP) çözümü için yeni bir Modifiye Edilmiş İkili Parçacık Sürü Optimizasyon (Modified Binary Particle Swarm Optimization-MBPSO) Algoritmasını önermişlerdir. BPSO algoritması ile karşılaştırıldığında, bu iyileştirilmiş algoritma, KP problemlerinin çözümünde sürüdeki çeşitliliği muhafaza etmek ve daha keşfedici hale getirmek için yeni bir olasılık fonksiyonunu kullanmaktadır. Ayrıca parçacığın hızının normalleştirilmesi için sigmoid fonksiyonu kullanılmıştır. Diğer bir çalışmada Yuan ve ark. (2009), BPSO'ya lamda yineleme yöntemini (lambda-iteration method) entegre ederek, Birim Yüklenme (Unit Commitment-UC) probleminin çözümü için İyileştirilmiş İkili PSO (Improved Binary PSO-IBPSO) yöntemini önermişlerdir. Deneysel sonuçlar, IBPSO'nun daha düşük üretim maliyeti ve daha kısa hesaplama süresi açısından literatürdeki bilinen diğer yöntemlerden üstün olduğunu göstermektedir.

ABC (Artificial Bee Colony) Algoritması kullanılarak yapılan çalışmalar: Kashan ve ark. (2012), ikili optimizasyon problemlerinin çözümü için Yapay Arı Kolonisi algoritmasını temel alan, Farklılık Tabanlı ABC Algoritması (Dissimilarity Based ABC Algorithm-DisABC) olarak isimlendirdikleri yeni bir yöntem tasarlamışlardır. DisABC, yeni bireyler üretmek için ikili vektörler arasındaki benzeşmezlik ölçüsünü (measure of dissimilarity) kullanmaktadır. Kiran ve Gunduz (2013), çözüm güncelleme denklemini *xor* lojik operatörü ile değiştirerek Lojik Kapı Tabanlı İkili ABC Algoritması (A Logic Gate-Based Binary ABC Algorithm-binABC) ile ikili optimizasyon problemlerini çözmeyi amaçlamışlardır. Önerilen binABC, UFLP setini çözmek için uygulanmış ve elde edilen sonuçlar, BPSO, DisABC ve IBPSO algoritmaları ile karşılaştırılmıştır. Önerilen yaklaşım, ikili optimizasyon problemlerinin çözümünde, çözüm kalitesi, sağlamlık ve sadelik açısından karşılaştırılan diğer algoritmalara alternatif bir yaklaşım olmuştur. Kiran (2015a), ABCbin (İkili ABC Algoritması - Binary ABC Algorithm) olarak adlandırdığı yönteminde, ABC yöntemi ile modülo tabanlı lojistik fonksiyonu kullanarak ikili optimizasyon problemlerini

çözmüştür. Bu çalışmada, yapay ajanlar sürekli olarak yapılandırılmış çözüm uzayında çalışmakta, probleme özgü uygunluk fonksiyonu hesaplanırken sürekli değerlere sahip karar değişkenleri modülo (mod_2) ve kırpma (floor) lojistik fonksiyonları ile ikili değerlere dönüştürülmektedir. ABCbin yönteminin performans değerlendirmesi, 15 adet test kümesine sahip UFLP üzerinde gerçekleştirilmiş ve CPSO, BPSO, IBPSO, binABC ve DisABC algoritmaları ile kıyaslanmıştır. Ozturk ve ark. (2015) aday çözümler oluşturmak için, genetik ve komşuluk operatörlerini kullanan yapay arı kolonisi algoritmasının yeni bir ikili versiyonunu (GB-ABC) geliştirmişlerdir. Temel ABC algoritmasının komşu arama mekanizmasına entegre edilmiş olan modifikasyon dört aşamadan oluşmaktadır.

- Güncel bir yiyecek kaynağının komşuluklarından, popülasyondan rasgele iki yiyecek kaynağı seçilir.
- Çocuk besin kaynakları üretmek için mevcut, iki komşu, küresel en iyi ve ikili sıfır değerleri ile oluşturulan çözümler arasında iki noktadan çaprazlama (two-point crossover) operatörü uygulanır.
- Torun yiyecek kaynakları üretmek için, çocuk yiyecek kaynaklarına takas operatörü uygulanır.
- Çocuk ve torun besin kaynakları arasında, en iyi besin kaynağını seçilir.

Temel ABC algoritmasının global ve yerel arama yetenekleri, GB-ABC çalışmasında ikili alana genişletilmiştir. Pampará ve Engelbrecht (2011) çalışmalarında, karar değişkenleri ikili değerler alan optimizasyon problemlerine uygulanması amacı ile ABC 'nin üç versiyonunu önermişleridir. Telekomünikasyon endüstrisinde sıklıkla kullanılan sinyal işleme tekniği olan Açı modülasyonu (AM) kullanılarak ikilileştirilen ABC versiyonu, karşılaştırılan diğer algoritmalardan daha kaliteli sonuçlar üretmiştir. Daha sonra önerilen Açı Modülasyonlu ABC (Angle-Modulated ABC–AMABC) yöntemi, Açı Modülasyonlu PSO (Angle-Modulated Particle Swarm Optimization-AMPSO) ve Açı Modülasyonlu DE (Angle-Modulated Differential Evolution-AMDE) algoritmaları ile karşılaştırılmıştır. Jia ve ark. (2014) tarafından yapılan çalışmada, BitABC olarak isimlendirdikleri, işçi arıların ve kâşif arıların hareketleri için bitsel (bitwise) *xor*, *ve* (and) ve *veya* (or) operatörleri kullanan etkili bir ikili ABC algoritması önerilmiştir. Hancer ve ark. (2015) tarafından yapılan çalışmada, evrimsel temelli benzerlik arama mekanizmalarının mevcut bir ikili ABC varyantına entegre edilmesiyle Özellik Seçimi (Feature Selection-FS) problemlerinin çözümü için ikili ABC algoritması

önerilmiştir. Özellik seçimi, özellik kümesindeki özellikler arasındaki karmaşık etkileşimleri araştırarak, alakasız veya gereksiz özellikleri ortadan kaldırmak amacı ile gerçekleştirilen temel bir veri önişleme adımıdır. Diğer bir çalışmada Singhal ve ark. (2015), Birim Yüklenme (Unit Commitment Problem-UC) problemi için, Modifiye Edilmiş İkili ABC (Modified Binary ABC-MBABC) Algoritmasını ve Dinamik Ekonomik Sevkiyat (Dynamic Economic Dispatch-DED) yöntemine dayanan yeni bir yaklaşımı önermişlerdir. MBABC algoritması birimleri devreye alma ve devre dışı bırakmak için kullanılırken, optimum dağıtım çözümü DED yöntemi kullanılarak belirlenmektedir. Önerilen MBABC algoritması, UC probleminde yeni ikili çözümler üretmek için, ikili dizeler arasındaki farkın ölçüsüne dayanan yeni bir mekanizma kullanmaktadır. Buna ek olarak, MBABC algoritmasında, terk edilmiş çözümü global en iyi çözüm ile değiştiren akıllı bir kaşif arı fazı önerilmiştir. MBABC tarafından elde edilen çözüm kalitesi, dağıtık arama alanını sağlayan genetik çaprazlama ile melezleştirerek artırılmıştır. Elde edilen sonuçların literatürdeki diğer yöntemlerle karşılaştırılması ile Modifiye Edilmiş İkili ABC-Genetik Çaprazlama (Modified Binary ABC-Genetic Crossover - MBABC-GC) Algoritmasının sağlamlık (robustness) açısından üstünlüğü doğrulanmıştır.

Tabu Arama (Tabu Search-TS) Algoritması kullanılarak yapılan çalışmalar: Al-Sultan ve Al-Fawzan (1999) tarafından yapılan çalışmada, saf ikili problemler içeren UFLP'yi çözmek için TS tabanlı yeni bir yöntem önerilmiştir. Michel ve Van Hentenryck (2004), Depo Konum Problemini (Warehouse Location Problem) çözmek için TS'yi yeniden uyarlamışlar ve önerilen yöntemin daha gürbüz çözümler ürettiğini çalışmalarında belirtmişlerdir. Başka bir çalışmada ise Sun (2006), UFLP'yi çözmek için yeni bir TS sezgisel yöntem önermiştir.

Yerçekimsel Arama Algoritması (Gravitational Search Algorithm-GSA) kullanılarak yapılan çalışmalar: Rashedi ve ark. (2010), GSA'nın ikili bir versiyonu olan BGSA yöntemini önermişlerdir. Bu çalışmada, lojistik fonksiyon olarak tanjant hiperbolik fonksiyonu kullanılarak nümerik test fonksiyonları çözülmüştür. Mirjalili ve ark. (2014b) ise ikili optimizasyon problemlerinin çözümü için GSA ile PSO yöntemlerini birlikte kullanmış, ikilileştirme için transfer fonksiyonu kullanarak BPSOGSA isminde melez bir algoritma önermişlerdir. Nezamabadi-pour (2015) çalışmasında, ikili optimizasyon problemlerini çözmek için GSA'da kuantum hesaplamayı kullanarak İkili Kuantum-Esinli GSA (Binary Quantum-Inspired GSA-BQIGSA) yöntemini önermiştir.

Bu çalışmada, GSA yönteminin pozisyon ve hız güncelleme kuralları, kuantum biti, süper pozisyon ve modifiye edilmiş rotasyon Q-gates stratejisi ile değiştirilmiştir.

Harmoni Arama (Harmony Search-HS) Algoritması kullanılarak yapılan çalışmalar: Wang ve ark. (2010b), HS algoritmasına yeni bir akort ayarlama kuralı ekleyerek Ayrık İkili HS'yi (Discrete Binary HS-DBHS) önermişlerdir. Salman ve ark. (2015) ise ikili optimizasyon problemleri için uyarlanabilir olasılıksal HS (Adaptive Probabilistic Harmony Search-APHS) algoritmasını sunmuşlardır. Kong ve ark. (2015b) çalışmalarında, 0-1 sırt çantası problemlerinin çözümü için, HS yöntemindeki doğaçlama sürecinde (improvisation process) yapılan değişiklik ile Basitleştirilmiş İkili Harmoni Arama (Simplified Binary Harmony Search-SBHS) algoritmasını önermişlerdir. SBHS ve diğer HS yöntemleri arasındaki temel fark doğaçlama (improvisation) sürecindedir. Akort Ayarlama Oranı (Pitch Adjustment Rate-PAR) ve adım bant genişliği (step bandwidth-bw) yerine armoni belleğinde saklanan armoniler arasındaki farklar, yeni çözümler üretmek için kullanılmıştır. Ayrıca, algoritmanın yakınsama kabiliyetini artırmak için Harmonik Bellek Dikkate Alma Oranı (Harmony Memory Considering Rate-HMCR) değeri boyut sayısına bağlı olarak dinamik bir şekilde ayarlanmıştır. Popülasyon çeşitliliğini (population diversity) daha da artırmak için, uygulanabilir olmayan çözümler etrafında, sezgisel tabanlı yerel arama (local search) yapılmış ve daha iyi kalitede çözümler elde edilmesi amaçlanmıştır. Diğer bir çalışmada, HS'nin yeni bir ikili versiyonu olarak önerilen NBHS yöntemi, büyük ölçekli Çok Boyutlu Sırt Çantası Probleminin çözümünde kullanılmıştır (Kong ve ark., 2015a).

Farksal Gelişim (Differential Evolution-DE) Algoritması kullanılarak yapılan çalışmalar: Pampara ve ark. (2006) yaptığı çalışmada, DE (Storn ve Price, 1997) yönteminin ikili değerler ile çalışabilmesi için açı modülasyonu isminde, optimizasyon alanında yeni bir haritalama yöntemi kullanılmıştır. Telekomünikasyon alanında kullanılan açı modülasyonu tekniğinde, trigonometrik fonksiyonlar kullanılarak ikili bit dizesi oluşturulabilmektedir. Açı modülasyonunda problemin karmaşıklığı 4 boyutlu sürekli değerler olmakta ve bu nedenle problemin karmaşıklığı indirgenmiş olmaktadır. Elde edilen deneysel sonuçlar neticesinde tekniğin etkinliği ve DE'nin ikili alanda çalışabilme yeteneği kanıtlanmıştır. Engelbrecht ve Pampara (2007) çalışmalarında, Açı Modülasyonlu DE (Angle Modulated Differential Evolution-AMDE), İkili DE (Binary DE-binDE) ve Normalizasyon DE (Normalization DE-normDE) isminde üç yaklaşım önermişlerdir. BinDE algoritmasındaki bir çözümün her bir gerçek değerli bileşeni, bitsel olarak 0 veya 1 üretme olasılığıdır. İkili değeri belirlemek için kayan noktalı çözüm

vektörünü, olasılıkların bir vektörü olarak yorumlamaktadır. NormDE algoritmasında ilk olarak her bir çözüm vektörüne uygulanan normalizasyon işlemi ile tüm bileşenler doğrusal olarak [0, 1] aralığında ölçeklenir. Normalleştirilmiş değer 0.5'ten küçükse, buna karşılık gelen ikili değer 0 olarak, aksi halde 1 olarak kullanılır. Yang (2008) araştırma makalesinde, aritmetik operatörler yerine farklı operatörler kullanan, temel DE'nin mutasyon denklemi gibi bir denklem sunarak ikili problemleri çözmeyi amaçlamıştır. Wang ve ark. (2010a), ikili diferansiyel evrim algoritmasının iyileştirilmiş bir versiyonu olan MBDE algoritmasını önermişlerdir. Bir diğer çalışmada, ikili aday çözümlerin üretilmesinde lojik operatörler (*xor*, *ve*, *veya* ve *değil* operatörleri) kullanılmıştır (Deng ve ark., 2011). Bu çalışmada, ikili optimizasyon problemlerini çözmek için İkili Kodlanmış DE (Binary Encoding DE-BDE) yöntemi önerilmiştir. Kashan ve ark. (2013) yaptığı çalışmada, ikili vektörler arasında benzeşmezlik (dissimilarity) ölçüsünü kullanarak DE algoritmasını ikili optimizasyon problemleri için yeniden tasarlamışlardır. Chen ve ark. (2015) makalesinde, İkili Öğrenme DE (Binary Learning DE-BLDE) algoritması, PSO algoritmasındaki öğrenme mekanizmasından ilham alınarak geliştirilmiştir. Bu yaklaşım ile, son popülasyondan öğrenilen bilgi ile küresel optimal çözümler verimli bir şekilde bulunabilmektedir. Banitalebi ve ark. (2016) tarafından önerilen Kendinden Uyarlamalı İkili Farksal Gelişim (Self-Adaptive Binary Differential Evolution-SabDE) algoritmasında benzeşmezlik (dissimilarity) ölçüsü kullanılmıştır. SabDE yönteminde ayrıca, aday çözümlerinin nasıl üretildiğini seçmek için adaptif bir mekanizma ve parametre değerlerinin ayarlanması için ise kaotik bir süreç kullanılmıştır. Önerilen algoritma, yüksek boyutlu sırt çantası problemlerinde ve Evrimsel Hesaplama Kongresinde (Congress on Evolutionary Computation - CEC 2015) sunulmuş 15 adet öğrenme temelli problemler üzerinde çalıştırılmış ve güncel algoritmalar ile kıyaslanmıştır. Deneysel sonuçlar, önerilen algoritmanın rekabetçi bir performans sergilediğini ve bazı durumlarda mevcut algoritmalardan daha üstün olduğunu ortaya koymaktadır.

Yapay Alg Algoritması (Artificial Algae Algorithm-AAA) kullanılarak yapılan çalışmalar: Zhang ve ark. (2016) tarafından yapılan çalışmada, farklı eğri yapısına sahip iki adet lojistik transfer fonksiyonu kullanılarak AAA ikilileştirilmiş, yönteme onarım operatörleri (repair operators) ve seçkin yerel arama (elite local search) eklenmiştir. Önerilen yöntemin verimliliğini göstermek için, 94 probleme sahip Çok Boyutlu Sırt Çantası Problemi (Multidimensional Knapsack Problem-MKP) seti kullanılarak kıyaslamalar gerçekleştirilmiştir. Elde edilen sonuçlar ile İkili AAA'nın

(Binary AAA-BAAA) karşılaştırılan diğer algoritmalara üstünlüğü gösterilmiştir. Beşkirli ve ark. (2018) çalışmalarında, rüzgar türbinlerini bir rüzgar çiftliğine konumlandırıp ikili optimizasyon problemi olarak çözülmesini amaçlamışlardır. Türbin yerleştirme işlemi 2x2km alan için tasarlanmış, alanın yüzeyi ikili kodlama kullanılarak 10x10 ızgaraya ve 20x20 ızgaraya bölünerek hesaplanmıştır. AAA, on adet transfer fonksiyonunu kullanılarak ikilileştirilmiş, türbin yerleştirme problemine uygulanmış ve en iyi sonucu elde eden algoritma İkili Yapay Alg Algoritması olarak isimlendirilmiştir.

Diğer sürü zekâsı ve evrimsel hesaplama yöntemleri kullanılarak yapılan çalışmalar: Kratica ve ark. (2001) tarafından ve Jaramillo ve ark. (2002) tarafından yapılan çalışmalarda, lokasyon problemlerinin çözümünde genetik algoritmalar kullanılmıştır. Bir diğer çalışmada Komşuluk Arama (Neighborhood Search) yöntemleri, ikili bir problem olan UFLP'yi çözmek için kullanılmıştır (Ghosh, 2003). MKP'yi çözmek için ise İkili Karınca Sistemi (Binary Ant System-BAS) olarak adlandırılan yeni bir ACO temelli yaklaşım geliştirilmiştir (Kong ve ark., 2008). BAS yönteminde, ikili çözüm yapısı için özel olarak tasarlanmış bir feromon yerleştirme yöntemi kullanılmış ve çözüm üretme prosedüründe mümkün olmayan çözümlerin (infeasible solutions) üretilmesine izin verilmiştir. Diğer bir çalışmada Gortázar ve ark. (2010), Dağılım Arama Metodunu (Scatter Search Methodology) ikili problemleri çözmek için kullanmışlardır. Azad ve ark. (2012) ise Çok Boyutlu 0–1 Sırt Çantası Problemini (Multidimensional 0–1 Knapsack Problem) çözmek için Yapay Balık Sürüsü Algoritmasının (Artificial Fish Swarm Algorithm-AFSA) bir ikili versiyonunu sunmuşlardır. Bu çalışmada, uygun olmayan çözümler bir kod çözme algoritması kullanılarak uygun çözümlere dönüştürülmüştür. Azad ve ark. (2013) çalışmalarında, UFLP'nin çözümü için AFSA'nın basitleştirilmiş bir ikili versiyonu olan S-bAFSA (Simplified Binary AFSA) algoritmasını sunmuşlardır. Bu çalışmada yeni bireyler çaprazlama ve mutasyon kullanılarak oluşturulmuştur. Çözümlerin kalitesini iyileştirmek için popülasyon periyodik olarak yeniden ilklendirilmektedir. Elde edilen çözümün doğruluğunu artırmak için, belirli bir sayıda yerel arama (local search) uygulanmıştır. Diğer bir çalışmada Yarasa Algoritması (Bat Algorithm-BA) ikili problemleri çözmek için uyarlanmıştır (Babaoğlu, 2016). Laalaoui ve M'Hallah (2016) çalışmalarında, sezgisel bir değişken komşuluk arama algoritmasını önermiş ve büyük boyutlu test örneklerini çözmek için kullanmışlardır. Larrañaga ve Lozano (2001) tarafından önerilen ve temelde Genetik Algoritmaya benzeyen Dağılım Tahmin Algoritması (Estimation of Distribution Algorithm-EDA)

bireyleri oluşturmak için olasılık modeli kullanmaktadır. Bu nedenle, Olasılıksal Model Oluşumlu Genetik Algoritma (Probabilistic Model-Building Genetic Algorithm) olarak da isimlendirilmektedir. EDA, ikili optimizasyon problemlerinde yaygın olarak kullanılmaktadır. Literatürde, ayrık ve sürekli optimizasyon problemleri için farklı versiyonları bulunmaktadır. Tek Değişkenli Marjinal Dağılım Algoritması (Univariate Marginal Distribution Algorithm-UMDA) ise EDA algoritmasının farklı bir türü olarak literatürde yer almaktadır (Mühlenbein, 1997). Olasılıksal vektör kullanımı ve yeni bireylerin oluşturulmasında örnekleme kullanımından dolayı, UMDA yöntemi çözüm uzayını EDA yönteminden daha iyi taramaktadır (Hashemi ve Meybodi, 2011). Tohyama ve ark. (2011) tarafından yapılan çalışmada UFLP'nin çözümü için, iyi bir çözüm ihtimali bulunan çözüm alanını arayan işlem ile mutasyon operatörü birlikte kullanılarak bütün çözüm uzayını verimli bir şekilde arayabilen genetik algoritma önerilmiştir.

3. MATERYAL VE YÖNTEM

3.1. Yapay Alg Algoritması (AAA)

Son yıllarda sürekli ve ayrık karakteristiklere sahip problemlerin çözümü için sürü zekâsına dayalı birçok algoritma önerilmiştir. Literatürde İngilizce kısaltılmış ismi AAA (Artificial Algae Algorithm) olarak anılan Yapay Alg Algoritması da sürü zekası temel alınarak, gerçek hayattaki alglerden esinlenilerek üretilmiş ve sürekli optimizasyon problemlerinin çözümü için geliştirilmiştir (Uymaz, 2015; Uymaz ve ark., 2015b; 2015a).

Algler çok çeşitli türleri olan, ayrı çekirdek zarları ve klorofilleri bulunan fotosentetik ökaryot canlılardır. Çok hücreli algler (makroalgler) makro bitkiler görünümündedir ve deniz yosunu olarak da bilinirler. Tek hücreli algler ise fitoplankton veya mikroalgler olarak isimlendirilmişlerdir. AAA optimizasyon algoritması, doğada var olan mikroalglerin davranışlarından ilham alınarak geliştirilmiş ve literatüre kazandırılmış bir yöntemdir (Uymaz, 2015).

Algler, tatlı sularda, denizlerde ve hatta buzullar gibi çok farklı ortamlarda yaşayabilen canlılar olup, ayrıca tuzlu su ve kaplıcalar gibi zorlu yaşam şartlarına da uyum sağlayabilmişlerdir (Graham ve Wilcox, 2000). Yapay algler, gerçek dünyadaki algler gibi, fotosentez yapabilmek için ışık kaynağına doğru helisel biçimde hareket edebilir, ortama uyum sağlayarak baskın türü değiştirebilir ve mitoz bölünme ile çoğalabilirler. Alglerin yiyecek arama davranışları, bir besin kaynağına doğru birlikte hareket etmeyi temel almaktadır. Alglerin bu yaşam davranışlarından esinlenilerek geliştirilen AAA algoritması "Evrimsel Süreç", "Adaptasyon Süreci" ve "Helisel Hareket" olarak adlandırılan üç temel bölümden oluşmaktadır. AAA algoritmasında her bir alg kolonisi optimizasyon problemi için olası bir çözüme karşılık gelmekte ve optimizasyon problemindeki her karar değişkeni alg hücresi olarak temsil edilmektedir. Ayrıca algoritmada, kesme kuvveti(shear force), enerji kaybı(energy loss) ve adaptasyon oranı(adaptation rate) olmak üzere üç adet yönteme özgü kontrol parametresi mevcuttur. Parametrelerin ayarlanmasından sonra, algoritmanın ilklendirilme aşamasında *N* adet alg kolonisi Denklem 3.1 kullanılarak üretilmektedir (Uymaz, 2015).

$$x_{i,j}^{0} = L_j + r_{i,j} \times \left(H_j - L_j\right) \ , \ i = 1,2,\dots,N \ \ ve \ \ j = 1,2,\dots,D \qquad (3.1)$$

Denklemde, x_i i'nci alg kolonisini, $x_{i,j}^0$ i'nci alg kolonisinin *t=0* anındaki j'inci alg hücresini, L_j j'inci boyut için arama uzayının alt limitini, H_j j'inci boyut için arama uzayının üst limitini, $r_{i,j}$ i'nci alg kolonisinin j'inci hücresi için [0,1] aralığında üretilen rastgele sayıyı, *N* alg kolonileri sayısını (popülasyon) ve *D* ise problemin boyutunu ifade etmektedir.

Alg kolonilerinin sürtünme yüzey alanlarının hesaplamaları, enerji değerleri hesaplamaları ve evrimsel süreç fazındaki işlemlerde kullanılan alg kolonilerinin büyüklük değerleri başlangıçta tüm alg kolonileri için 1 (bir) olarak belirlenir. i'nci alg kolonisini *t+1* anındaki büyüklük değeri aşağıdaki Denklem 3.2 kullanılarak hesaplanmaktadır.

$$G_i^{t+1} = G_i^t + \mu_i^t \, G_i^t \quad , \; i = 1,2, \dots, N \tag{3.2}$$

Denklemde, G_i^t i'inci alg kolonisinin *t* anındaki büyüklük değerini, G_i^{t+1} i'nci alg kolonisinin *t+1* anındaki büyüklük değerini, μ_i^t i'nci alg kolonisinin *t* anındaki özgül büyüme hızını ve *N* ise alg kolonileri sayısını (popülasyon) ifade etmektedir. Denklem 3.2'de kullanılan özgül büyüme hızı μ_i^t, Denklem 3.3 kullanılarak hesaplanmaktadır.

$$\mu_i^t = \frac{f_i^t}{\left(\frac{G_i^t}{2}\right) + f_i^t} \tag{3.3}$$

Denklemde, f_i^t i'nci alg kolonisinin *t* anındaki uygunluk değerini ifade etmektedir. Her bir alg kolonisi için elde edilen optimizasyon problemine özgü amaç fonksiyon değerleri, Denklem 3.4 yardımıyla normalize edilerek her bir alg kolonisinin uygunluk değerleri hesaplanmaktadır.

$$f_i^t = \frac{Obj_i^t - worst\,(\overrightarrow{Obj})}{best\,(\overrightarrow{Obj}) - worst\,(\overrightarrow{Obj})} \tag{3.4}$$

Denklemde, Obj_i^t *t* anında i'nci alg kolonisi için hesaplanmış amaç fonksiyon değerini, $best(\overrightarrow{Obj})$ alg kolonileri arasındaki en iyi amaç fonksiyon değerlerini ve $worst(\overrightarrow{Obj})$ ise alg kolonileri arasındaki en kötü amaç fonksiyon değerlerini ifade

etmektedir. Eğer optimizasyon problemi bir minimizasyon problemi ise, alg kolonileri içerisindeki en iyi çözüm, amaç fonksiyon değeri en düşük olan alg kolonisi, en kötü çözüm ise amaç fonksiyon değerleri en yüksek olan alg kolonisidir.

Alg kolonilerinin hareketleri büyüklüklerine göre farklılık göstermektedir. Büyüyen alg kolonisinin sürtünme yüzeyi de büyüdüğünden dolayı helisel hareket frekansı artar ve hareketleri yavaşlar. Bu davranış biçimi, yöntemin yerel arama (sömürü) (exploitation) yeteneğini artırmaktadır. Bunun aksine, küçük alg kolonisinin ise sürtünme yüzeyi daha az olduğundan dolayı hareket adımları daha büyük olmaktadır. Bu davranış biçimi ise AAA yöntemine global arama (keşif) (exploration) yeteneği kazandırmaktadır. i'nci alg kolonisinin *t* anındaki sürtünme yüzeyi τ_i^t, Denklem 3.5 kullanılarak hesaplanmaktadır (Uymaz, 2015).

$$\tau_i^t = 2\pi \left(\sqrt[3]{\frac{3G_i^t}{4\pi}} \right)^2 \tag{3.5}$$

Helisel hareket fazında, asıl amaç çözüm alanını aramaktır. Bu amaçla, her alg kolonisi için aşağıda belirtilen üç farklı denklem (Denklem 3.6, Denklem 3.7 ve Denklem 3.8) kullanılarak her alg kolonisi kendi konumundan başka bir konuma taşınır ve aday çözüm üretilir. Helisel olarak hareket edecek olan alg kolonisi ilk olarak aday çözüme kopyalanır ve aşağıdaki üç denklem kullanılarak rastgele belirlenmiş üç farklı boyutta hareket sağlanmış olur (Uymaz, 2015).

$$x_{i,m}^{t+1} = x_{i,m}^t + \left(x_{j,m}^t - x_{i,m}^t\right)(\Delta - \tau_i^t)\,\rho \tag{3.6}$$

$$x_{i,k}^{t+1} = x_{i,k}^t + \left(x_{j,k}^t - x_{i,k}^t\right)(\Delta - \tau_i^t)\cos\alpha \tag{3.7}$$

$$x_{i,l}^{t+1} = x_{i,l}^t + \left(x_{j,l}^t - x_{i,l}^t\right)(\Delta - \tau_i^t)\sin\beta \tag{3.8}$$

Denklemlerde, $x_{i,m}^t$, $x_{i,k}^t$ ve $x_{i,l}^t$ i'nci alg kolonisi x_i'nin *t* anında rastgele seçilen alg hücrelerini (boyut), $x_{j,m}^t$, $x_{j,k}^t$ *ve* $x_{j,l}^t$ i'nci alg kolonisi için *t* anında rastgele seçilen alg kolonisi (ışık kaynağı) x_j^t'nin seçilen boyutlardaki alg hücrelerini, x_i^{t+1} aday çözümü, τ_i^t, Denklem 3.5 kullanılarak hesaplanan i'nci alg kolonisinin *t* anındaki sürtünme yüzeyini, ρ [-1,1] aralığında üretilen rastgele sayıyı, α ve β $[0, 2\pi]$ aralığında üretilen rastgele

açıyı ve Δ ise algoritmaya özgü kontrol parametresi olan kesme kuvvetini (shear force) ifade etmektedir (Uymaz, 2015).

Helisel hareket fazında kullanılan Denklem 3.6 doğrusal hareketi, Denklem 3.7 ve Denklem 3.8 ise açısal hareketleri sağlamaktadır. Tek boyutlu problemler için Denklem 3.6, iki boyutlu problemler için Denklem 3.7 ve Denklem 3.8 kullanılmaktadır. Problem boyutunun üç veya daha fazla olduğu durumlarda ise üç denklem birlikte kullanılarak helisel hareket gerçekleştirilmiş olur (Uymaz, 2015).

Popülasyondaki her bir alg kolonisi belirli bir enerji değerine sahiptir. Her iterasyon başlangıcında, alg kolonilerinin büyüklük değerlerinin [0,1] aralığında normalize edilmesi ile enerji değerleri hesaplanmaktadır. Alg kolonisin her bir iterasyonda yapabileceği helisel hareket sayısı, yani çözüm uzayında kaç defa yer değiştirme hakkının olduğu (aday çözüm üretimi), sahip olduğu enerji miktarına göre belirlenir (Uymaz ve ark., 2015b).

Gerçekleştirilen helisel harekete bağlı olarak alg kolonileri tarafından bir miktar enerji tüketilmektedir. Helisel hareket sonrasında, i'nci alg kolonisinin *t+1* anındaki enerjisi, Denklem 3.9 kullanılarak hesaplanmaktadır (Uymaz, 2015).

$$E_i^{t+1} = E_i^t - \frac{e}{2} \tag{3.9}$$

Denklemde, E_i^t i'nci alg kolonisinin helisel hareketinden önceki enerjisini, E_i^{t+1} i'nci alg kolonisinin helisel hareketini gerçekleştirdikten sonraki enerjisini ve *e* ise algoritmaya özgü kontrol parametresi olan enerji kaybı (energy loss) parametresini ifade etmektedir (Uymaz ve ark., 2015b).

Alg kolonisinin, helisel hareketi sonrasında çözüm uzayında daha iyi bir pozisyona gidip-gitmemesi durumuna göre kaybettiği enerji miktarı değişmektedir. Bulunduğu noktadan, uygunluk değeri daha iyi bir noktaya hareket ederek daha iyi bir çözüm bulan alg kolonisi, sadece helisel hareketinden dolayı enerji kaybına uğramakta, daha iyi bir çözüm bulamadığı takdirde ise kaybettiği enerjiye ek olarak Denklem 3.9'a göre tekrar bir enerji kaybına uğramaktadır. Helisel hareket sonrasında çözüm uzayında daha iyi bir pozisyon bulamayan alg kolonisinin enerjisi, enerji kaybı (energy loss) parametresinin (*e*) değeri kadar azaltılmış olmaktadır. Bir alg kolonisi, enerjisi tükenene kadar, arama uzayında yeni lokasyonlar bulmaya çalışır, enerjisi tükendiğinde daha iyi

bir çözüm bulunamadı ise bu koloninin açlık(starvation) sayacı 1(bir) arttırılır (Uymaz, 2015).

Bütün alg kolonileri için helisel hareket aşaması tamamlandıktan, dolayısıyla bütün alg kolonilerinin enerjileri tükendikten sonra evrimsel süreç işletilir. Bütün alg kolonilerinin büyüklük değerleri Denklem 3.2 kullanılarak tekrar hesaplanır. Algoritmadaki evrim sürecinde, hesaplanan büyüklük değerine göre popülasyondaki en büyük ve en küçük alg kolonisi belirlenir ve en büyük alg kolonisinin rastgele seçilen bir boyutu, en küçük alg kolonisinin aynı boyutuna Denklem 3.10 kullanılarak kopyalanır (Uymaz ve ark., 2015b).

$$x_{s,n}^{t+1} = x_{b,n}^{t} \quad n \in \{1,2,\dots,D\} \tag{3.10}$$

Denklemde, $x_{s,n}^{t+1}$ en küçük alg kolonisinin rastgele seçilmiş n'inci boyutunun *t+1* anındaki değerini, $x_{b,n}^{t}$ en büyük alg kolonisinin rastgele seçilmiş n'inci boyutunun *t* anındaki değerini ve D ise problemin boyutunu ifade etmektedir (Uymaz, 2015).

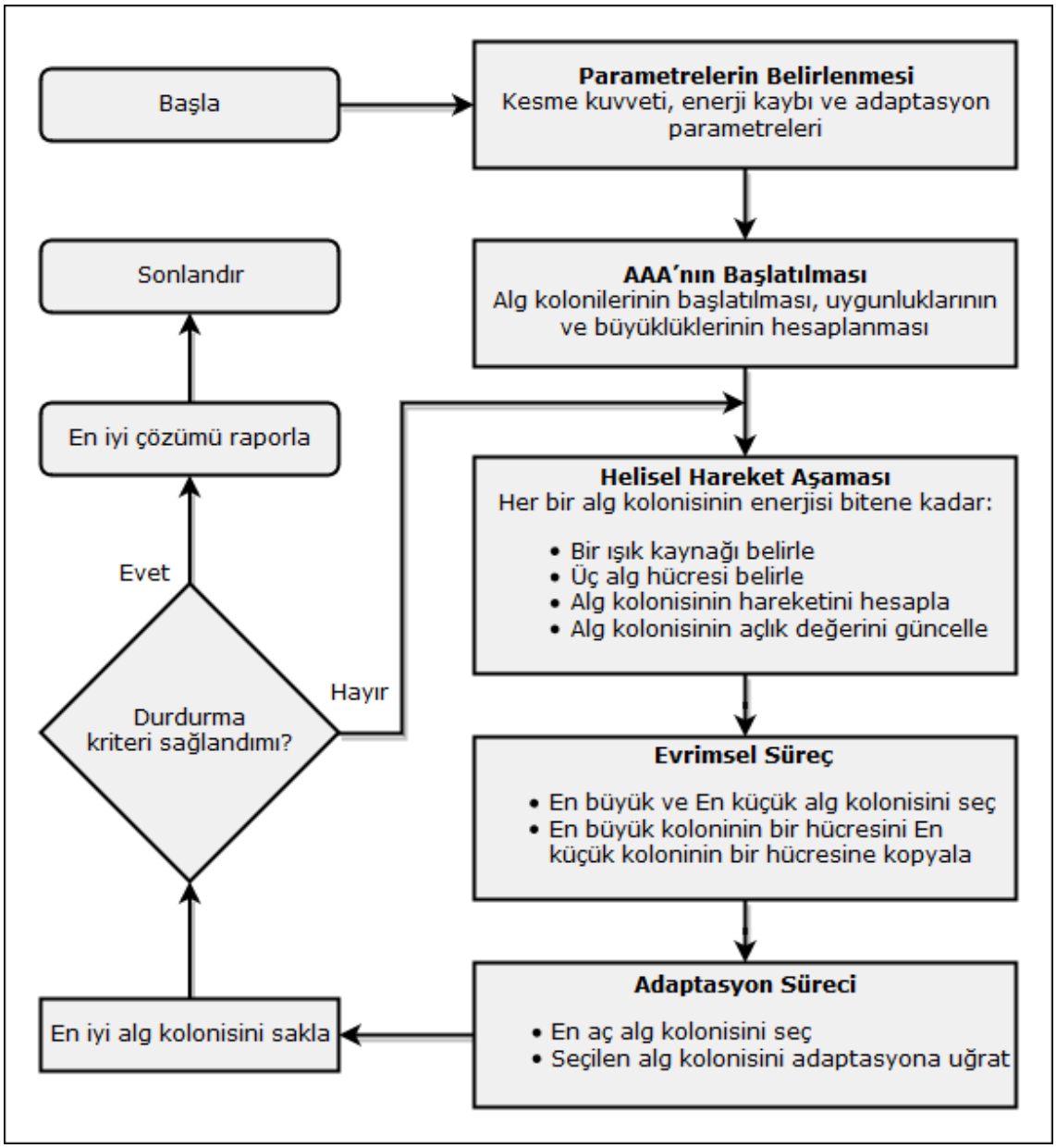
AAA'da, evrimsel süreçten hemen sonra adaptasyon aşamasına geçilir. Adaptasyon aşaması da evrimsel hareket aşaması gibi her bir iterasyonda bir defa işletilir. Evrimsel hareket aşamasının işletilmesi için bir şart bulunmazken, adaptasyon aşamasının her bir iterasyonda çalışıp çalışmayacağına, yönteme özgü kontrol parametresi olan adaptasyon oranı (adaptation rate) kullanılarak karar verilir. [0,1] aralığında rastgele belirlenmiş bir sayı adaptasyon oranı (adaptation rate) parametresinden küçük ise adaptasyon aşaması işletilmekte, aksi taktirde ilgili iterasyon için adaptasyon aşaması çalıştırılmadan diğer işlemlere geçilmektedir (Uymaz ve ark., 2015b).

Adaptasyon aşaması, yeterince büyüyememiş olan alg kolonisinin en büyük alg kolonisine benzemeye çalıştığı bir süreç olup alg kolonilerinin açlık seviyelerine göre belirlenmektedir. Algoritma ilklendirildiğinde bütün alg kolonileri için açlık (starvation) sayacı değeri 0 (sıfır) olarak belirlenir ve gerçekleştirilen helisel hareketler sonrasında enerjisi tükenen alg kolonisi için bu değer 1 (bir) artırılır. Adaptasyon aşamasında popülasyondaki açlık sayacı değeri en yüksek (en aç) olan alg kolonisi belirlenir. Belirlenen en aç alg kolonisinin her bir boyutu, en büyük alg kolonisi kullanılarak Denklem 3.11'deki gibi hesaplanmaktadır (Uymaz ve ark., 2015a).

$$x_{st,j}^{t+1} = x_{st,j}^{t} + r\left(x_{b,j}^{t} - x_{st,j}^{t}\right) \;,\; j = 1,2,\dots,D \tag{3.11}$$

Denklemde, x_{st} en aç alg kolonisini, x_b en büyük alg kolonisini, r [0,1] aralığında üretilmiş rastgele sayıyı, D problemin boyutunu ve j ise boyut indisini ifade etmektedir.

Algoritmanın işletilmesi süresince hesaplanan uygunluk değeri sayısı, algoritmanın parametrelerinden olan Maksimum Fonksiyon Çağırma Sayısına (Maximum Number of Function Evaluations-MaxFEs) eriştiğinde, algoritma sonlanır ve popülasyondaki en iyi çözüm raporlanır. Yapay alg algoritmasının genel akış şeması Şekil 3.1'de verilmiştir.



Şekil 3.1 AAA'nın genel akış şeması (Uymaz, 2015).

3.2. Optimizasyon Problemleri

İkili optimizasyon algoritmalarının test edilebilmesi ve karşılaştırılabilmesi için literatürde sunulmuş birçok test veri kümesi mevcuttur (Özellik Seçimi, Birim Yüklenme Problemleri, Sırt Çantası Problemleri, ambulans istasyonu yer seçimi, itfaiye lokasyon seçimi, tesis yerleşim problemleri, çizelgeleme problemleri vb.). Tez kapsamında yeni ikilileştirme yöntemleri geliştirildiğinden dolayı araştırmacılar ve okuyucular açısından daha faydalı olması açısından ambulans istasyonu yer seçimi, itfaiye lokasyon seçimi ve çizelgeleme problemleri gibi gerçek dünya problemleri yerine yöntemlerin performanslarını analiz etmek amacıyla literatürdeki iyi bilinen ve güncel kıyas problemleri kullanılmıştır. Bu problemlerin matematik modelleri ve anlatımları alt başlıkta sunulmuştur.

3.2.1 Kapasitesiz Tesis Yerleşim Problemleri (UFLP)

Bu tezde, karşılaştırmalarda kullanılan test problemlerinden ilki olan ve literatürde sıklıkla kullanılan Kapasitesiz Tesis Yerleşim Problemleri (Uncapacitated Facility Location Problems-UFLP), 15 adet problemi içeren bir set olarak OR-Library'den (Beasley, 1990a) alınmıştır. UFLP setindeki problemler sadece ikili karar değişkenlerine sahip olup, bu özelliği nedeniyle tez kapsamında önerilen yöntemlerin performans analizleri ve karşılaştırmaları öncelikle bu problem seti kullanılarak yapılmıştır. Problemin temelinde müşteriler, tedarikçiler (tesisler) ve bu ikisi arasında gerçekleşen siparişlerden dolayı oluşan maliyetler bulunmaktadır. Her bir tesisin bir açılış maliyeti ve müşterilerin her bir tesise olan uzaklığına bağlı olarak ulaşım maliyetleri verilmektedir. Her bir müşterinin talebi tesisler tarafından karşılanmak zorunda olup tesislerin herhangi bir kapasite kısıtlaması bulunmamaktadır. Bir başka ifadeyle, tek bir tesisin açık olması durumunda bile bu tesis bütün müşterilere hizmet verebilmektedir. Problemin amacı, bütün müşterilerin hizmet alacağı şekilde, minimum toplam maliyetle açılacak tesislerin belirlenmesidir. Bütün tesislerin sayısı ve bulundukları konum bilinmekte ancak bu tesislerden hangilerinin açılacağı bilinmemektedir. Birden fazla tesis açılması durumunda, müşteriler, en düşük ulaşım maliyetine sahip olan tesisten hizmet alacaktır (Ghosh, 2003). Problemde n adet potansiyel tesis yeri mevcut olduğunda ve en az bir adet tesisin açık olması şartı ile olası çözümlerin sayısı 2^n-1 olacaktır. Tesis sayısının artması ile olası çözümlerin sayısının üstel olarak arttığından dolayı UFLP setindeki problemler

NP-Zor (Np-Hard) problem kategorisinde yer almaktadır (Krarup ve Pruzan, 1983; Monabbati ve Kakhki, 2015). Bu tip problemlerin çözümünde klasik yöntemler yetersiz kalmakta, tam çözümü garanti etmeseler bile, birçok evrimsel veya sürü zekâsı tabanlı yaklaşımlar literatürde yaygın olarak kullanılmaktadır. UFLP setindeki problemlerin matematiksel modeli (Ghosh, 2003) Denklem 3.12'de verilmiştir:

$AA = \{aa_1, aa_2, \dots, aa_{mm}\}$, Potansiyel tesis yerleri kümesi

$JJ = \{jj_1, jj_2, \dots, jj_{nn}\}$, Müşterilerin kümesi

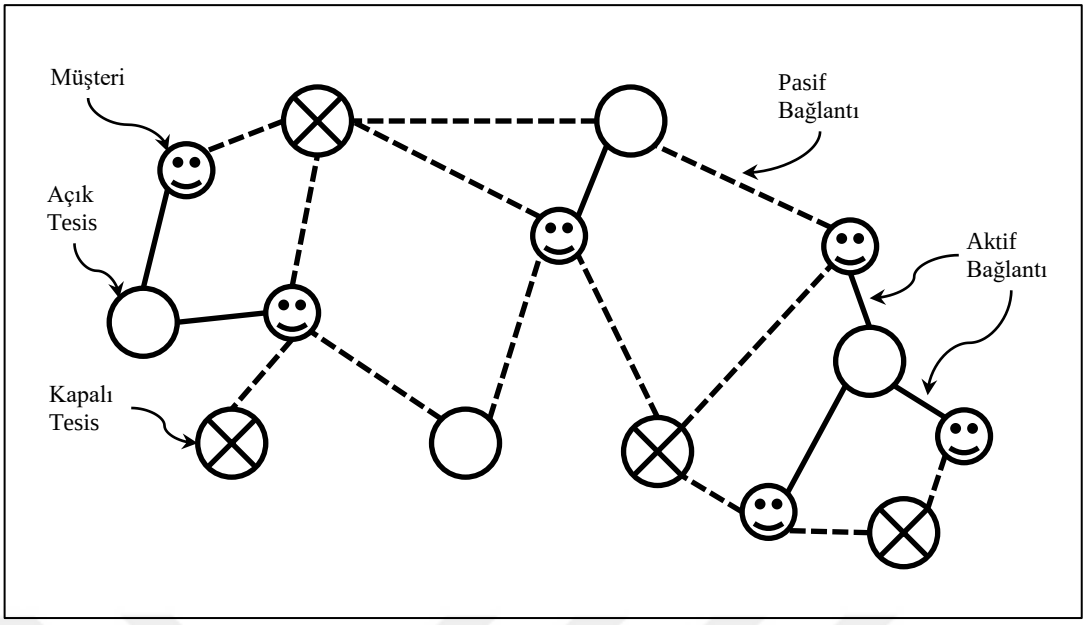
$FF = \{ff_1, ff_2, \dots, ff_{mm}\}$, Her tesis için tanımlanmış olan tesislerin sabit maliyeti

$$CC = \begin{bmatrix} cc_{1,1} & cc_{1,2} & \dots & cc_{1,nn} \\ \dots & & & \\ cc_{mm,1} & cc_{mm,2} & \dots & cc_{mm,nn} \end{bmatrix}$$, Transfer maliyet matrisi

$$\arg min = \left\{ \delta(SS) = \sum_{aa \in SS} ff_{aa} + \sum_{jj \in JJ} min\left\{ cc_{aa,jj} \middle| aa \in SS \right\} \right\} \quad (3.12)$$

Denklemde, *SS* kümesi, *AA* kümesinin boş olmayan alt kümesidir. Problemin çözümü için minimum maliyet koşulunu sağlayan *SS* kümesinin bulunması amaçlanmaktadır. Bir tesis *SS* alt kümesinin üyesi ise 1, aksi takdirde 0 olarak belirtilmektedir. Problemde *mm* adet potansiyel tesis yeri mevcut ise, çözümün temsili mm-boyutlu, ikili değerler alabilen bir vektör olacaktır. Bu çözüm vektöründeki her bir boyut bir tesisi temsil etmekte ve ilgili boyuttaki ikili değer tesisin açık veya kapalı olduğunu ifade etmektedir. Örneğin, çözüm vektöründe bulunan beşinci boyutun ikili değeri *1* ise beşinci tesis açık, *0* ise bu tesis açılmamış yani kapalı anlamındadır. Problemin daha iyi anlaşılabilmesi için basit bir gösterimi Şekil 3.2'de verilmiştir.

Şekil 3.2'de açık tesisler, kapalı tesisler ve müşteriler farklı şekiller ile ifade edilmiştir. Müşteriler ile tesisler arasındaki bağlantılar ise düz ve kesikli çizgiler ile belirtilmiştir. Her bir müşteri açık tesislerden en yakın olanından hizmet almaktadır. Düz çizgiler (aktif bağlantı) müşterinin hizmet aldığı tesisi, kesikli çizgiler (pasif bağlantı) ise müşterinin hizmet almadığı tesis veya tesisleri göstermektedir. Müşterilerin hizmet almadığı bütün tesisler ile arasında pasif bağlantı olmasına karşın, şeklin daha anlaşılır olması adına sadece müşterinin çevresindeki tesisler ile arasındaki pasif bağlantılar gösterilmiştir.



Şekil 3.2. UFLP setindeki problemlerin basit bir gösterimi

Karşılaştırmalarda kullanılan UFLP setine ait 15 adet problemin adı, boyutsallığı (tesis sayısı X müşteri sayısı) ve optimum maliyet bilgisi Çizelge 3.1'de verilmiştir.

Çizelge 3.1. Karşılaştırmalarda kullanılan UFLP test seti

Problem Adı	Boyutu	Optimum Değeri
Cap71	16 X 50	932615.75
Cap72	16 X 50	977799.40
Cap73	16 X 50	1010641.45
Cap74	16 X 50	1034976.98
Cap101	25 X 50	796648.44
Cap102	25 X 50	854704.20
Cap103	25 X 50	893782.11
Cap104	25 X 50	928941.75
Cap131	50 X 50	793439.56
Cap132	50 X 50	851495.33
Cap133	50 X 50	893076.71
Cap134	50 X 50	928941.75
CapA	100 X 1000	17156454.48
CapB	100 X 1000	12979071.58
CapC	100 X 1000	11505594.33

Çizelge 3.1'de verilen ilk dört problem (Cap71, Cap72, Cap73 ve Cap74) 16 adet tesis ve 50 adet müşteri barındıran, *az boyutlu* olarak isimlendirilen problemlerdir. Genel olarak, bu problemleri çözmek diğerlerine göre nispeten daha kolaydır. Cap101, Cap102, Cap103 ve Cap104 problemlerinde, 25 adet tesis ve 50 adet müşteri olup *orta boyutlu* olarak isimlendirilmiştir. Cap131, Cap132, Cap133 ve Cap134 problemlerinde 50 adet tesis ve 50 adet müşteri olup *büyük boyutlu* problemlerdir. *Çok büyük boyutlu* problemler olarak anılan CapA, CapB ve CapC problemleri ise 100 adet tesis ve 1000 adet müşteri sayısına sahip olup 2^{100}-1 olası çözümü mevcuttur. Çözüm uzayı, tesis sayısına bağlı olarak üstel bir şekilde katlanarak arttığından dolayı bu problemleri çözmek diğerlerine göre daha zordur.

3.2.2 Sürekli Nümerik Fonksiyonlar: CEC2015 Test Seti

Literatürde önerilen ikili optimizasyon algoritmalarının performansları sıklıkla sürekli fonksiyonların optimizasyonu üzerinde de araştırılmaktadır (Omran ve ark., 2005; Sadri ve Suen, 2006; Khanesar ve ark., 2007). Sürekli karar değişkenlerine sahip fonksiyonlarının optimizasyonu için evrimsel hesaplama veya sürü zekası algoritmalarının kullanılabilmesine rağmen problemin istenilen şekilde genelleştirilebilmesi ve optimum çözümlerinin bilinmesinden dolayı, ikili optimizasyon yöntemlerinin birbirleriyle kıyaslanması bu problemler üzerinde kolay ve açıklayıcı olmaktadır. Bundan dolayı araştırmacılar ikili optimizasyon algoritmalarının performanslarını bu tip nümerik fonksiyonlar üzerinde de test etmektedirler. Bu bağlamda, tez kapsamında önerilen yöntemlerin performansları, UFLP'nin yanı sıra sürekli nümerik fonksiyonlarda da kıyaslanmıştır.

Optimizasyon alanında, her yıl çeşitli konu başlıkları ile düzenlenen, konular için farklı test setleri sunulan, belirlenen konularda yarışmaların düzenlendiği ve literatürde alanında saygın ve güvenilir bir yer edinmiş olan CEC (IEEE Congress on Evolutionary Computation) kongresinde 2015 yılında sunulmuş olan CEC2015 (bound constrained single-objective computationally expensive numerical optimization problems) (Chen ve ark., 2014) problemleri, ikinci bir test seti olarak karşılaştırmalarda kullanılmıştır. CEC2015 test seti 4 ana kategoriden ve 15 farklı problemden oluşmaktadır. Bu kategoriler Tek Tepeli Fonksiyonlar (Unimodal Functions), Basit Çok Tepeli Fonksiyonlar (Simple Multimodal Functions), Melez Fonksiyonlar (Hybrid Functions) ve Birleşik Fonksiyonlardır (Composition Functions). Bu fonksiyonlarla ilgili ayrıntılı bilgi

Çizelge 3.2'de verilmiştir. Tüm CEC2015 problemleri minimizasyon problemleri olup karar değişkenleri [-100,100] aralığındadır ve Çizelge 3.2'deki son sütun (Fmin) fonksiyonların optimum değerini göstermektedir.

Çizelge 3.2. Karşılaştırmalarda kullanılan cec2015 test seti

Kategori	No.	Fonksiyonlar	İlgili Temel Fonksiyonlar	Fmin
Tek Tepeli Fonksiyonlar (Unimodal Functions)	1	Rotated Bent Cigar	Bent Cigar	100
	2	Rotated Discus	Discus	200
Basit Çok Tepeli Fonksiyonlar (Simple Multimodal Functions)	3	Shifted and Rotated Weierstrass	Weierstrass	300
	4	Shifted and Rotated Schwefel's	Schwefel's	400
	5	Shifted and Rotated Katsuura	Katsuura	500
	6	Shifted and Rotated HappyCat	HappyCat	600
	7	Shifted and Rotated HGBat	HGBat	700
	8	Shifted and Rotated Expanded Griewank's plus Rosenbrock's	Griewank's Rosenbrock's	800
	9	Shifted and Rotated Expanded Scaffer's F6	Expanded Scaffer's F6	900
Melez Fonksiyonlar (Hybrid Functions)	10	Melez Fonksiyon 1 (N = 3)	Schwefel's Rastrigin's High Conditioned Elliptic	1000
	11	Melez Fonksiyon 2 (N = 4)	Griewank's Weierstrass Rosenbrock's Scaffer's F6	1100
	12	Melez Fonksiyon 3 (N = 5)	Katsuura HappyCat Griewank's Rosenbrock's Schwefel's Ackley's	1200
Birleşik Fonksiyonlar (Composition Functions)	13	Birleşik Fonksiyon 1 (N = 5)	Rosenbrock's High Conditioned Elliptic Bent Cigar Discus High Conditioned Elliptic	1300
	14	Birleşik Fonksiyon 2 (N = 3)	Schwefel's Rastrigin's High Conditioned Elliptic	1400
	15	Birleşik Fonksiyon 3 (N = 5)	HGBat Rastrigin's Schwefel's Weierstrass High Conditioned Elliptic	1500

Test setindeki, temel fonksiyonların ve bu temel fonksiyonlar kullanılarak oluşturulmuş test fonksiyonlarının denklemlerine ve daha ayrıntılı açıklamalarına (Chen ve ark., 2014) çalışmasından erişilebilir.

CEC2015 setindeki problemlerin karar değişkenleri sürekli değerler aldığından dolayı, ikili optimizasyon yöntemleri ile bu problemleri çözebilmek ve her bir bireyin uygunluk değerlerinin hesaplanabilmesi için ikili değerlerin sürekli değerlere dönüştürülmesi gerekmektedir. Bu dönüşüm için kullanılan denklem aşağıda verilmiştir:

$$C_j = L_j + \frac{(H_j - L_j) DecVal_j}{MaxVal} \tag{3.13}$$

Denklemde, C_j bireyin j'inci boyutunun sürekli değerini, H_j j'inci boyutun alabileceği değerin üst sınırını, L_j j'inci boyutun alabileceği değerin alt sınırını, $DecVal_j$ j'inci boyut için verilen ikili sayının ondalık sayı sistemindeki tamsayı değerini ve $MaxVal$ ise her boyut için belirlenen bit uzunluğuna göre ikili sayının en fazla alabileceği ondalık tamsayı değerini ifade etmektedir.

İkili değerler ile temsil edilen boyut için 4 bit uzunluk belirlendiği, boyuttaki ikili değerin "1011" olduğu ve boyutun sürekli değerinin [-10,+10] aralığında olduğu varsayıldığında bu boyut için sürekli değer dönüşümü aşağıdaki gibi örneklenmiştir:

$$C_j = -10 + \frac{\big(10 - (-10)\big) \times 11}{15}$$
$$C_j = -10 + 14.66$$
$$C_j = 4.66$$

Örnekte, ikili optimizasyon yöntemi, popülasyondaki bir bireyin j'inci boyutu için "1011" ikili değerini ürettiği varsayılmış ve bu bireyin uygunluk değerlerinin hesaplanabilmesi için ikili değer Denklem 3.13 kullanılarak sürekli 4.66 değerine dönüştürülmüştür.

4. AAA TABANLI YENİ İKİLİ YAKLAŞIMLAR VE DENEYSEL ÇALIŞMALAR

4.1. AAA İle Yeni Bir İkilileştirme Operatörü (binAAA)

4.1.1. Önerilen Yöntem

İkili optimizasyon problemlerinde, karar değişkenleri {0,1} kümesinin bir öğesi olduğundan dolayı, AAA algoritmasının önerilen temel versiyonu bu tip problemler için çözüm üretememektedir. Bunun sebebi AAA algoritmasındaki iç mekanizmalardan olan helisel hareket ve adaptasyon mekanizmalarının kayan nokta sayılar (gerçek sayılar) ile çalışacak şekilde tasarlanmış olmasıdır. Bu nedenle, bu mekanizmaların çıktısı, ikili optimizasyon problemlerinin amaç fonksiyonunun değerlendirilmesi için doğrudan kullanılamamaktadır. Bu sorunun çözümü için, önceki bölümlerde belirtildiği üzere literatürde çeşitli ikilileştirme yöntemleri önerilmiştir. Bu tez kapsamında, literatürdeki ikilileştirme yöntemlerine alternatif olarak, basit ama etkili yeni bir sezgisel ikilileştirme operatörü önerilmiştir. İkili optimizasyon problemlerinin çözülebilmesi için bu yeni operatör AAA'ya uyarlanmış ve İkili AAA (binary AAA-binAAA) yöntemi önerilmiştir (Korkmaz ve ark., 2018).

Önerilen yöntemde, alg kolonileri ikili değerler ile ilklendirilmektedir. Popülasyondaki alg kolonilerinden yeni aday çözümlerin elde edilebilmesi için helisel hareket fazı, ikili değerler ile çalışabilecek şekilde yeniden uyarlanmıştır. AAA algoritmasındaki helisel hareket fazı için önerilen yeni sezgisel yöntemin denklemi aşağıdaki gibi verilmiştir:

$$V_i = X_i$$
$$P = \{ m, k, l \} \ , \ m, k, l \in \{1,2, \dots, D\} \ , \ m \neq k \neq l \qquad (4.1)$$
$$V_{i,p} = \begin{cases} X_{kt,p} & , if\left(r_{i,p} > NR\right) \\ \sim X_{kt,p} & , aksi\ takdirde \end{cases} \qquad \forall p \in P$$

Denklemde, X_i mevcut çözümü (alg kolonisi), X_{kt} turnuva seçim yöntemiyle popülasyondan rastgele seçilmiş komşu çözümü (ikili çözüm vektörleri) ve V_i aday çözümü temsil etmektedir. m, k ve l ise bütün çözümler (X_i, X_{kt}, V_i) için rastgele belirlenen karar değişkenleri indisini ifade etmektedir. $\sim X_{kt,p}$ ifadesi $X_{kt,p}$ ifadesinin

mantıksal olarak değillenmiş, yani tersinin alınmış halidir. Örneğin $X_{kt,p}$ değeri 1 (bir) ise, bunun tersi olan $\sim X_{kt,p}$ değeri 0 (sıfır), $X_{kt,p}$ değeri 0 (sıfır) ise bunun değillenmiş $\sim X_{kt,p}$ değeri 1 (bir) olacaktır. Denklemde, $r_{i,p}$, [0,1] aralığında üretilen rastgele sayıyı ve NR ise yönteme eklenen mantıksal değil (logical not) olasılık parametresi olup [0,1] aralığında değerler alabilmektedir. Yöntem ilklendirilirken *NR* parametre değerini 0 (sıfır) olarak verdiğimizde, her bir boyutun komşu çözümden olduğu gibi kopyalanacağını garanti etmiş oluruz. Bunun aksine, NR parametresine 1 (bir) değerini verirsek, komşu çözümün ilgili boyutundaki ikili değerin mantıksal olarak değili (tersi) alınarak aday çözüme kopyalanacaktır. Eğer önerilen yöntemde Mantıksal Değil operatörü kullanılmamış olsaydı, her bir kolonininin aynı hücresinin aynı ikili değere sahip olması durumunda, bu hücrenin değeri, iterasyon boyunca değiştirilemeyecek ve durağanlık (stagnation) meydana gelecekti. Bu da algoritmanın arama yeteneğini kısıtlayacağı için, en iyi çözüme ulaşmasını geciktirecek veya engelleyecekti.

AAA'da yapılan diğer bir değişiklik ise Adaptasyon Sürecinde gerçekleştirilmiştir. AAA'nın sürekli versiyonundaki adaptasyon sürecindeki işlemler, sürekli çözüm uzayında tanımlanmış değerler üzerinde çalışmakta ve bu hali ile ikili değerler alan problemlerin çözümünde kullanılamamaktadır. AAA'nın temel versiyonunda, adaptasyon parametresi, her bir iterasyon için adaptasyon sürecinin işletilip işletilmemesine karar vermek için kullanılmaktadır. binAAA'da ise adaptasyon parametresi iki amaç için kullanılmaktadır. İlk amaç temel versiyondaki ile aynı olup, [0,1] aralığında rastgele üretilen bir sayı eğer adaptasyon parametresinden küçük ise bulunan iterasyonda adaptasyon süreci işletilmekte, değilse işletilmemektedir. İkinci amaçta, eğer adaptasyon süreci işletilecekse, her bir boyut için [0,1] aralığında rastgele üretilen sayı adaptasyon parametresinden küçük ise, popülasyondaki (alg kolonileri) en büyük kolonin (biggest colony) ilgili boyutu en aç kolonin (most starveling colony) boyutuna kopyalanmaktadır. Bu değişiklik ile popülasyonun iterasyon boyunca durağanlaşmaması (stagnation) amaçlanmıştır. Eğer popülasyondaki (alg kolonileri) en büyük kolonin (biggest colony) bütün boyutu en aç kolonin (most starveling colony) tümüne kopyalansaydı popülasyonda aynı karar değişkenlerine sahip iki birey olacaktı ve iterasyon ilerledikçe aynı bireylerin sayısı giderek artacaktı. Bu durum arama uzayındaki hareket kabiliyetini olumsuz yönde etkileyecek ve çözüme ulaşmada performansı kötü etkileyecekti.

4.1.2. Deneysel Sonuçlar ve Karşılaştırmalar

Yöntemlerin adil bir şekilde karşılaştırılabilmesi için, yöntemlerin ortak kontrol parametreleri birbirlerine eşit olarak seçilmiştir. Bu bağlamda bütün yöntemlerde, popülasyon büyüklüğü 40 ve durdurma kriteri olarak MaxFEs değeri 80,000 olarak kullanılmıştır. GA'da, Holland (1992a) tarafından yapılan çalışmada önerildiği üzere, çaprazlama oranı (crossover rate) 0.8 ve mutasyon oranı (mutation rate) 0.01 olarak seçilmiştir. Ebeveyn seçiminde turnuva seçim yöntemi kullanılmış, turnuva boyutu 10 olarak seçilmiştir. GA için 3 farklı çaprazlama operatörü kullanılmış ve bunların her birinin sonuçları binAAA ile ayrı ayrı kıyaslanmıştır. AAA'nın daha önce uyarlanmış ikili versiyonları (Sigmoid Transfer Fonksiyonu Kullanılmış İkili AAA- BAAA-Sig, Tanjant Hiperbolik Transfer Fonksiyonu Kullanılmış İkili AAA- BAAA-Tanh) (Zhang ve ark., 2016) makaleye uygun olarak tekrar kodlanmıştır. Adil bir karşılaştırma amacı ile elit yerel arama (elite local search) mekanizması yönteme dâhil edilmemiştir. Yöntemin önerildiği makalede (Zhang ve ark., 2016) kullanıldığı gibi, lojistik fonksiyonlarda, τ değeri BAAA-Tanh için 1.5 ve BAAA-Sig için 2 olarak kullanılmıştır. Ayrıca Zhang ve ark. (2016), AAA'nın (Uymaz ve ark., 2015b) temel çalışmasında olduğu gibi, enerji kaybı kontrol parametresini 0.3, kesme kuvveti parametresini 2 ve adaptasyon parametresini 0.5 olarak kullanmışlardır. Açı Modülasyonlu PSO'nun (AMPSO) kendine özgü kontrol parametreleri, sosyal ve bilişsel bileşenlerin katsayıları (coefficients of social and cognitive components) 1.49618 ve atalet ağırlığı (inertia weight) 0.729844 olarak kullanılmıştır. AMPSO'daki komşuluk topolojisi, 8x5 dikdörtgen matris olarak Von Neumann topolojisi kullanılmıştır (Pampara ve ark., 2005). binAAA yönteminde, algler ayrık çözüm uzayında çalıştığından dolayı kesme kuvveti parametresi kullanılmamaktadır. AAA'nın (Uymaz ve ark., 2015b) temel çalışmasında olduğu gibi, enerji kaybı kontrol parametresi 0.3 ve adaptasyon kontrol parametresi 0.5 olarak kullanılmıştır. binAAA yöntemine özgü NR kontrol parametresi ise 0.33 olarak kullanılmıştır.

binAAA yöntemi ilk olarak, Tek Noktadan Çaprazlamalı Genetik Algoritma (GA with Single-Point Crossover- GA-SP), İki Noktadan Çaprazlamalı Genetik Algoritma (GA with Two-Point Crossover- GA-TP), Bütün Noktalardan Çaprazlamalı Genetik Algoritma (GA with Uniform Crossover- GA-UP), BAAA-Tanh, BAAA-Sig ve AMPSO algoritmaları ile kıyaslanmıştır. Her bir problem için tüm algoritmalar 30 kez bağımsız

olarak çalıştırılmış ve elde edilen sonuçlardan *ortalama değer*, *standart sapma*, *optimum değeri bulma sayısı*, *ortalama çalışma süresi(saniye)* ve *GAP* değeri olarak Çizelge 4.1 – 4.4'de sunulmuştur. Çizelgelerdeki *GAP* değerleri aşağıdaki gibi hesaplanmaktadır:

$$Gap = \frac{f_{ort} - f_{opt}}{f_{opt}} \times 100 \quad (4.2)$$

Denklemde, *Gap* değeri yöntemin 30 defa çalıştırma sonucunda elde ettiği ortalama değer ile problemin optimum değeri arasındaki oransal fazlalığı, f_{ort} yöntemin 30 defa çalıştırma sonucunda elde ettiği ortalama değeri, f_{opt} ise probleminin optimum değerini ifade etmektedir.

Yapılan karşılaştırmanın daha anlaşılır ve görünür hale gelmesi için, her bir problem için en iyi algoritmanın ürettiği sonuçlar kalın yazı tipinde verilmiştir. Sonuçların daha iyi incelenebilmesi adına, problemler boyutlarına göre *az boyutlu, orta boyutlu, büyük boyutlu ve çok büyük boyutlu* olmak üzere 4 farklı grupta incelenmiş ve Çizelge 4.1-4.4'de gösterilmiştir.

Çizelge 4.1. binAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, az boyutlu Cap71, Cap72, Cap73 ve Cap74 problemleri üzerinde karşılaştırılması

Yöntem		Cap71	Cap72	Cap73	Cap74
GA-SP	Ortalama	**932615.750**	**977799.400**	1011314.476	**1034976.975**
	Gap	**0.00000**	**0.00000**	0.06659	**0.00000**
	Opt.Say.	**30**	**30**	19	**30**
	Std.Sap.	**0.000**	**0.000**	899.650	**0.000**
	Ort.Süre	**26.957**	**27.893**	27.994	**27.998**
GA-TP	Ortalama	**932615.750**	**977799.400**	1011130.923	**1034976.975**
	Gap	**0.00000**	**0.00000**	0.04843	**0.00000**
	Opt.Say.	**30**	**30**	22	**30**
	Std.Sap.	**0.000**	**0.000**	825.576	**0.000**
	Ort.Süre	**27.568**	**28.056**	28.050	**28.143**
GA-UP	Ortalama	**932615.750**	**977799.400**	1011069.739	**1034976.975**
	Gap	**0.00000**	**0.00000**	0.04238	**0.00000**
	Opt.Say.	**30**	**30**	23	**30**
	Std.Sap.	**0.000**	**0.000**	789.612	**0.000**
	Ort.Süre	**27.744**	**28.103**	28.101	**28.095**
BAAA-Tanh	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	Ort.Süre	**25.748**	**26.279**	**28.221**	**27.584**
BAAA-Sig	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	Ort.Süre	**25.868**	**26.927**	**28.219**	**27.461**
AMPSO	Ortalama	934372.581	980841.195	1011882.871	1035707.668
	Gap	0.18838	0.31109	0.12283	0.07060
	Opt.Say.	6	1	4	22
	Std.Sap.	1274.961	1706.767	1113.884	1232.432
	Ort.Süre	47.783	53.559	53.032	48.863
binAAA	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	Ort.Süre	**25.087**	**25.231**	**25.236**	**25.181**

Çizelge 4.2. binAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, orta boyutlu Cap101, Cap102, Cap103 ve Cap104 problemleri üzerinde karşılaştırılması

Yöntem		Cap101	Cap102	Cap103	Cap104
GA-SP	Ortalama	797193.286	**854704.200**	894351.782	**928941.750**
	Gap	0.06839	**0.00000**	0.06374	**0.00000**
	Opt.Say.	11	**30**	6	**30**
	Std.Sap.	421.655	**0.000**	505.036	**0.000**
	Ort.Süre	29.372	**28.730**	28.689	**32.706**
GA-TP	Ortalama	797164.610	**854704.200**	894329.179	**928941.750**
	Gap	0.06479	**0.00000**	0.06121	**0.00000**
	Opt.Say.	12	**30**	10	**30**
	Std.Sap.	428.658	**0.000**	540.160	**0.000**
	Ort.Süre	29.206	**28.931**	28.907	**32.992**
GA-UP	Ortalama	797107.258	**854704.200**	894427.382	**928941.750**
	Gap	0.05759	**0.00000**	0.07220	**0.00000**
	Opt.Say.	14	**30**	9	**30**
	Std.Sap.	436.524	**0.000**	522.784	**0.000**
	Ort.Süre	29.169	**28.870**	28.915	**33.046**
BAAA-Tanh	Ortalama	796677.114	**854704.200**	**893782.113**	**928941.750**
	Gap	0.00360	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	29	**30**	**30**	**30**
	Std.Sap.	157.066	**0.000**	**0.000**	**0.000**
	Ort.Süre	27.473	**26.334**	**25.851**	**25.435**
BAAA-Sig	Ortalama	**796648.438**	**854704.200**	**893782.113**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	Ort.Süre	**26.836**	**26.215**	**25.926**	**26.963**
AMPSO	Ortalama	805091.105	864430.422	902056.103	936284.584
	Gap	1.05977	1.13796	0.92573	0.79045
	Opt.Say.	0	0	1	7
	Std.Sap.	2380.637	2615.374	3372.271	5060.940
	Ort.Süre	45.560	45.307	45.278	45.281
binAAA	Ortalama	**796648.438**	**854704.200**	**893782.113**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	Ort.Süre	**26.348**	**26.116**	**26.231**	**25.800**

Çizelge 4.3. binAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, büyük boyutlu Cap131, Cap132, Cap133 ve Cap134 problemleri üzerinde karşılaştırılması

Yöntem		Cap131	Cap132	Cap133	Cap134
GA-SP	Ortalama	793980.104	**851495.325**	893891.911	**928941.750**
	Gap	0.06813	**0.00000**	0.09128	**0.00000**
	Opt.Say.	16	**30**	10	**30**
	Std.Sap.	720.877	**0.000**	685.076	**0.000**
	Ort.Süre	34.017	**35.107**	38.143	**36.748**
GA-TP	Ortalama	794012.905	**851495.325**	893740.954	**928941.750**
	Gap	0.07226	**0.00000**	0.07438	**0.00000**
	Opt.Say.	14	**30**	12	**30**
	Std.Sap.	690.560	**0.000**	655.920	**0.000**
	Ort.Süre	33.885	**35.930**	37.906	**36.609**
GA-UP	Ortalama	793865.023	851517.200	893808.891	**928941.750**
	Gap	0.05362	0.00257	0.08198	**0.00000**
	Opt.Say.	15	29	9	**30**
	Std.Sap.	433.467	119.817	628.654	**0.000**
	Ort.Süre	33.563	36.329	38.038	**36.588**
BAAA-Tanh	Ortalama	793525.591	**851495.325**	893333.515	**928941.750**
	Gap	0.01084	**0.00000**	0.02875	**0.00000**
	Opt.Say.	27	**30**	16	**30**
	Std.Sap.	262.498	**0.000**	324.451	**0.000**
	Ort.Süre	45.498	**57.039**	56.239	**56.622**
BAAA-Sig	Ortalama	**793439.563**	**851495.325**	**893076.713**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	Ort.Süre	**63.656**	**67.132**	**66.943**	**67.612**
AMPSO	Ortalama	828897.989	883474.328	915952.880	953576.580
	Gap	4.46895	3.75563	2.56150	2.65192
	Opt.Say.	0	0	0	0
	Std.Sap.	2906.739	4872.274	6499.467	8649.136
	Ort.Süre	46.668	46.379	46.185	46.106
binAAA	Ortalama	**793439.563**	**851495.325**	893082.539	**928941.750**
	Gap	**0.00000**	**0.00000**	0.00065	**0.00000**
	Opt.Say.	**30**	**30**	29	**30**
	Std.Sap.	**0.000**	**0.000**	31.914	**0.000**
	Ort.Süre	**26.764**	**26.834**	26.676	**26.815**

Çizelge 4.4. binAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, çok büyük boyutlu CapA, CapB ve CapC problemleri üzerinde karşılaştırılması

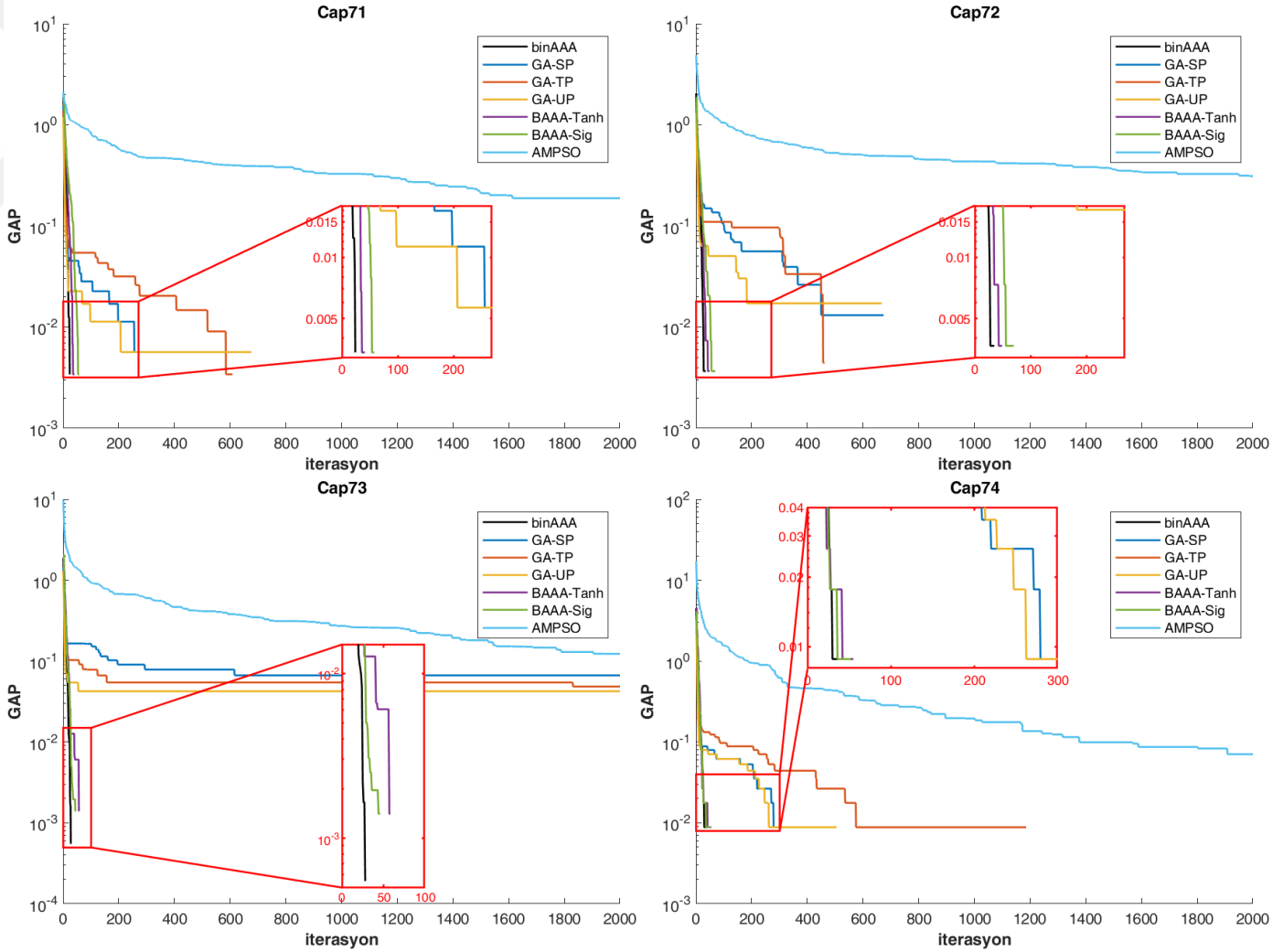
Yöntem		CapA	CapB	CapC
GA-SP	Ortalama	**17164354.456**	**13054858.045**	11586692.969
	Gap	**0.04605**	**0.58391**	0.70486
	Opt.Say.	**24**	**9**	2
	Std.Sap.	**22451.206**	**66658.649**	51848.248
	Ort.Süre	**741.535**	**743.370**	744.455
GA-TP	Ortalama	17205089.145	13063527.186	11577797.524
	Gap	0.28348	0.65071	0.62755
	Opt.Say.	24	11	0
	Std.Sap.	139690.216	89122.485	46346.052
	Ort.Süre	735.939	738.387	741.823
GA-UP	Ortalama	17166811.915	13107633.077	11578600.532
	Gap	0.06037	0.99053	0.63453
	Opt.Say.	24	3	0
	Std.Sap.	35181.974	79714.021	57031.219
	Ort.Süre	743.848	748.144	745.128
BAAA-Tanh	Ortalama	17471223.794	13153617.764	11676427.752
	Gap	1.83470	1.34483	1.48479
	Opt.Say.	3	0	0
	Std.Sap.	225123.921	73978.543	101438.607
	Ort.Süre	864.755	473.660	479.762
BAAA-Sig	Ortalama	17210900.533	13093705.559	11583462.068
	Gap	0.31735	0.88322	0.67678
	Opt.Say.	16	1	1
	Std.Sap.	90743.456	62168.803	45788.678
	Ort.Süre	880.452	475.592	470.911
AMPSO	Ortalama	18276969.242	13913640.156	12452998.999
	Gap	6.53116	7.20058	8.23430
	Opt.Say.	0	0	0
	Std.Sap.	308496.611	127995.741	102830.234
	Ort.Süre	777.856	767.374	538.486
binAAA	Ortalama	17196684.161	13066644.044	**11555763.878**
	Gap	0.23449	0.67472	**0.43604**
	Opt.Say.	21	3	**3**
	Std.Sap.	78259.572	57572.956	**43963.880**
	Ort.Süre	453.994	445.788	**446.128**

binAAA yöntemi *az boyutlu, orta boyutlu ve büyük boyutlu* problemlerden *Cap133* problemi hariç bütün problemlerde, 30 çalıştırmanın hepsinde en uygun çözümü bulmuştur. *Cap133* probleminde ise 29 çalıştırmada optimum değere erişmiş olup sadece

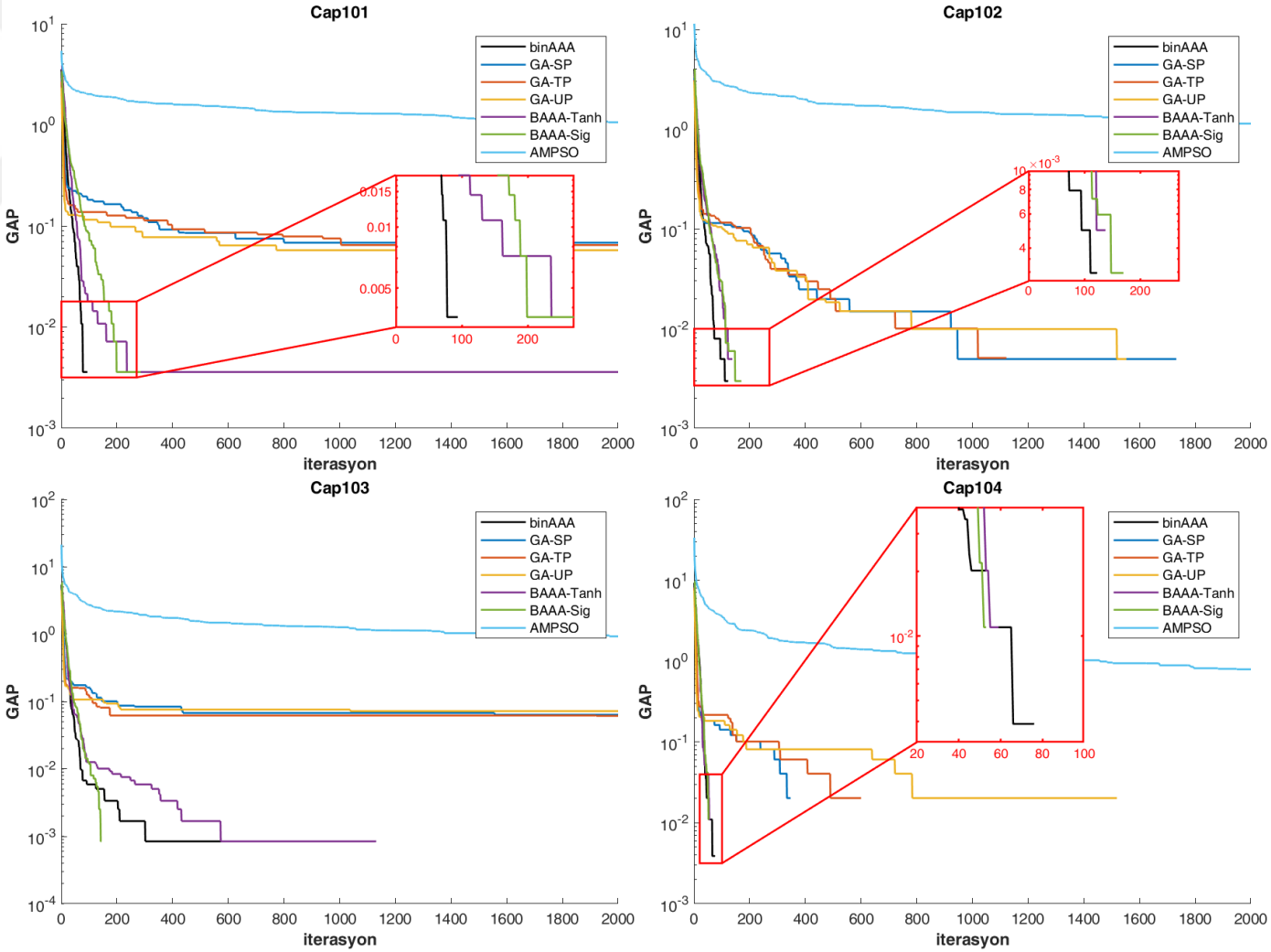
1 çalıştırmada erişememiştir. *Çok büyük boyutlu* problemlere baktığımızda ise CapA ve CapB problemlerinde Genetik Algoritmalar binAAA'dan biraz daha iyi sonuç üretmiş, CapC probleminde ise binAAA yöntemi karşılaştırılan bütün algoritmalar içerisinde en iyi sonucu elde etmiştir. Genetik Algoritmalar çaprazlama operatörü ile iyi bir global arama yeteneğine sahip olduğundan dolayı yüksek boyutlu problemler üzerinde daha kaliteli sonuçlar üretebilmektedir. Fakat mutasyon operatörünün yetersizliğinden dolayı yerel aramada zayıf kalmakta ve düşük boyutlu problemler üzerinde iyi kalitede çözümler üretememektedir.

AMPSO yöntemi haricindeki bütün algoritmalar Cap71, Cap72 ve Cap74 problemleri için optimum sonuca ulaşmıştır. *Standart sapma* değerleri göz önüne alındığında ise binAAA yöntemi problemlerin çözümünde diğer algoritmalara göre daha güçlü bir performans sergilemektedir. Ayrıca yöntemlerin ortalama çalışma süreleri incelendiğinde, binAAA yöntemi Cap103 ve Cap104 problemleri haricinde bütün problemlerde en az çalışma süresine sahip olduğu görülmektedir.

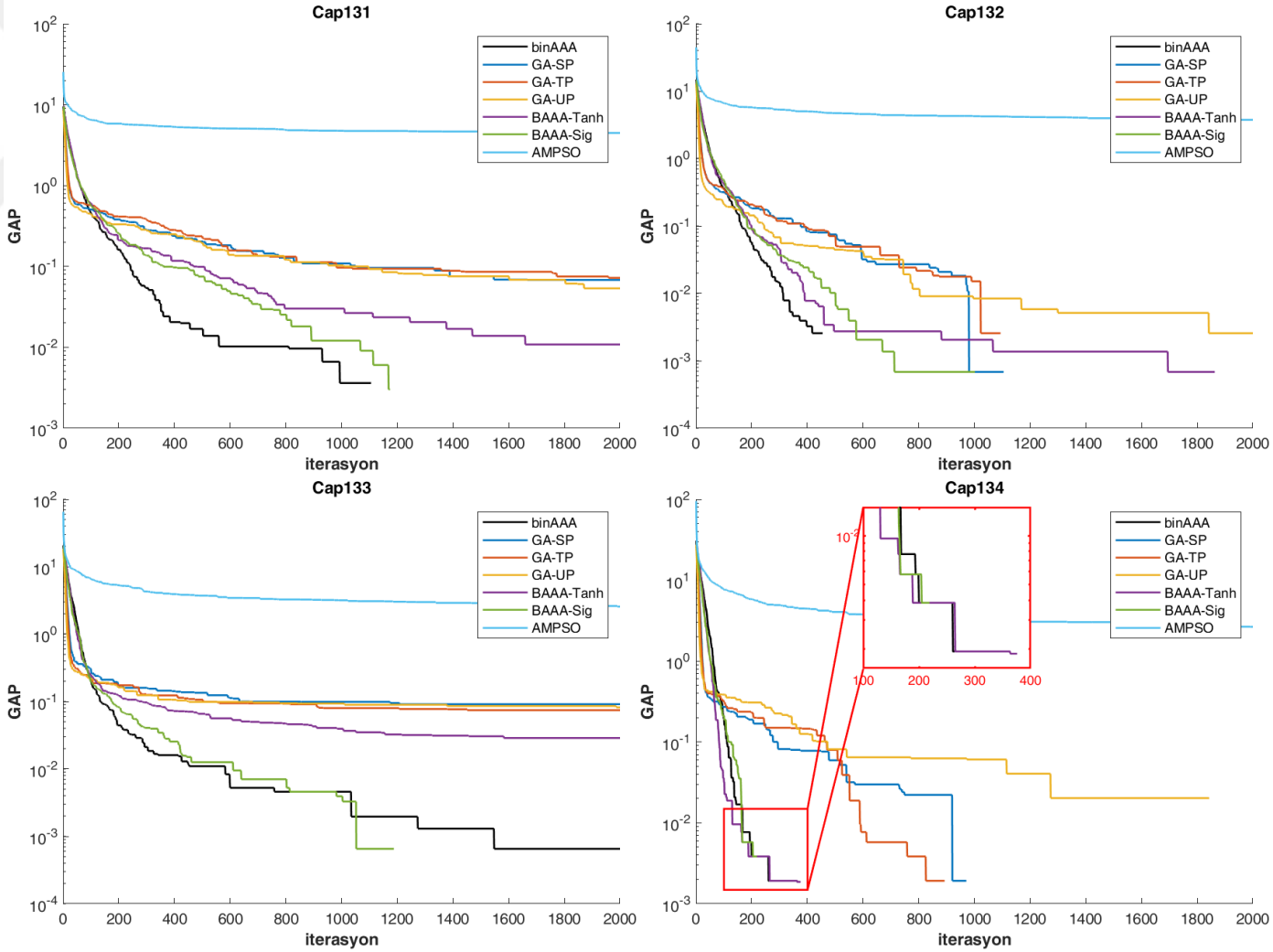
Karşılaştırılan algoritmaların çözüme ulaşma hızlarının daha iyi değerlendirilebilmesi için her bir problem için üretilen yakınsama eğrileri Şekil 4.1-4.4'de sunulmuştur.



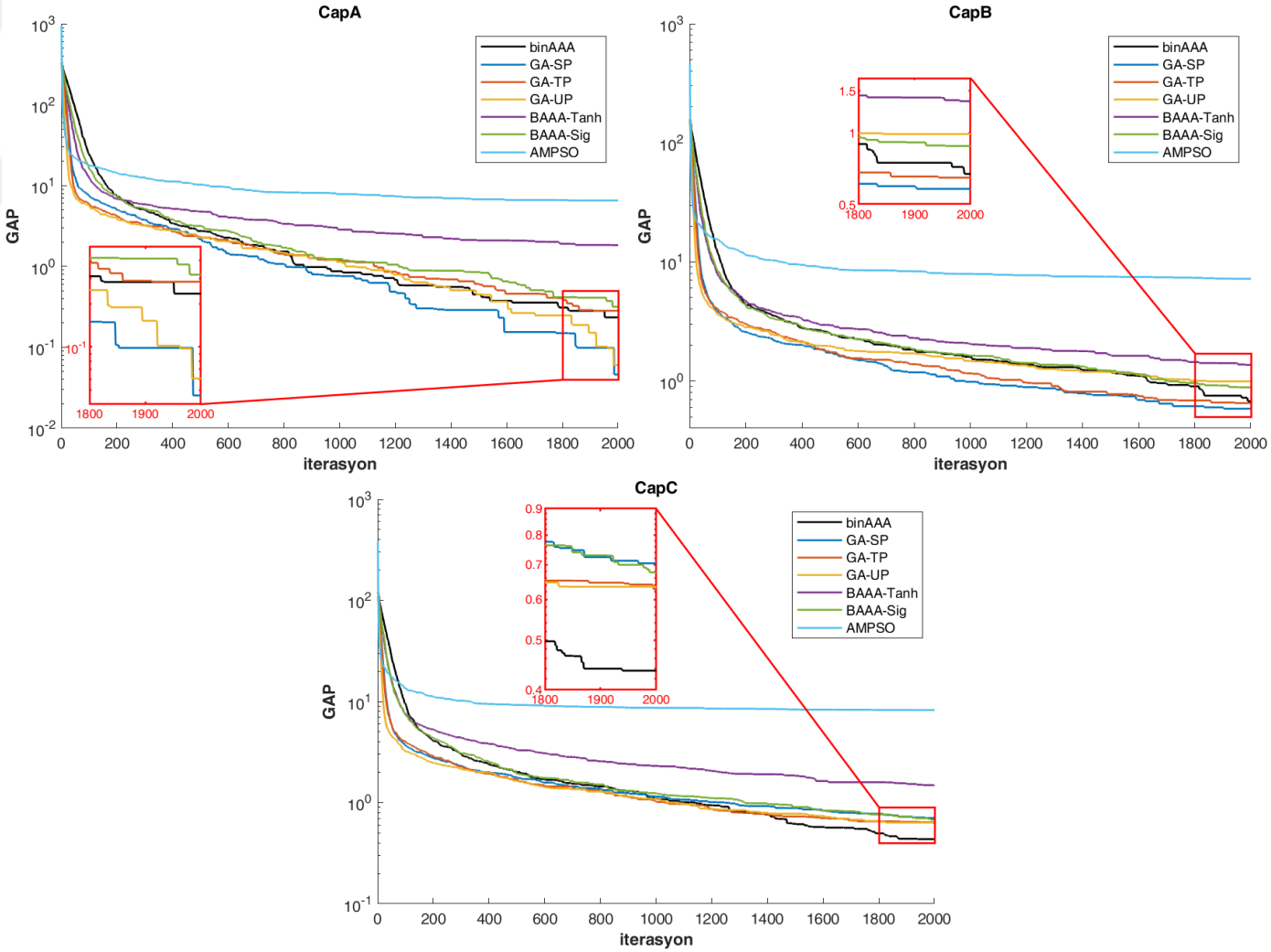
Şekil 4.1. binAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, az boyutlu Cap71, Cap72, Cap73 ve Cap74 problemleri için ürettikleri yakınsama eğrileri



Şekil 4.2. binAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, orta boyutlu Cap101, Cap102, Cap103 ve Cap104 problemleri için ürettikleri yakınsama eğrileri



Şekil 4.3. binAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, büyük boyutlu Cap131, Cap132, Cap133 ve Cap134 problemleri için ürettikleri yakınsama eğrileri



Şekil 4.4. binAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, çok büyük boyutlu CapA, CapB ve CapC problemleri için ürettikleri yakınsama eğrileri

binAAA ve diğer algoritmaların 15 farklı test problemleri için ürettikleri yakınsama grafikleri incelendiğinde, önerilen algoritmanın *çok büyük boyutlu* problemler haricinde optimum çözümlere hızlı yakınsadığı görülmektedir. *Çok büyük boyutlu* problemler için ise binAAA, optimuma yakın çözümler için kabul edilebilir bir yakınsama karakteristiği göstermektedir.

binAAA yönteminin performansının daha iyi anlaşılması için literatürden toplanan sonuçlar (Kiran ve Gunduz, 2013; Kiran, 2015a) ile ikinci bir karşılaştırma daha gerçekleştirilmiştir. IBPSO (Yuan ve ark., 2009), BPSO (Kennedy ve Eberhart, 1997), DisABC (Kashan ve ark., 2012), binABC (Kiran ve Gunduz, 2013) ve ABCbin (Kiran, 2015a) algoritmalarının sonuçları (Kiran ve Gunduz, 2013; Kiran, 2015a) makalelerden direkt olarak alınarak, binAAA algoritması ile yapılan kıyaslama Çizelge 4.5'de verilmiştir.

Çizelge 4.5. Sonuçları literatürden derlenen ikili olarak yapılandırılmış algoritmaların, binAAA yöntemi ile UFLP seti üzerinde kıyaslaması

Problem	BPSO		IBPSO		binABC		DisABC		ABCbin		binAAA	
	Gap	Std.Sap.	Gap	Std.Sap.	Gap	Std.Sap.	Gap	Std.Sap.	Gap	Std.Sap.	Gap	Std.Sap.
Cap71	**0.0000**	**0.00**	0.0374	587.49	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
Cap72	**0.0000**	**0.00**	0.2749	1844.64	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
Cap73	0.0242	634.62	0.1980	1513.78	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
Cap74	0.0088	500.27	0.4031	4426.67	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
Cap101	0.0462	566.44	0.5968	3799.52	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
Cap102	0.0148	386.76	0.7317	3249.38	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
Cap103	0.0422	485.26	0.6410	4978.98	**0.0000**	**0.00**	**0.0000**	**0.00**	0.0051	85.67	**0.0000**	**0.00**
Cap104	0.0810	1951.81	0.9964	10845.26	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
Cap131	0.1317	1207.63	2.4236	4244.29	**0.0000**	**0.00**	0.6196	2337.64	0.1967	1065.73	**0.0000**	**0.00**
Cap132	0.0914	1196.19	3.6014	11569.02	**0.0000**	**0.00**	0.0945	813.37	0.0199	213.28	**0.0000**	**0.00**
Cap133	0.1115	821.28	5.2626	14905.27	0.1215	200.24	0.0309	359.03	0.0747	561.34	**0.0007**	**31.91**
Cap134	0.1346	2285.42	7.6338	15788.86	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
CapA	2.1785	374302.81	37.8862	3357138.19	2.9622	236833.50	**0.1522**	**74782.61**	3.1723	268685.20	0.2345	78259.57
CapB	1.9490	176206.07	55.2701	1406575.70	2.5081	91430.13	3.3027	109738.50	2.8154	88452.80	**0.6747**	**57572.96**
CapC	1.4870	92977.85	45.5561	1245252.20	2.5800	82312.70	4.6968	95778.78	2.0374	78162.20	**0.4360**	**43963.88**

Çizelge 4.5'deki sonuçlar incelendiğinde binAAA yönteminin, bütün fonksiyonlar için IBPSO yönteminden daha iyi sonuçlar ürettiği görülmektedir. binAAA ve BPSO algoritması Cap71 ve Cap72 problemlerinde aynı değerlere sahip olup, diğer bütün problemlerde binAAA daha iyi performans sergilemiştir. binAAA yöntemini binABC ile kıyasladığımızda, binAAA Cap133, CapA, CapB ve CapC problemlerinde binABC den daha iyi değerler üretmiş, diğer bütün problemlerde ise 30 çalıştırmanın tümünde optimum sonuca erişmişlerdir. DisABC yöntemi CapA probleminde bütün algoritmalar içinde en iyi sonuca sahip olup, binAAA yöntemi Cap131, Cap132, Cap133, CapB ve CapC problemlerinin çözümünde DisABC den daha iyidir. ABCbin yöntemi ise 8 problemde binAAA ile aynı sonucu üretmişlerdir fakat kalan 7 problemde binAAA daha iyi değerlere sahiptir.

Genel olarak bakıldığında, binAAA ve diğer 11 farklı yöntem ile 15 farklı UFLP problemleri üzerinde yapılan kıyaslamalar sonucunda, binAAA yönteminin rekabetçi ve alternatif bir ikili optimizasyon algoritması olduğu görülmektedir.

4.2. Lojik Xor ve Stigmerjik Davranış Kullanılarak Geliştirilen İkili AAA (SAAA)

4.2.1. Önerilen Yöntem

Bu tez kapsamında önerilen ikinci yaklaşımda, ikili optimizasyon problemlerinin çözümü için geliştirilen Stigmerjik AAA (Stigmergic AAA- SAAA) yöntemi iki farklı güncelleme mekanizması barındırmaktadır (Korkmaz ve Kiran, 2018). Bu mekanizmanın ilki, aday çözüm üretmek için lojik tabanlı *XOR* operatörü kullanmakta, ikinci mekanizmada ise ilk mekanizmadan edinilen bilgi kullanılarak stigmerjik davranış temelinde yeni çözümler üretilmektedir. Hangi güncelleme mekanizmasının işletileceğini belirlemek için kullanılan denklem aşağıdaki gibidir:

$$Güncelleme\ mekanizması = \begin{cases} UM_1\ , if\ (r < UMSP\ )\ ve\ C_{01}(t) \neq 0\ ve\ C_{10}(t) \neq 0 \\ UM_2\ , aksi\ takdirde \end{cases} \quad (4.3)$$

Denklemde, UM_1 Güncelleme Mekanizması-1'i (Update Mechanism-1), UM_2 Güncelleme Mekanizması-2'yi (Update Mechanism-2), UMSP algoritmaya özgü kontrol parametresi olan Güncelleme Mekanizması Seçme Olasılığını (Update Mechanism Selection Probability), r [0,1] aralığında üretilen rastgele sayıyı ifade etmektedir. Denklemdeki C_{01} ve C_{10} tamsayı değişkenleri olup sonraki bölümlerde detaylı anlatılmıştır.

4.2.1.1 Güncelleme Mekanizması-1 (UM_1): Lojik XOR Tabanlı İkili AAA

AAA algoritmasının önerilen ilk versiyonu, sürekli değerler alan karar değişkenleri barındıran problemleri çözmek için önerilmiştir. İkili optimizasyon problemlerinde karar değişkenleri 0 yada 1 değerlerini aldığından dolayı, bu problemleri çözmek için AAA algoritmasının ikili optimizasyon problemlerini çözecek şekilde uyarlanması gerekmektedir. Bu amaçla önerilen ikinci yöntemde AAA algoritmasının ana bileşenlerinden olan *başlangıç aşaması*, *çözüm güncelleme kuralları (helisel hareket)* ve *adaptasyon fazı* olmak üzere 3 bileşen yeniden uyarlanmıştır. AAA algoritmasının bir diğer fazı olan *Evrimsel süreç*, ikili değerler ile çalışmak için uygun olduğundan hiçbir değişiklik yapılmadan orijinal hali ile kullanılmıştır. XOR tabanlı İkilileştirme işlemi, bazı değişiklikler yapılarak ABC (Kiran ve Gunduz, 2013) algoritmasında kullanıldığı

gibi AAA algoritmasına da uyarlanmıştır. B*aşlangıç Aşamasında* temel AAA'nın kullandığı Denklem 3.1 üzerinde modifikasyon yapılarak Denklem 4.4 üretilmiştir.

$$X_{i,j} = \begin{cases} 0 \ , if \ (r_{i,j} < a) \\ 1 \ , aksi \ takdirde \end{cases} , i = 1,2, \dots, N \ ve \ j = 1,2, \dots, D \quad (4.4)$$

Burada, a [0,1] aralığında seçilebilen olasılık değeri olup, karar değişkenlerinin rastgele dağıtılabilmesi için 0.5 olarak belirlenmiştir. $r_{i,j}$ [0,1] aralığında üretilen rastgele sayıyı, $X_{i,j}$ popülasyondaki i'nci bireyin (alg kolonisi) j'inci hücresini (boyut) ifade etmektedir.

Alg kolonileri (popülasyon) ilklendirildikten sonra temel AAA'daki helisel harekete ait olan Denklem 3.6, Denklem 3.7 ve Denklem 3.8, ABC algoritmasında (Kiran ve Gunduz, 2013) yapıldığı gibi değiştirilmiş ve aşağıda verilmiştir:

$$V_m = X_{i,m} \oplus [\ \varphi\ (\ X_{i,m} \oplus X_{kt,m}\ )\ ] \quad (4.5)$$

$$V_k = X_{i,k} \oplus [\ \varphi\ (\ X_{i,k} \oplus X_{kt,k}\ )\ ] \quad (4.6)$$

$$V_l = X_{i,l} \oplus [\ \varphi\ (\ X_{i,l} \oplus X_{kt,l}\ )\ ] \quad (4.7)$$

$$i, kt \in \{1,2, \dots, N\}\ ,\ i \neq kt\ ,\quad m, k, l \in \{1,2, \dots, D\}\ ,\ m \neq k \neq l$$

Denklemde, V aday çözümü, X popülasyondaki bir bireyi, $\oplus$ lojik *XOR* operatörünü, φ lojik *DEĞİL* operatörünü (%50 olasılık ile), N popülasyonun büyüklüğünü, D problemin boyutunu, kt turnuva seçim yöntemiyle popülasyondan rastgele seçilmiş komşu çözüm indisini, m, k ve l ise rastgele seçilmiş karar değişkenleri indisini ifade etmektedir. Dolayısıyla V_m aday çözümün m'inci boyutunu, V_k k'inci boyutunu ve V_l l'inci boyutunu belirtmektedir. Böylece, çözüm güncelleme mekanizmasında, AAA'nın temel versiyonunda olduğu gibi, her bir güncelleme işleminde 3 boyut güncellenmiş olmaktadır.

Helisel hareket fazından sonra, aday çözümün uygunluk değeri hesaplanmakta ve asıl bireyin uygunluk değeri ile karşılaştırılmaktadır. Eğer aday çözümün uygunluk değeri asıl bireyin uygunluk değerinden daha iyi ise, aday çözüm asıl bireyin yerine kopyalanmakta ve ikinci güncelleme mekanizması için gerekli olan bilgi bu aşamada elde edilmektedir. Eski ve yeni çözümün karar değişkenleri tek tek karşılaştırılmakta ve değişime uğramış olan karar değişkenleri belirlenmektedir. Eğer karar değişkeninin eski değeri 1 ve yeni değeri 0 ise bu durumda C_{10} sayacının değeri bir artırılmaktadır. Buna

karşılık, eğer karar değişkeninin eski değeri 0 ve yeni değeri 1 ise bu durumda C_{01} sayacının değeri bir artırılmaktadır. Bir başka ifadeyle, C_{10} sayacı, aday çözümün asıl çözümün yerini aldığı durumlarda, karar değişkenlerinin önceki değeri 1 iken 0 olarak değişenlerinin sayısını tutmaktadır. Buna karşılık, C_{01} sayacı ise 0 değerinden 1 değerine değişim sayısını barındırmaktadır. C_{01} ve C_{10} sayaçları sırasıyla Denklem 4.8 ve Denklem 4.9'daki gibi hesaplanmaktadır:

$$C_{01}(t+1) = \begin{cases} C_{01}(t)+1 \,, Obj(V) < Obj(X_i)\ ve\ X_{i,p}=0\ ve\ V_p=1 \\ C_{01}(t) \quad\quad , aksi\ takdirde \end{cases} , \forall p \in P \qquad (4.8)$$

$$C_{10}(t+1) = \begin{cases} C_{10}(t)+1 \,, Obj(V) < Obj(X_i)\ ve\ X_{i,p}=1\ ve\ V_p=0 \\ C_{10}(t) \quad\quad , aksi\ takdirde \end{cases} , \forall p \in P \qquad (4.9)$$

$$P = \{ m, k, l \} \ , \ m, k, l \in \{1,2, \dots, D\} \ , \ m \neq k \neq l$$

Burada, V aday çözümü, X popülasyondaki bireyi, Obj probleme özgü amaç fonksiyon değerini belirtmektedir. Ayrıca, $Obj(V) < Obj(X_i)$ ifadesi minimizasyon problemleri için geçerlidir. Eğer problem maksimizasyon problemi ise işaretin yönünün tersine çevrilmesi gerekmektedir. ($Obj(V) > Obj(X_i)$)

Yöntemin ikili değerler ile çalışabilmesi için yapılan bir diğer değişiklik ise adaptasyon fazında yapılmıştır. AAA'nın temel versiyonunda, [0,1] aralığında üretilen rasgele sayı, adaptasyon kontrol parametresinden daha az ise, adaptasyon fazı işletilmekte ve en aç koloni, alg kolonileri içerisindeki (popülasyondaki) en büyük koloniye Denklem 3.11 kullanılarak benzetilmektedir. Önerilen yaklaşımda ise, adaptasyon kontrol parametresi, en büyük ve en aç kolonilerin tüm hücreleri (boyutları) için kullanılmaktadır. Denklemin yeniden uyarlanmış hali Denklem 4.10'da verilmiştir.

$$X_{st,j}(t+1) = \begin{cases} X_{b,j}\ (t) \ , if\ (r_j < Ap\ ) \\ X_{st,j}\ (t) \ , aksi\ takdirde \end{cases} \quad \forall j \in D \ , \ j = 1,2, \dots, D \qquad (4.10)$$

Denklemde, $X_{st,j}$ en aç alg kolonisinin j'inci boyutunu, $X_{b,j}$ en büyük alg kolonisinin j'inci boyutunu, r_j j'inci boyut için [0,1] aralığında üretilen rastgele sayıyı ve Ap ise Adaptasyon Kontrol parametresini ifade etmektedir. Yöntemin daha iyi anlaşılması adına Güncelleme Mekanizması-1'in sözde kodu Şekil 4.5'de verilmiştir.

```
1) kt, turnuva seçim yöntemiyle popülasyondan rastgele seçilmiş komşu çözüm indisini,
2) P, [1, Boyut] arasında ve birbirinden farklı rasgele seçilmiş 3 adet boyutun indislerini,
3) V, aday çözümü,
4) φ, 0.5 değerine sahip değilini alma (lojik NOT) oranını ifade etmektedir.
5) V=X(i)
6) For p=1 to 3
7)    If rand< φ then
8)       V( P(p) ) = XOR ( X(i, P(p) ) , XOR ( X(i, P(p) ), X(kt, P(p) )  ) )
9)    else
10)      V( P(p) ) = XOR ( X(i, P(p) ) , ~XOR ( X(i, P(p) ), X(kt, P(p) )  ) )
11)   end if
12) end for
13) If Obj(V)<Obj( X(i) ) then
14)   For p=1 to 3
15)      If X(i, P(p) )==0 and V( P(p) )==1 then C01=C01+1 end if
16)      If X(i, P(p) )==1 and V( P(p) )==0 then C10=C10+1 end if
17)   end for
18)   X(i)=V
19) end if
```

Şekil 4.5. Güncelleme Mekanizması-1'in sözde kodu

4.2.1.2 Güncelleme Mekanizması-2 (UM_2): Stigmerjik Davranış

Stigmerji (stigmergy) kavramı ilk olarak Fransız zoolog Pierre-Paul Grassé tarafından 1959 yılında Entomoloji (böcekbilim) alanında tanıtılmıştır (Grassé, 1959). Bu çalışmanın amaçlarından birisi, basit termit bireylerinin büyük termit tümseklerini (höyük) nasıl oluşturabildiğini bulmaktır. Grassé'ye göre, termitler yuvalarını inşa ederken çok basit kurallara uyma eğilimi göstermektedirler (Crina ve Ajith, 2006). Termitlerin eylemleri, herhangi bir amaçlı plan ile baştan sona koordineli değildir. Bunun aksine, çevrenin anlık durumuna bağlı olarak basit davranış sergilerler (Crina ve Ajith, 2006). Bu anlık davranış Grassé tarafından stigmerji olarak adlandırılmaktadır (Grassé, 1959). Bu davranış sadece termitlerde değil, aynı zamanda karıncalar, arılar ve diğer birçoklarında da gözlenmektedir. Bu bağlamda, AAA algoritması için, stigmerjik davranışa sahip yeni bir ikili yaklaşım önerilmiştir. Bu yaklaşımda, geçmişte gerçekleştirilen eylemlere (C_{01} ve C_{10} sayaçları) ait bilgiler, gelecekte gerçekleştirilecek hareketlere rehberlik etmek için kullanılmaktadır. Diğer bir deyişle, problem üzerinde işlenmiş çözümlerden elde edilen bilgiler, alg kolonilerinin davranışlarına yol göstermektedir. Güncelleme Mekanizması-1'de (UM_1) C_{01} ve C_{10} sayaçları güncellenmekte, Güncelleme Mekanizması-2'de (UM_2) ise aday çözümlerin

üretilmesinde bu sayaçlar kullanılmaktadır. Aday çözümlerin üretilmesinde kullanılan olasılık oranları Denklem 4.11 ve Denklem 4.12'daki gibi hesaplanmaktadır.

$$p_{01}(t+1) = \frac{C_{01}(t)}{C_{01}(t)+C_{10}(t)} \tag{4.11}$$

$$p_{10}(t+1) = \frac{C_{10}(t)}{C_{01}(t)+C_{10}(t)} \tag{4.12}$$

Denklemlerde, $p_{01}(t+1)$ C_{01} sayacının *t+1* anındaki olasılık oranını ve $p_{10}(t+1)$ ise C_{10} sayacının *t+1* anındaki olasılık oranını ifade etmektedir. Bu olasılıklar daha sonra aday çözümlerin üretilmesinde kullanılmaktadır. AAA algoritmasının temel versiyonunda, helisel hareket aşamasında, yeni aday çözümler üretilirken rastgele seçilmiş üç farklı boyut güncellenmektedir. Buna paralel olarak, önerilen yöntemde de, aday çözümlerin üretilmesi için en çok üç boyut güncellenmektedir. Bu amaçla, yönteme Boyut Seçilme Olasılığı (Dimension Selection Probability-DSP) isimli yeni bir kontrol parametresi eklenmiştir. Belirlenen üç adet boyutun her biri için [0,1] aralığında üretilen rasgele sayı, eğer DSP parametresinden küçük ise, bu boyutta hareket gerçekleştirilmektedir.

Aday çözüm *V*'nin üretilmesi için Denklem 4.13 ve Denklem 4.14 kullanılmaktadır.

$$V_{A(a)} = \begin{cases} 0 & , \ r_1 < p_{10} \\ V_{A(a)} & , \ aksi\ takdirde \end{cases} \tag{4.13}$$

$$V_{B(b)} = \begin{cases} 1 & , \ r_1 \geq p_{10} \\ V_{B(b)} & , \ aksi\ takdirde \end{cases} \tag{4.14}$$

Burada, *A* aday çözümdeki *1* değerine sahip olan boyutların indekslerini, *B* ise aday çözümdeki *0* değerine sahip olan boyutların indekslerini barındırmaktadır. *a*, 1 ile *A*'nın boyut sayısı arasında belirlenmiş rastgele bir tamsayıdır. *b*, 1 ile *B*'nin boyut sayısı arasında belirlenmiş rastgele bir tamsayıdır. Dolayısıyla $V_{A(a)}$, *V* aday çözüm içerisindeki, *1* değerine sahip rastgele seçilmiş bir boyutu(hücreyi) ifade etmektedir. $V_{B(b)}$ ise aday çözüm içerisindeki, *0* değerine sahip rastgele seçilmiş bir boyutu göstermektedir. Denklemdeki , r_1 değeri [0,1] aralığında üretilmiş rastgele bir sayıdır. Eğer r_1 sayısı p_{10} olasılık değerinden düşük ise, aday çözümdeki *1* değerine sahip olan bir karar değişkeni ($V_{A(a)}$)) *0* olarak değiştirilmektedir. Bunun aksine, eğer üretilen r_1 sayısı, p_{10} olasılık

değerine eşit veya büyük ise, aday çözümdeki *0* değerine sahip olan bir karar değişkeni ($V_{B(b)}$)) *1* olarak değiştirilmektedir.

Sonuç olarak, minimizasyon problemleri için, aday çözümün uygunluk değeri, popülasyondaki aday çözüm üretmek için seçilmiş bireyin uygunluk değerinden daha az ise, aday çözüm bu bireye kopyalanır. Bu işlem için Denklem 4.15 kullanılmaktadır.

$$X_i(t+1) = \begin{cases} V(t) \ , \ Obj(\ V(t)\ ) < \ Obj(\ X_i(t)\ ) \\ X_i(t) \ , \ aksi\ takdirde \end{cases} \quad (4.15)$$

Denklemde, $X_i(t)$, t anındaki i'nci bireyi, $V(t)$ t anındaki aday çözümü, $Obj(\ X_i(t)\ )$ t anındaki i'nci bireyin uygunluk değerini, $Obj(\ V(t)\ )$ t anındaki aday çözümün uygunluk değerini, $X_i(t+1)$ ise uygunluk değerlerine göre seçim yapıldıktan sonra, popülasyonun $t+1$ anındaki i'nci bireyini ifade etmektedir. Güncelleme Mekanizması-2'nin sözde kodu Şekil 4.6'da verilmiştir.

```
1)  V, aday çözümü ifade etmektedir
2)  For d=1 to 3
3)    If rand < DSP then
4)      If rand < p10 then
5)        A=find(V==1)
6)        a=rand(size of A)
7)        V( A(a) )=0
8)      else if
9)        B=find(V==0)
10)       b=rand(size of B)
11)       V( B(b) )=1
12)     end if
13)   end if
14) end for
15) If Obj(V)<Obj( X(i) ) then
16)   X(i)=V
17) end if
```

Şekil 4.6. Güncelleme Mekanizması-2'nin sözde kodu

4.2.2. Deneysel Sonuçlar ve Karşılaştırmalar

Önerilen yöntemin performansının daha iyi değerlendirilebilmesi için karşılaştırmalar iki farklı deney seti üzerinden gerçekleştirilmiştir. İlk karşılaştırma için OR-Library (Beasley, 1990a) 'de yayınlanmış, 15 farklı probleme sahip olan UFLP test seti kullanılmış ve SAAA yöntemi, BAAA yönteminin 2 farklı versiyonu, GA yöntemin

3 farklı versiyonu ve BPSO yöntemi ile kıyaslanmıştır. Önerilen yöntem ve karşılaştırılan diğer yöntemler, Mathworks şirketinin ticari bir ürünü olan MATLAB ortamında kodlanmıştır. İkinci karşılaştırma için ise CEC2015 (bound constrained single-objective computationally expensive numerical optimization problems) (Chen ve ark., 2014) test seti kullanılmış ve önerilen yöntemin performansının değerlendirilebilmesi için SBHS, BLDE, İkili Hibrit Topolojili PSO Algoritması (Binary Hybrid Topology PSO Algorithm- BHTPSO-QI), Genetik Operatör Tabanlı Yapay Arı Kolonisi Algoritması (Genetic Operators Based Artificial Bee Colony Algorithm-GBABC), BQIGSA ve SabDE yöntemleri ile kıyaslanmıştır. SAAA yöntemi Mathworks şirketinin ticari bir ürünü olan MATLAB ortamında kodlanmış, karşılaştırılan diğer algoritmaların sonuçları ise doğrudan Banitalebi ve ark. (2016) tarafından yapılan çalışmadan (SabDE) alınmıştır. Ayrıca bütün algoritmalar için çalıştırma ortamı olarak, Intel® Core™ i7-3820 3.60GHz işlemcili, 16GB belleğe sahip ve Windows 10 Pro 64-bit işletim sistemi ile çalışan bir kişisel bilgisayar kullanılmıştır.

4.2.2.1 UFLP Test Seti Kullanılarak Yapılan Karşılaştırmalar

Algoritmaların adil bir şekilde kıyaslanabilmesi için, yöntemlerin ortak kontrol parametreleri birbirine eşit olarak seçilmiştir. Bu bağlamda, popülasyon büyüklüğü 40 ve sonlandırma koşulu olarak MaxFEs değeri 80,000 olarak seçilmiştir. MaxFEs değerine paralel olarak, gerekli yöntemler için sonlandırma koşulu olarak, maksimum iterasyon sayısı değeri 2,000 olarak kullanılmıştır. GA'nın 3 farklı versiyonu (tek noktadan çaprazlama, iki noktadan çaprazlama ve çok noktadan çaprazlama) için, Holland (1992a) tarafından yapılan çalışmada olduğu üzere, çaprazlama oranı 0.8 ve mutasyon oranı 0.01 olarak seçilmiştir. BAAA'nın iki versiyonu için, Zhang ve ark. (2016) tarafından yapılan çalışmada olduğu üzere, kesme kuvveti kontrol parametresi 2, enerji kaybı kontrol parametresi 0.3 ve adaptasyon oranı kontrol parametresi 0.5 olarak kullanılmıştır. Yine aynı çalışmada (Zhang ve ark., 2016) kullanıldığı gibi, eğrinin değişim eğilimi (changing trend of the curve) (T) kontrol parametresi, *Tanh(x)* lojistik fonksiyonu için 1.5 ve *Sig(x)* lojistik fonksiyonu için 2 olarak kullanılmıştır. BPSO algoritması için, parçacık hızının üst limiti (V_{max}) 6, parçacık hızının alt limiti (V_{min}) -6 ve pozitif ivme sabitleri (c_1, c_2) 2 olarak seçilmiştir. SAAA yöntemi için ise, AAA'nın temel versiyonunda (Uymaz ve ark., 2015b) olduğu üzere, enerji kaybı kontrol parametresi 0.3 ve adaptasyon oranı kontrol parametresi 0.5 olarak kullanılmıştır. AAA'nın temel versiyonunda kullanılan kesme

kuvveti kontrol parametresi önerilen yöntemde kullanılmamıştır. Temel versiyonda, karar değişkenleri sürekli değerler almaktadır ve kesme kuvveti kontrol parametresi, yeni aday çözümler üretmek için kullanılmaktadır. Önerilen yöntemde karar değişkenleri ikili değerler aldığından dolayı, yeni aday çözümler üretmek için kesme kuvveti kontrol parametresine ihtiyaç duyulmamıştır. SAAA yöntemi için önerilen, yönteme özgü kontrol parametrelerinden, UMSP kontrol parametresinin değeri 0.5 ve DSP kontrol parametresinin değeri 0.66 olarak kullanılmıştır.

Her bir problem için tüm algoritmalar 30 kez bağımsız olarak çalıştırılmış ve elde edilen sonuçlar ortalama değer, standart sapma, optimum değeri bulma sayısı, ortalama çalışma süresi(saniye), işaret ve Gap değeri olarak Çizelge 4.6, Çizelge 4.7, Çizelge 4.8 ve Çizelge 4.9'da sunulmuştur. Çizelgelerdeki *optimum değeri bulma sayısı* satırları, 30 farklı çalıştırmadan kaç tane çalıştırmada optimum sonuca erişildiğini ifade etmektedir. Çizelgelerdeki *işaret* satırları, SAAA yönteminin sonuçları ile karşılaştırılan diğer yöntemlerin sonuçları arasında istatistiksel olarak anlamlı bir farkın olup olmadığının belirlenmesi adına 0.05 güven aralığında gerçekleştirilen Wilcoxon işaretli sıralama testinin (Wilcoxon, 1945) sonuçlarıdır. Çizelgelerde bulunan *işaret* satırlarındaki artı (+) işareti, SAAA yöntemi ile ilgili yöntemin sonuçlarının istatistiksel olarak birbirlerinden farklı olduğunu göstermektedir. Çizelgelerde bulunan *işaret* satırlarındaki eksi (-) işareti ise SAAA yöntemi ile ilgili yöntemin sonuçlarının istatistiksel olarak birbirlerinden farklı olmadığını göstermektedir. Çizelgelerdeki *Gap* satırları, yöntemin 30 defa çalıştırma sonucunda elde ettiği ortalama değer ile problemin optimum değeri arasındaki oransal fazlalığı ifade etmekte ve Denklem 4.2 kullanılarak hesaplanmaktadır.

Yapılan karşılaştırmanın daha anlaşılır ve görünür hale gelmesi için, her bir problem için en iyi algoritmanın değerleri kalın yazı tipinde verilmiştir. Sonuçların daha iyi incelenebilmesi adına, problemler boyutlarına göre *az boyutlu, orta boyutlu, büyük boyutlu ve çok büyük boyutlu* olmak üzere dört farklı grupta incelenmiş ve sırasıyla Çizelge 4.6, Çizelge 4.7, Çizelge 4.8 ve Çizelge 4.9'da gösterilmiştir.

Çizelge 4.6. SAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, az boyutlu Cap71, Cap72, Cap73 ve Cap74 problemleri üzerinde karşılaştırılması

Yöntem		Cap71	Cap72	Cap73	Cap74
GA-SP	Ortalama	**932615.750**	**977799.400**	1011314.476	**1034976.975**
	Gap	**0.00000**	**0.00000**	0.06659	**0.00000**
	Opt.Say.	**30**	**30**	19	**30**
	Std.Sap.	**0.000**	**0.000**	899.650	**0.000**
	İşaret	**-**	**-**	+	**-**
	Ort.Süre	**26.957**	**27.893**	27.994	**27.998**
GA-TP	Ortalama	**932615.750**	**977799.400**	1011130.923	**1034976.975**
	Gap	**0.00000**	**0.00000**	0.04843	**0.00000**
	Opt.Say.	**30**	**30**	22	**30**
	Std.Sap.	**0.000**	**0.000**	825.576	**0.000**
	İşaret	**-**	**-**	+	**-**
	Ort.Süre	**27.568**	**28.056**	28.050	**28.143**
GA-UP	Ortalama	**932615.750**	**977799.400**	1011069.739	**1034976.975**
	Gap	**0.00000**	**0.00000**	0.04238	**0.00000**
	Opt.Say.	**30**	**30**	23	**30**
	Std.Sap.	**0.000**	**0.000**	789.612	**0.000**
	İşaret	-	-	+	-
	Ort.Süre	**27.744**	**28.103**	28.101	**28.095**
BAAA-Tanh	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	İşaret	**-**	**-**	**-**	**-**
	Ort.Süre	**25.748**	**26.279**	**28.221**	**27.584**
BAAA-Sig	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	İşaret	**-**	**-**	**-**	**-**
	Ort.Süre	**25.868**	**26.927**	**28.219**	**27.461**
BPSO	Ortalama	**932615.750**	**977799.400**	1010886.187	1035068.312
	Gap	**0.00000**	**0.00000**	0.02422	0.00882
	Opt.Say.	**30**	**30**	26	29
	Std.Sap.	**0.000**	**0.000**	634.625	500.272
	İşaret	**-**	**-**	**-**	**-**
	Ort.Süre	**34.935**	**34.765**	34.950	34.822
SAAA	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	Ort.Süre	**27.901**	**28.078**	**27.528**	**27.239**

Çizelge 4.7. SAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, orta boyutlu Cap101, Cap102, Cap103 ve Cap104 problemleri üzerinde karşılaştırılması

Yöntem		Cap101	Cap102	Cap103	Cap104
GA-SP	Ortalama	797193.286	**854704.200**	894351.782	**928941.750**
	Gap	0.06839	**0.00000**	0.06374	**0.00000**
	Opt.Say.	11	**30**	6	**30**
	Std.Sap.	421.655	**0.000**	505.036	**0.000**
	İşaret	+	**-**	+	**-**
	Ort.Süre	29.372	**28.730**	28.689	**32.706**
GA-TP	Ortalama	797164.610	**854704.200**	894329.179	**928941.750**
	Gap	0.06479	**0.00000**	0.06121	**0.00000**
	Opt.Say.	12	**30**	10	**30**
	Std.Sap.	428.658	**0.000**	540.160	**0.000**
	İşaret	+	**-**	+	**-**
	Ort.Süre	29.206	**28.931**	28.907	**32.992**
GA-UP	Ortalama	797107.258	**854704.200**	894427.382	**928941.750**
	Gap	0.05759	**0.00000**	0.07220	**0.00000**
	Opt.Say.	14	**30**	9	**30**
	Std.Sap.	436.524	**0.000**	522.784	**0.000**
	İşaret	+	**-**	+	**-**
	Ort.Süre	29.169	**28.870**	28.915	**33.046**
BAAA-Tanh	Ortalama	796677.114	**854704.200**	**893782.113**	**928941.750**
	Gap	0.00360	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	29	**30**	**30**	**30**
	Std.Sap.	157.066	**0.000**	**0.000**	**0.000**
	İşaret	**-**	**-**	**-**	**-**
	Ort.Süre	27.473	**26.334**	**25.851**	**25.435**
BAAA-Sig	Ortalama	**796648.438**	**854704.200**	**893782.113**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	İşaret	**-**	**-**	**-**	**-**
	Ort.Süre	**26.836**	**26.215**	**25.926**	**26.963**
BPSO	Ortalama	796992.553	854788.703	894223.572	929318.098
	Gap	0.04320	0.00989	0.04939	0.04051
	Opt.Say.	18	28	14	28
	Std.Sap.	428.658	321.588	521.237	1432.239
	İşaret	+	-	+	-
	Ort.Süre	41.814	41.554	41.375	41.177
SAAA	Ortalama	**796648.438**	**854704.200**	**893782.113**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	Ort.Süre	**28.177**	**27.923**	**27.838**	**27.592**

Çizelge 4.8. SAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, büyük boyutlu Cap131, Cap132, Cap133 ve Cap134 problemleri üzerinde karşılaştırılması

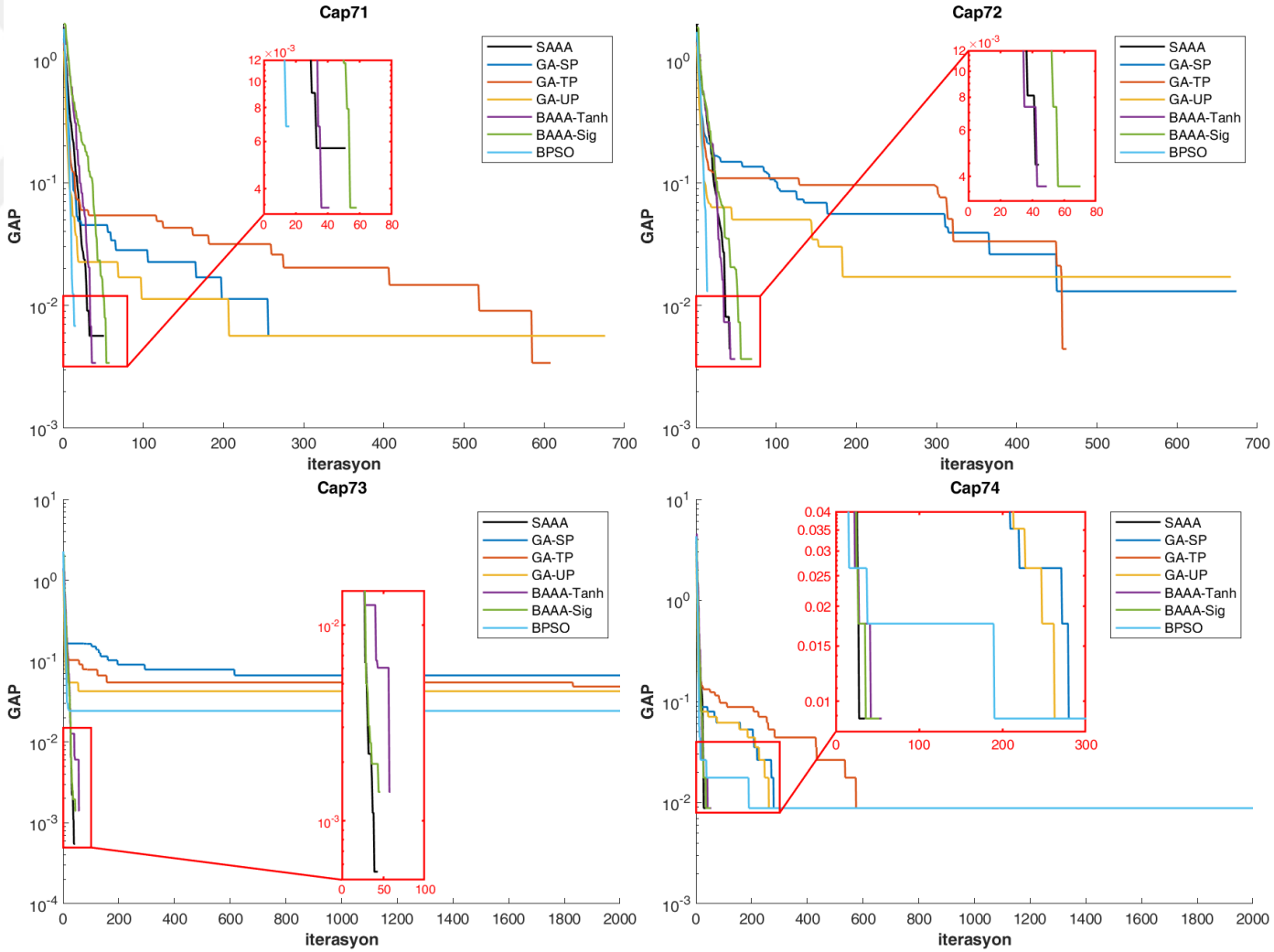
Yöntem		Cap131	Cap132	Cap133	Cap134
GA-SP	Ortalama	793980.104	**851495.325**	893891.911	**928941.750**
	Gap	0.06813	**0.00000**	0.09128	**0.00000**
	Opt.Say.	16	**30**	10	**30**
	Std.Sap.	720.877	**0.000**	685.076	**0.000**
	İşaret	+	**-**	+	**-**
	Ort.Süre	34.017	**35.107**	38.143	**36.748**
GA-TP	Ortalama	794012.905	**851495.325**	893740.954	**928941.750**
	Gap	0.07226	**0.00000**	0.07438	**0.00000**
	Opt.Say.	14	**30**	12	**30**
	Std.Sap.	690.560	**0.000**	655.920	**0.000**
	İşaret	+	**-**	+	**-**
	Ort.Süre	33.885	**35.930**	37.906	**36.609**
GA-UP	Ortalama	793865.023	851517.200	893808.891	**928941.750**
	Gap	0.05362	0.00257	0.08198	**0.00000**
	Opt.Say.	15	29	9	**30**
	Std.Sap.	433.467	119.817	628.654	**0.000**
	İşaret	+	-	+	**-**
	Ort.Süre	33.563	36.329	38.038	**36.588**
BAAA-Tanh	Ortalama	793525.591	**851495.325**	893333.515	**928941.750**
	Gap	0.01084	**0.00000**	0.02875	**0.00000**
	Opt.Say.	27	**30**	16	**30**
	Std.Sap.	262.498	**0.000**	324.451	**0.000**
	İşaret	-	**-**	+	**-**
	Ort.Süre	45.498	**57.039**	56.239	**56.622**
BAAA-Sig	Ortalama	**793439.563**	**851495.325**	**893076.713**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	İşaret	**-**	**-**	**-**	**-**
	Ort.Süre	**63.656**	**67.132**	**66.943**	**67.612**
BPSO	Ortalama	794797.761	851991.551	893816.653	930756.565
	Gap	0.17118	0.05828	0.08285	0.19536
	Opt.Say.	10	21	10	18
	Std.Sap.	1505.749	1055.238	690.192	2594.211
	İşaret	+	+	+	+
	Ort.Süre	60.083	59.759	59.733	59.516
SAAA	Ortalama	**793439.563**	**851495.325**	**893076.713**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
	Ort.Süre	**28.080**	**28.539**	**28.336**	**28.011**

Çizelge 4.9. SAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, çok büyük boyutlu CapA, CapB ve CapC problemleri üzerinde karşılaştırılması

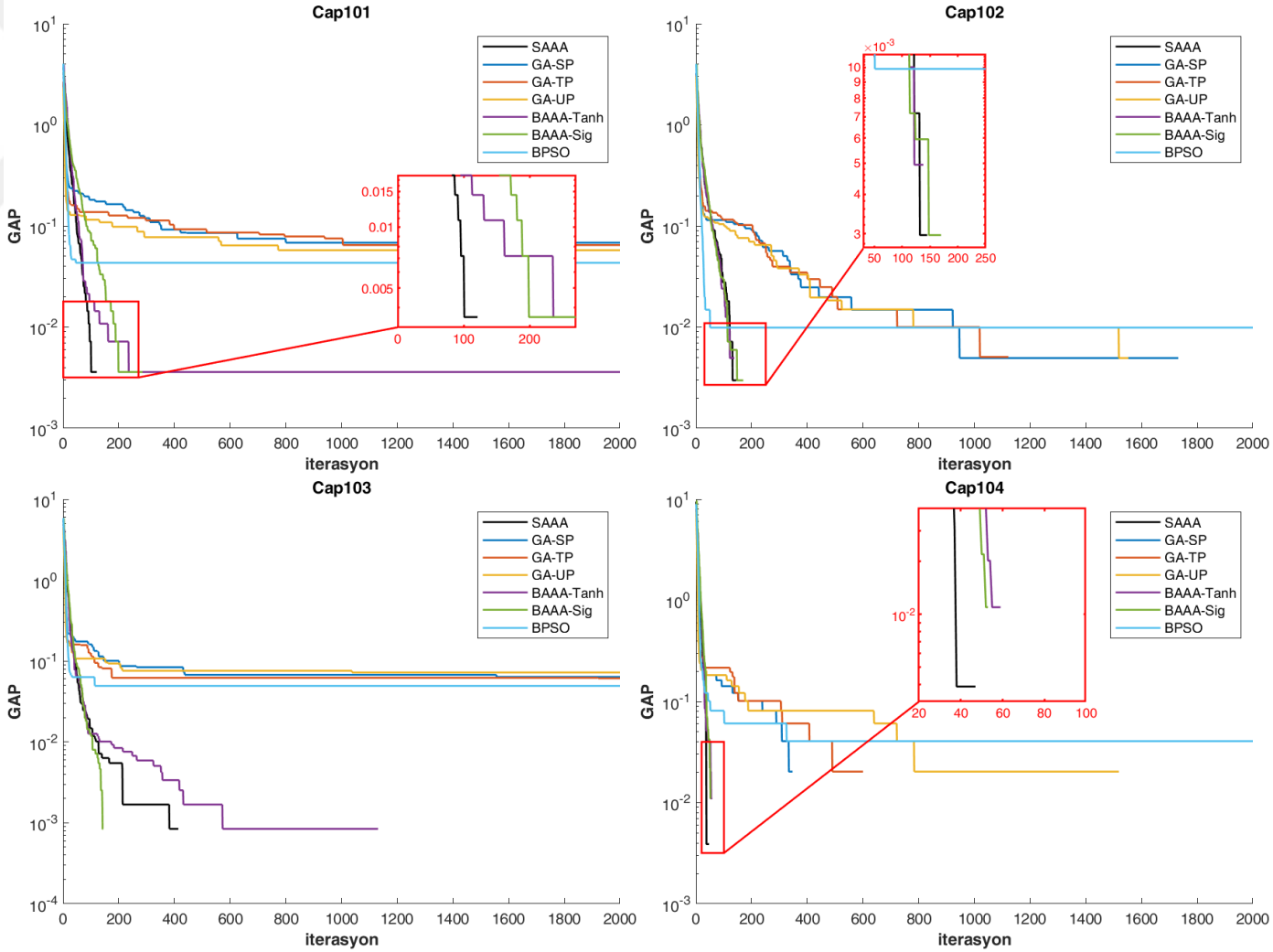
Yöntem		CapA	CapB	CapC
GA-SP	Ortalama	17164354.456	13054858.045	11586692.969
	Gap	0.04605	0.58391	0.70486
	Opt.Say.	24	9	2
	Std.Sap.	22451.206	66658.649	51848.248
	İşaret	+	+	+
	Ort.Süre	741.535	743.370	744.455
GA-TP	Ortalama	17205089.145	13063527.186	11577797.524
	Gap	0.28348	0.65071	0.62755
	Opt.Say.	24	11	0
	Std.Sap.	139690.216	89122.485	46346.052
	İşaret	+	+	+
	Ort.Süre	735.939	738.387	741.823
GA-UP	Ortalama	17166811.915	13107633.077	11578600.532
	Gap	0.06037	0.99053	0.63453
	Opt.Say.	24	3	0
	Std.Sap.	35181.974	79714.021	57031.219
	İşaret	+	+	+
	Ort.Süre	743.848	748.144	745.128
BAAA-Tanh	Ortalama	17471223.794	13153617.764	11676427.752
	Gap	1.83470	1.34483	1.48479
	Opt.Say.	3	0	0
	Std.Sap.	225123.921	73978.543	101438.607
	İşaret	+	+	+
	Ort.Süre	864.755	473.660	479.762
BAAA-Sig	Ortalama	17210900.533	13093705.559	11583462.068
	Gap	0.31735	0.88322	0.67678
	Opt.Say.	16	1	1
	Std.Sap.	90743.456	62168.803	45788.678
	İşaret	+	+	+
	Ort.Süre	880.452	475.592	470.911
BPSO	Ortalama	17446511,870	13161205,473	11692212,797
	Gap	1,69066	1,40329	1,62198
	Opt.Say.	8	5	1
	Std.Sap.	319855,431	135326,728	115156,444
	İşaret	+	+	+
	Ort.Süre	534,103	539,601	541,035
SAAA	Ortalama	**17156454.478**	**13011234.616**	**11539496.443**
	Gap	**0.00000**	**0.24781**	**0.29466**
	Opt.Say.	**30**	**15**	**1**
	Std.Sap.	**0.000**	**39224.744**	**29766.311**
	Ort.Süre	**461.906**	**470.094**	**455.470**

CapB ve CapC hariç tüm problemlerde, SAAA yöntemi 30 çalıştırmanın tümünde optimum çözümü bulmuştur. Az boyutlu problemlerde, karşılaştırılan ve önerilen algoritmalar çözüm kalitesi ve sağlamlık açısından benzer performans göstermektedirler. Problemlerin boyut sayısı arttığında, SAAA yöntemi çözüm kalitesi açısından karşılaştırılan algoritmalardan daha üstün bir performans sergilemektedir. Özellikle çok büyük boyutlu problemlerde (CapA, CapB ve CapC), bu performans farklılıkları karşılaştırılan algoritmalar arasında açıkça görülmektedir.

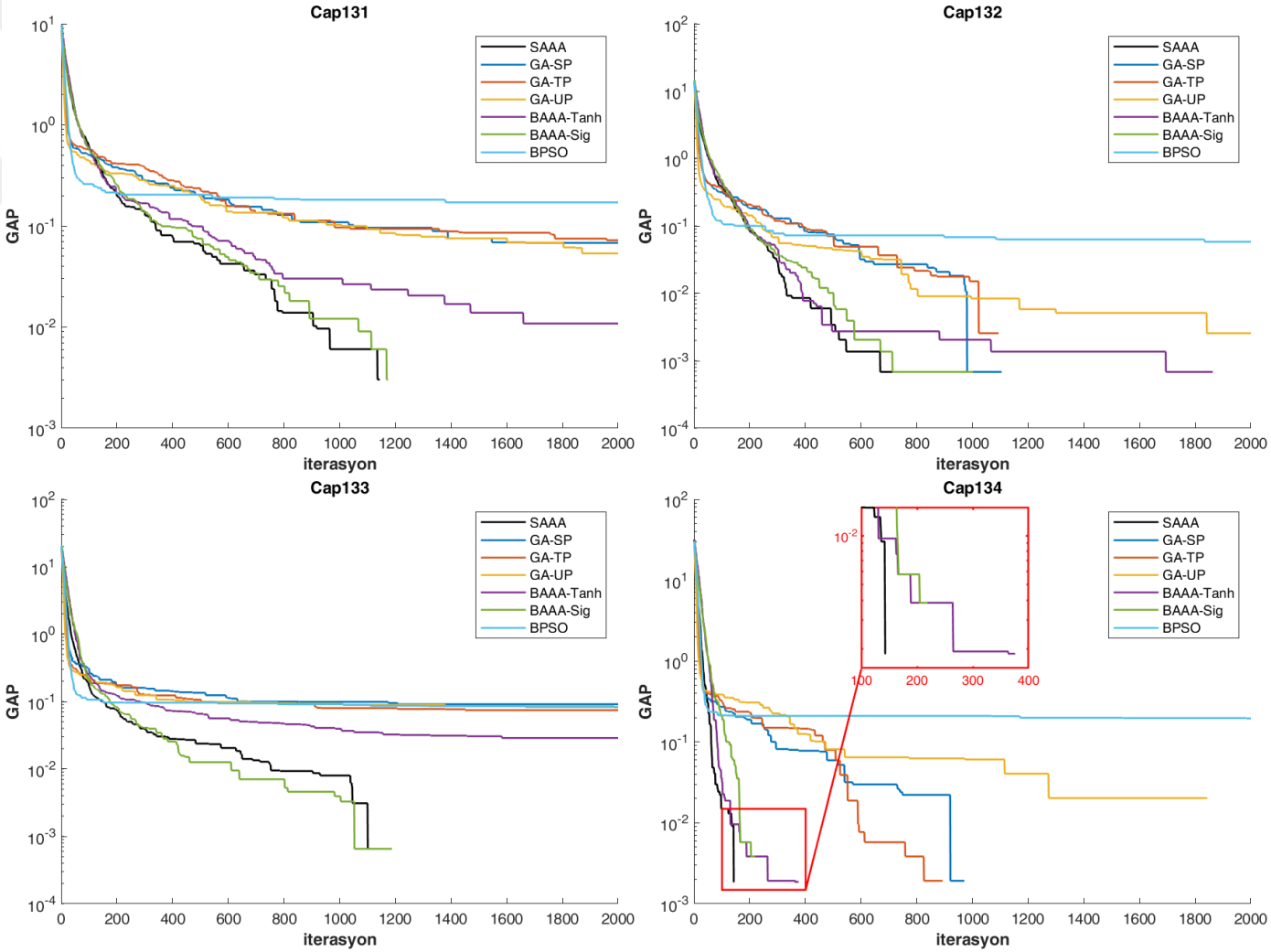
Karşılaştırılan algoritmaların çözüme ulaşma hızlarının daha iyi değerlendirilebilmesi için her bir problem için üretilen yakınsama eğrileri Şekil 4.7-4.10'da sunulmuştur.



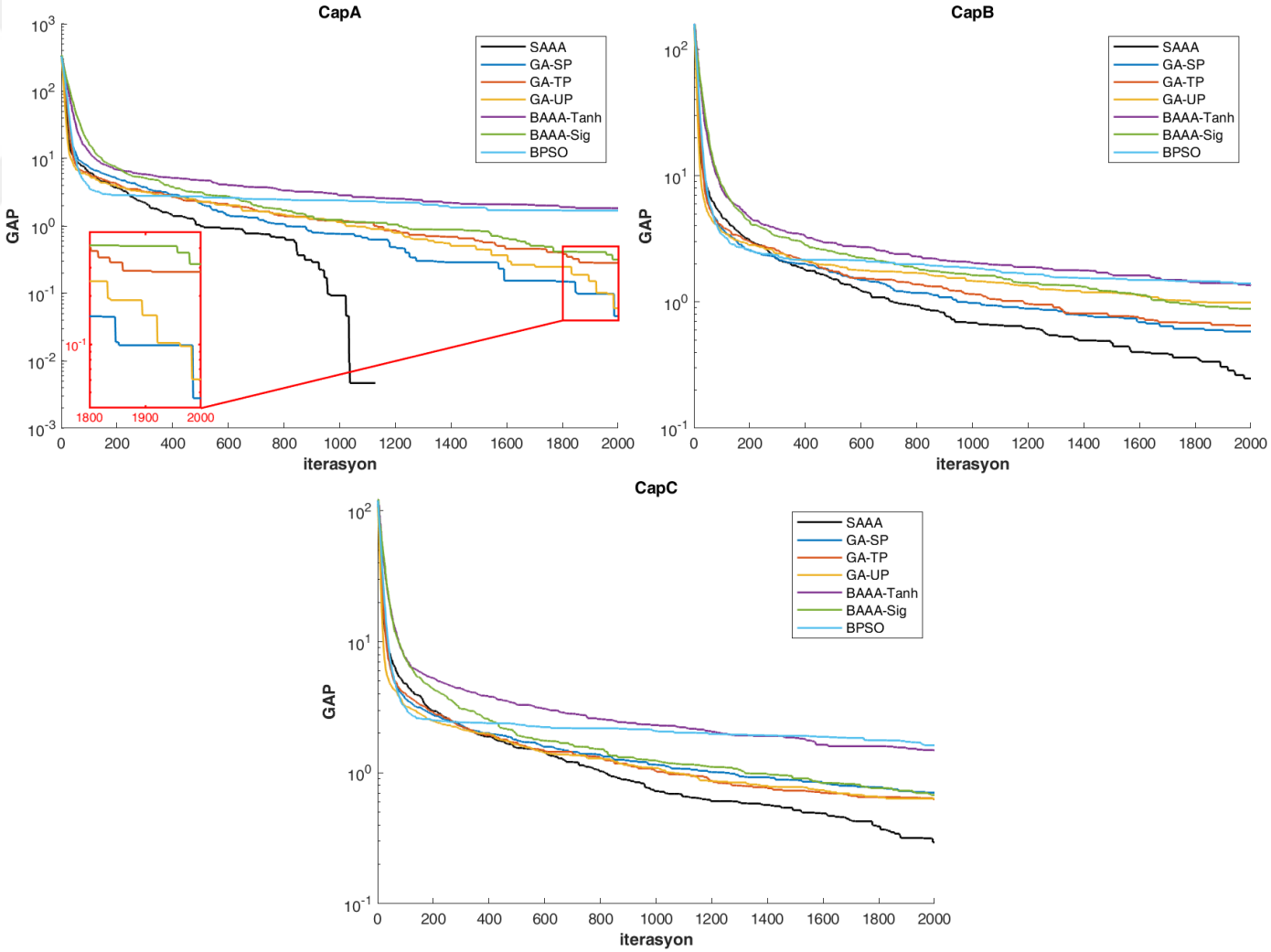
Şekil 4.7. SAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, az boyutlu Cap71, Cap72, Cap73 ve Cap74 problemleri için ürettikleri yakınsama eğrileri



Şekil 4.8. SAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, orta boyutlu Cap101, Cap102, Cap103 ve Cap104 problemleri için ürettikleri yakınsama eğrileri



Şekil 4.9. SAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, büyük boyutlu Cap131, Cap132, Cap133 ve Cap134 problemleri için ürettikleri yakınsama eğrileri



Şekil 4.10. SAAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, çok büyük boyutlu CapA, CapB ve CapC problemleri için ürettikleri yakınsama eğrileri

Çizelgelerden elde edilen sonuçlar ile algoritmaların yakınsama özelliklerinin karşılaştırılmasında da benzerlikler mevcuttur. Az boyutlu problemlerde, karşılaştırılan ve önerilen algoritmalar yakınsama hızları açısından birbirlerine yakın performans sergilemektedirler. Problemlerin boyut sayısı arttığında, önerilen algoritma, yakınsama hızı açısından karşılaştırılan algoritmalardan daha üstün bir performans sergilemektedir. Özellikle Şekil 4.10'da görüldüğü üzere, SAAA yöntemi, CapA probleminde optimum değere çok hızlı bir şekilde ulaşabilmiştir. Yine aynı şekilde, önerilen algoritma, CapB ve CapC problemleri için, diğer algoritmalara kıyasla daha hızlı bir şekilde yakın optimum değerlerine erişebilmiştir.

Bir algoritmanın başarısı sadece yöntemin davranış şekline veya algoritmik tasarıma bağlı değildir, aynı zamanda optimizasyon problemlerinin boyutsallığına ve karakteristiğine de bağlıdır. Genetik algoritma iyi bir global arama yeteneğine sahiptir ve yüksek boyutlu problemler üzerinde daha kaliteli sonuçlar üretebilmektedir. Ancak yerel aramada diğer algoritmalara nazaran daha zayıf kalmaktadır.

Kromozomlar, arama uzayında yakın noktalar üzerinde toplandığında, Genetik algoritma, çaprazlama operatöründen kaynaklanan bir durağanlaşma (stagnation) davranışı göstermektedir. Bu problemin üstesinden gelebilmek için, mutasyon olasılığı parametresinin optimizasyon probleminin özelliklerine ve boyutlarına göre ayarlanması gerekmektedir. Bu durum Çizelge 4.6-4.9'da açıkça görülmektedir. Bu çizelgelere göre, algoritma daha yüksek boyutlu problemler üzerinde daha iyi veya karşılaştırılabilir çözümler üretebilirken, düşük boyutlu problemler üzerinde iyi kalitede veya karşılaştırılabilir çözümler üretememektedir. Önerilen algoritma, neredeyse tüm durumlarda GA'lardan daha iyidir, çünkü algoritmanın ikilileştirme yapısı sayesinde hem keşif (exploration) hem de sömürü (exploitation) yapabilmektedir. GA'nın davranışının aksine, PSO algoritması en iyi çözümleri (kişisel ve global) takip etmektedir ve bu davranış şekli düşük boyutlu problemlerin çözümünde faydalı olmaktadır. Fakat büyük boyutlu problemlerde yerel minimuma takılabilmektedir. Çünkü en iyi çözümlerin de dâhil olduğu tüm çözümler bir arada hareket etmekte ve elde edilen çözümler birbirine benzemekte ve keşif (exploration) kabiliyeti zayıflamaktadır. AAA'nın ikili varyantlarını düşündüğümüzde, algoritmalar yalnızca popülasyondaki var olan bilgileri kullanmaktadır. Yöntemler, CapA, CapB ve CapC hariç diğer problemlerde rekabetçi sonuçlar üretebilmektedir. SAAA yöntemi ise, stigmerjik davranışı sayesinde, hem popülasyondaki bilgiyi hem de probleme özgü bilgiyi kullanmakta ve daha kaliteli çözümler üretebilmektedir.

UFLP test seti kullanılarak yapılan ilk karşılaştırmada, önerilen yöntem BPSO, BAAA varyantları ve GA'lar ile kıyaslanmıştır. UFLP test seti kullanılarak yapılan ikinci karşılaştırmada ise, önerilen yöntem, literatürdeki çalışmalardan derlenen sonuçlar ile kıyaslanmıştır. CPSO, Farklılık Tabanlı DE Algoritması (Dissimilarity Based DE Algorithm-DisDE) ve binDE yöntemlerinin sonuçları doğrudan Kashan ve ark. (2013) tarafından yapılan çalışmadan, ABCbin yönteminin sonuçları doğrudan Kiran (2015a) tarafından yapılan çalışmadan ve DisABC ve binABC yöntemlerinin sonuçları ise doğrudan Kiran ve Gunduz (2013) tarafından yapılan çalışmadan alınmıştır. Bu algoritmaların uygulama detayları ilgili çalışmalardan elde edilebilir. Elde edilen bu sonuçlara dayanılarak yapılan karşılaştırma Çizelge 4.10'da sunulmuştur.

Çizelge 4.10. SAAA yöntemi ile ABC, DE ve PSO'nun ikili sürümleri ile UFLP test seti üzerinde karşılaştırılması

Problemler	CPSO		ABCbin		DisDE		binDE		DisABC		binABC		SAAA	
	Gap	Sıra	Gap	Sıra	Gap	Sıra	Gap	Sıra	Gap	Sıra	Gap	Sıra	Gap	Sıra
Cap71	5.0E-02	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap72	7.0E-02	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap73	6.0E-02	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap74	7.0E-02	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap101	1.4E-01	3	**0.0E+00**	**1**	7.2E-03	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap102	1.5E-01	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap103	1.6E-01	4	5.1E-03	3	8.4E-04	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap104	1.8E-01	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap131	7.5E-01	5	2.0E-01	3	**0.0E+00**	**1**	3.6E-03	2	6.2E-01	4	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap132	7.8E-01	5	2.0E-02	3	**0.0E+00**	**1**	5.0E-03	2	9.5E-02	4	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap133	7.3E-01	7	7.5E-02	5	1.5E-02	3	1.4E-02	2	3.1E-02	4	1.2E-01	6	**0.0E+00**	**1**
Cap134	8.9E-01	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
CapA	2.2E+01	7	3.2E+00	6	3.7E-02	2	1.3E+00	4	1.5E-01	3	3.0E+00	5	**0.0E+00**	**1**
CapB	1.1E+01	7	2.8E+00	5	**6.7E-02**	**1**	1.5E+00	3	3.3E+00	6	2.5E+00	4	2.5E-01	2
CapC	9.7E+00	7	2.0E+00	4	**5.8E-02**	**1**	1.6E+00	3	4.7E+00	6	2.6E+00	5	2.9E-01	2
Ort.Sıra		3.93		2.47		1.33		1.67		2.40		2.07		1.13
DOS		7		6		2		3		5		4		1

Çizelge 4.10’ da her bir problem için, yöntemlerin sonuçları değerlendirilerek sıralama yapılmıştır. Örneğin Cap71 problemi için, CPSO yöntemi haricindeki bütün algoritmalar optimum değere eriştiğinden dolayı birinci sırada ve CPSO ise ikinci sıradadır. Çizelgedeki *Ort.Sıra* satırında ise her bir algoritmanın sıralaması toplanıp problem sayısına bölünerek elde edilen değer mevcuttur. DOS (Düzeltilmiş Ortalama Sıra) satırında ise yöntemlerin bütün problemlerdeki ortalama sıralama değerleri kullanılarak genel bir sıralama daha yapılmıştır.

Çizelge 4.10’dan görüldüğü üzere, önerilen yöntem DOS sıralamasında ilk sırada yer almaktadır. Ayrıca 15 problemin 13’ünde en uygun çözümleri elde ederek problem bazında birinci sırada yer almayı başarmıştır. CapB ve CapC problemlerinde DisDE yöntemi birinci sırada yer alırken, SAAA yöntemi ise ikinci sıradadır. DisDE algoritması, aday çözümleri üretmek için Jaccard’ın benzerliğini kullanır ve keşif yeteneği diğer algoritmalara göre nispeten daha iyidir. Ancak DisDE sömürü (exploitation) konusunda aynı başarıya sahip değildir. Cap101, Cap103, Cap133 ve CapA problemleri için ürettiği sonuçlar incelendiğinde, sıralamada önerilen yöntemin gerisinde kaldığı görülmektedir.

Bütün sonuçlar genel olarak incelendiğinde, önerilen algoritma sadece düşük boyutlu problemlerde değil, aynı zamanda yüksek boyutlu problemler için de dengeli bir keşif ve sömürü kabiliyeti sunmaktadır.

4.2.2.2 CEC2015 Problem Seti Kullanılarak Yapılan Karşılaştırma

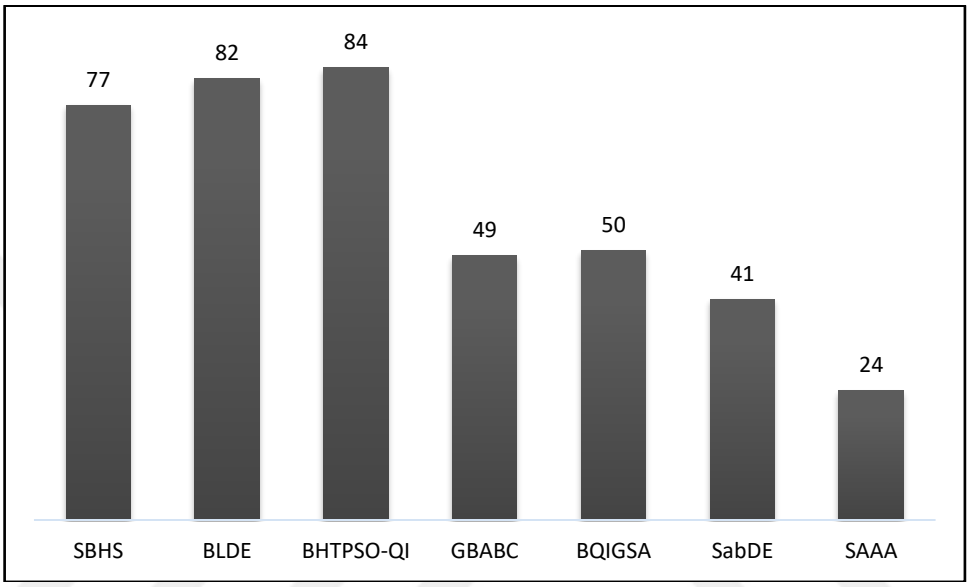
Önerilen yöntem için yapılan bir diğer deneysel çalışma CEC2015 (bound constrained single-objective computationally expensive numerical optimization problems) (Chen ve ark., 2014) problem seti üzerinde gerçekleştirilmiştir. Önerilen algoritma bu problemler ile 30 kez çalıştırılmış ve elde edilen sonuçlar yakın zamanda önerilmiş algoritmaların sonuçları ile karşılaştırılmıştır. Bu yöntemlerin sonuçları doğrudan Banitalebi ve ark. (2016) tarafından yapılan çalışmadan alınmıştır. Algoritmaların adil bir şekilde karşılaştırılabilmesi için, önerilen algoritmanın ortak kontrol parametreleri (Banitalebi ve ark., 2016) ilgili çalışmadaki değerler ile kullanılmıştır. Bu bağlamda, popülasyon büyüklüğü değeri 50 ve durdurma kriteri olarak MaxFEs değeri 100,000 olarak kullanılmıştır. Karar değişkenleri sürekli değerler alan problemin boyutu (D) 10 olarak seçilmiş ve bu karar değişkenlerini ikili değerler ile ifade etmek amacı ile her boyut için 50 bit kullanılmıştır. Böylece popülasyondaki her bir birey ikili değerler alabilen 500 adet boyuta sahip olacaktır. Her bir bireyin uygunluk

değerlerinin hesaplanabilmesi için 500 bitlik bu ikili değerler, sürekli değerlere dönüştürülmesi gerekmektedir. Bu dönüşüm işlemi Denklem 3.13 kullanılarak hesaplanmaktadır. AAA'nın temel çalışmasında (Uymaz ve ark., 2015b) olduğu gibi, enerji kaybı parametresi değeri 0.3 ve adaptasyon parametresi değeri 0.5 olarak kullanılmıştır. SAAA yönteminde, algler ayrık çözüm uzayında çalıştığından dolayı temel AAA'da var olan kesme kuvveti parametresi kullanılmamaktadır. SAAA'daki yönteme özgü kontrol parametrelerinden, UMSP kontrol parametresinin değeri 0.5 ve DSP kontrol parametresinin değeri 0.66 olarak kullanılmıştır. Bu koşullar altında, elde edilen sonuçlar *ortalama değer*, *standart sapma* ve *sıra* değerleri ile Çizelge 4.11'de sunulmuştur. Çizelgede gösterilen *sıra* değerleri, her bir problem için yöntemlerin ortalama değerleri temel alınarak yapılmıştır. Ayrıca her bir problem için birinci olan yöntemin değerleri kalın yazı tipinde verilmiştir.

Çizelge 4.11. SAAA yöntemi ile yakın zamanda önerilen algoritmaların CEC2015 test seti üzerinde karşılaştırılması

		SBHS	BLDE	BHTPSO-QI	GBABC	BQIGSA	SabDE	SAAA
F1	Ortalama	3.700E+08	1.087E+11	7.510E+08	2.729E+07	8.419E+07	3.093E+08	**2.140E+07**
	Std.Sap.	2.271E+08	2.351E+11	2.203E+09	2.267E+07	7.354E+07	1.168E+08	**2.443E+07**
	Sıra	5	7	6	2	3	4	**1**
F2	Ortalama	4.493E+09	4.747E+09	5.132E+09	2.864E+09	7.834E+09	2.541E+09	**1.458E+04**
	Std.Sap.	1.790E+06	3.209E+08	4.685E+08	2.374E+09	6.527E+09	5.008E+09	**5.211E+03**
	Sıra	4	5	6	3	7	2	**1**
F3	Ortalama	3.203E+02	3.201E+02	3.203E+02	3.202E+02	3.202E+02	3.200E+02	**3.061E+02**
	Std.Sap.	7.013E-02	1.093E-02	7.426E-02	2.641E+02	2.641E+02	2.044E-02	**9.017E-01**
	Sıra	5	3	5	4	4	2	**1**
F4	Ortalama	4.478E+02	4.378E+02	4.478E+02	4.358E+02	4.389E+02	**4.116E+02**	5.014E+02
	Std.Sap.	5.171E+00	7.754E+00	1.122E+01	3.599E+02	3.621E+02	**7.606E+00**	6.983E+01
	Sıra	5	3	5	2	4	**1**	6
F5	Ortalama	2.592E+03	1.934E+03	1.804E+03	1.108E+03	1.542E+03	9.330E+02	**5.004E+02**
	Std.Sap.	1.959E+02	3.088E+02	3.926E+02	9.736E+02	1.275E+03	9.464E+01	**1.445E-01**
	Sıra	7	6	5	3	4	2	**1**
F6	Ortalama	5.312E+04	2.484E+06	6.799E+06	7.442E+06	5.582E+05	4.625E+04	**6.003E+02**
	Std.Sap.	1.512E+04	7.369E+06	1.186E+07	1.321E+07	6.055E+05	4.076E+04	**9.229E-02**
	Sıra	3	5	6	7	4	2	**1**
F7	Ortalama	8.368E+02	8.472E+02	8.510E+02	7.589E+02	7.392E+02	7.752E+02	**7.002E+02**
	Std.Sap.	7.981E+00	1.289E+01	1.694E+01	4.668E+02	4.324E+02	4.155E+03	**1.545E-01**
	Sıra	5	6	7	3	2	4	**1**
F8	Ortalama	3.344E+08	7.104E+09	5.407E+07	3.949E+07	2.948E+06	2.395E+07	**8.052E+02**
	Std.Sap.	1.810E+09	5.651E+09	8.312E+07	2.442E+08	2.469E+06	5.432E+07	**1.475E+00**
	Sıra	6	7	5	4	2	3	**1**
F9	Ortalama	1.189E+03	1.190E+03	1.209E+03	1.017E+03	1.048E+03	1.177E+03	**9.033E+02**
	Std.Sap.	5.210E+00	1.736E+01	2.050E+01	8.397E+02	8.643E+02	4.102E+01	**2.616E-01**
	Sıra	5	6	7	2	3	4	**1**
F10	Ortalama	9.038E+08	1.326E+10	1.320E+09	9.909E+05	4.426E+04	2.416E+07	**1.293E+04**
	Std.Sap.	6.596E+08	1.724E+10	2.073E+09	3.659E+06	4.477E+04	8.862E+07	**1.068E+04**
	Sıra	5	7	6	3	2	4	**1**
F11	Ortalama	1.737E+03	1.599E+03	1.632E+03	1.159E+03	1.172E+03	1.114E+03	**1.105E+03**
	Std.Sap.	1.661E+01	3.313E+01	3.371E+01	9.557E+02	9.669E+02	1.131E+01	**8.947E-01**
	Sıra	7	5	6	3	4	2	**1**
F12	Ortalama	1.445E+03	1.440E+03	1.437E+03	1.264E+03	1.255E+03	**1.224E+03**	1.249E+03
	Std.Sap.	5.683E-01	3.247E+00	7.552E+00	1.044E+03	1.035E+03	**1.302E+00**	2.176E+01
	Sıra	7	6	5	4	3	**1**	2
F13	Ortalama	1.464E+10	2.882E+10	5.365E+08	**1.446E+03**	1.452E+03	2.815E+09	1.627E+03
	Std.Sap.	9.749E+09	2.073E+10	2.294E+09	**1.193E+03**	1.197E+03	4.158E+09	5.024E+00
	Sıra	6	7	4	**1**	2	5	3
F14	Ortalama	2.736E+03	4.267E+03	5.005E+03	2.162E+03	3.356E+03	1.727E+03	**1.600E+03**
	Std.Sap.	2.220E+02	2.569E+03	2.703E+03	2.091E+03	2.869E+03	4.411E+02	**3.715E+00**
	Sıra	4	6	7	3	5	2	**1**
F15	Ortalama	1.700E+03	1.700E+03	1.734E+03	2.012E+03	**1.530E+03**	1.700E+03	1.614E+03
	Std.Sap.	1.552E-04	2.622E-05	1.851E+02	1.659E+03	**1.262E+03**	2.177E-05	1.257E+02
	Sıra	3	3	4	5	**1**	3	2

Çizelge 4.11'de görüldüğü üzere, SabDE algoritması F4 ve F12 problemleri için diğer algoritmalardan daha kaliteli sonuçlar üretmiştir. F13 problemi için en iyi sonuçlar GBABC yöntemi tarafından sağlanmıştır. Bunların dışındaki bütün problemler için, önerilen algoritma, ortalama değerler ve standart sapma değerleri temel alınarak yapılan karşılaştırmada, çözüm kalitesi ve sağlamlık (robustness) açısından diğer yöntemlerden daha üstündür.



Şekil 4.11. Sıralama mekanizmasına göre algoritmaların karşılaştırılması.

Yöntemlerin performanslarının daha iyi anlaşılması adına, Çizelge 4.11'deki *sıra* değerleri, her bir algoritma için toplanarak grafiksel olarak Şekil 4.11'de verilmiştir. Şekil 4.11'deki her algoritma için var olan değerin az olması yöntemin daha başarılı olduğunu göstermektedir. Bu açıdan bakıldığında SAAA yöntemi, CEC2015 problem seti üzerinde yapılan kıyaslamada, diğer algoritmalara göre çok daha iyi sonuçlar ürettiği görülmektedir. SAAA, sıralama toplamında 24 değeri ile birinci iken, SabDE algoritması 41 değeri ile ikinci olabilmiştir. En kötü performans ise 84 sıralama toplam değeri ile BHTPSO-QI yöntemine aittir.

4.3. Popülasyon Etkisi Kullanılarak Geliştirilen İkili AAA (PI-AAA)

4.3.1. Önerilen Yöntem

İkili optimizasyon problemlerini çözmek için, sürekli karar değişkenleri ile çalışan temel algoritmanın, sürekli değerler ile çalışan kısımları, aday çözüm üretme tekniği uygulanmadan önce değiştirilmesi gerekmektedir. B*aşlangıç Aşamasında* temel AAA'nın kullandığı Denklem 3.1 üzerinde modifikasyon yapılarak Denklem 4.16 üretilmiştir.

$$X_{i,j} = \begin{cases} 0\,, if\ (r_{i,j} < a) \\ 1\,, aksi\ takdirde \end{cases} \quad , i = 1,2,\dots,N\ ve\ j = 1,2,\dots,D \tag{4.16}$$

Burada, a [0,1] aralığında seçilebilen olasılık değeri olup, karar değişkenlerinin rastgele dağıtılabilmesi için 0.5 olarak belirlenmiştir. $r_{i,j}$ [0,1] aralığında üretilen rastgele sayıyı, $X_{i,j}$ popülasyondaki i'nci bireyin (alg kolonisi) j'inci hücresini (boyut) ifade etmektedir.

Optimizasyon problemi için spesifik bir fonksiyon olan objektif fonksiyondan sonra her bir alg kolonisinin uygunluğu hesaplanır ve her bir alg kolonisinin büyüklüğü elde edilir. Hesaplanan bu değerler, optimizasyon problemlerinin türünden bağımsız olduğundan dolayı temel versiyondaki gibi bırakılmış ve değiştirilmemiştir. Ancak Denklem 3.6, Denklem 3.7 ve Denklem 3.8'te verilen aday üretim tekniklerinin değiştirilmesi gerekmektedir. Çünkü bu denklemler sürekli değerler ile çalışmakta ve sonuç olarak da sürekli değerler üretmektedirler.

Bu çalışmada, ikili aday çözümlerin oluşturulması için, *popülasyon etkisi* (population influence) yöntemi tanımlanmıştır. Bu yöntem, aday çözümler üretebilmek için, mevcut çözümü, en iyi çözümü ve alg kolonilerinden rastgele seçilen komşu çözümleri kullanmaktadır. PI-AAA yöntemine ait aday çözüm üretme fonksiyonunun sözde kodu Şekil 4.12'de verilmiştir:

```
1)  Girdi: Xc, Xb, NS, NNS,C1, C2, C3
2)  V= Xc;
3)  A={j,m,n} rastgele seçilmiş boyut indisleri
4)  For each element (k) in A
5)    P0=P1=0.1;
6)    if Xc,k = 0 then { P1=P1+C1; }
7)      else { P0=P0+C1; }
8)    if Xb,k = 0 then {P0=P0+C2; }
9)      else { P1=P1+C2; }
10)   For i=1 to NNS
11)     if NSi,k = 0 then { P0=P0+C3/NNS; }
12)       else { P1=P1+ C3/NNS; }
13)   End For
14)   if(rand < P0) then Vk=0;
15)     else Vk=1;
16) EndFor
17) Return V
```

Şekil 4.12. PI-AAA yöntemine ait aday çözüm üretme fonksiyonunun sözde kodu

4.12'de verilen fonksiyonda, X_c mevcut çözümü, X_b popülasyondaki en iyi çözümü, NS (Neighbor Solutions) popülasyondan rastgele seçilen komşu çözümleri ve NNS (Number of Neighbor Solutions) ise komşu çözümlerin sayısını ifade etmektedir. Başlangıç aşamasında, mevcut çözüm aday çözüme kopyalanır ve üç adet (j, m ve n) karar değişkeni rastgele seçilir.

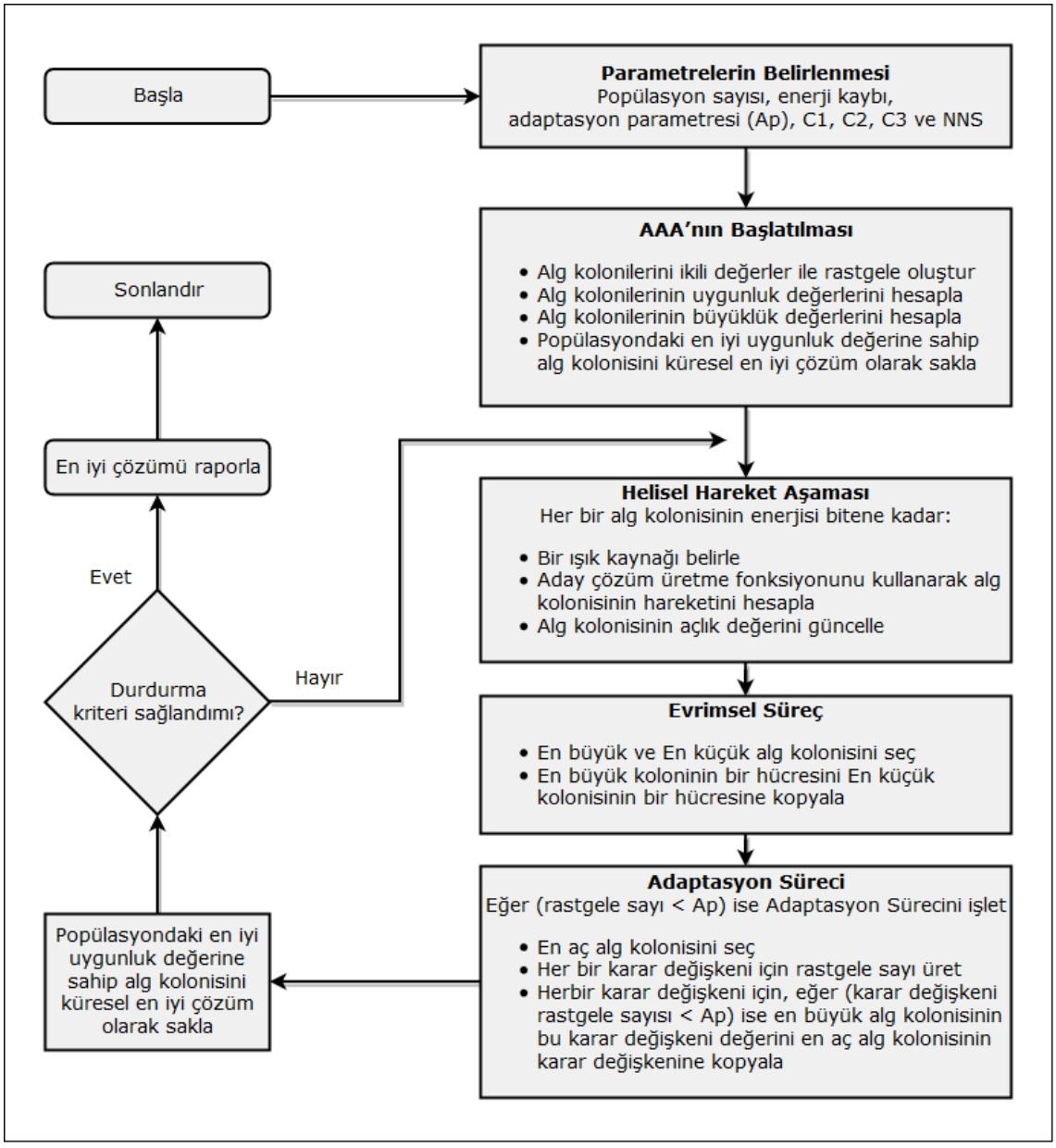
Her bir boyut için, mevcut, en iyi çözüm ve komşu çözümlerin etkileri, ilgili karar değişkeninin durumu da dikkate alınarak hesaplanır. P_0 ve P_1 değerleri, ilgili karar değişkeni için 0 ve 1 olma olasılıklarını ifade etmektedir ve başlangıç değeri olarak ikisine de 0.1 değeri atanmıştır. Eğer mevcut çözümün ilgili boyutunun değeri 0 ise, P_1 değeri C_1 oranında artırılır, aksi halde P_0 değeri C_1 oranında artırılır. PI-AAA (Popülasyon Etkili AAA - Population Influenced AAA) yöntemine özgü tanımlanmış kontrol parametresi olan C_1, aday çözüm üretirken kullanılan, mevcut çözümden etkilenme oranıdır. C_2 kontrol parametresi ise en iyi çözümden etkilenme oranıdır. Eğer en iyi çözümün ilgili boyutunun değeri 0 ise, P_0 değeri C_2 oranında artırılır. Bunun aksine, en iyi çözümün ilgili boyutunun değeri 1 ise, P_1 değeri C_2 oranında artırılır. C_3 kontrol parametresi ise komşu çözümlerden etkilenme oranıdır. Komşu çözümlerin etkisi, seçilen komşu sayısıyla orantılıdır ve her bir komşunun etkisini belirlemek amacı ile C_3 parametresi komşuların sayısına bölünür. Eğer rastgele seçilmiş komşu çözümlerden birinin ilgili boyutunun değeri 0 ise, P_0 değeri C_3/NNS oranında artırılır. Bunun aksine, rastgele seçilmiş komşu çözümlerden birinin ilgili boyutunun değeri 1 ise, P_1 değeri C_3/NNS oranında artırılır.

Ayrıca, P_0 ve P_1 olasılık değerlerinin toplamı başlangıçta 0.2 olarak belirlendiği için, P_0 ve P_1 olasılık değerlerinin toplamının 1 olması gerektiğinden dolayı, C_1, C_2 ve C_3 kontrol parametrelerinin değerlerinin toplamı 0.8 olmalıdır. Sonuç olarak, PI-AAA yönteminde yeni bir aday çözüm üretilirken, mevcut çözüm, en iyi çözüm ve komşu çözümler kullanılmakta ve bunlar mevcut popülasyondan elde edilmektedir. Dolayısıyla, aday çözümler, popülasyon etkisi altında üretilmektedir. Deneylerde, C_1, C_2, C_3 ve NNS kontrol parametrelerinin değerlerinin önerilen yöntemin performansı için önemli faktörler olması nedeniyle etkileri analiz edilmiş ve bu parametreler için uygun değerler belirlenmiştir.

Temel AAA'da yapılan bir diğer değişiklik ise, algoritmanın adaptasyon aşamasında yapılmıştır. Temel AAA'nın adaptasyon aşamasında sürekli değerler kullanılmakta ve sonuç olarak yine sürekli değerler üretilmektedir. Bu nedenle, adaptasyon aşaması yeniden düzenlenerek, karar değişkenleri ikili değerler alan bir popülasyon ile çalışabilmesi için uyarlanmıştır. Bu bağlamda, yönteme özgü adaptasyon kontrol parametresi, algoritma içerisinde iki farklı işlev için kullanılmıştır. İlk olarak temel AAA'da olduğu üzere, [0,1] aralığında rastgele üretilen bir sayı eğer adaptasyon parametresinden küçük ise, ilgili iterasyonda adaptasyon fazı işletilmekte, rastgele üretilen sayı eğer adaptasyon parametresinden büyük veya eşit ise ilgili iterasyonda adaptasyon fazı işletilmemektedir. Mevcut iterasyonda eğer adaptasyon fazı işletilecekse, adaptasyon parametresinin ikinci işlevi devreye girmektedir. Popülasyondaki en büyük ve en aç koloni seçildikten sonra her bir boyut için [0,1] aralığında rastgele sayı üretilmekte ve adaptasyon parametresi ile karşılaştırılmaktadır. Üretilen sayı adaptasyon parametresinden küçük ise, popülasyondaki en büyük kolonin ilgili boyutu, en aç kolonin aynı boyutuna kopyalanmaktadır. Temel AAA'da, adaptasyon fazında tüm boyutlar kullanılırken, PI-AAA yönteminde ise adaptasyon parametresine bağlı olarak bazı boyutlar kullanılmaktadır. Eğer adaptasyon süreci bütün boyutlarda işletilirse, popülasyondaki en büyük koloni, olduğu gibi en aç koloniye kopyalanır ve popülasyonda aynı karar değişkenlerine sahip iki birey oluşur. Bu durum popülasyonda durağanlaşmaya (stagnation) neden olur ve yöntemin performansını olumsuz etkiler. Bu nedenle temel AAA'daki adaptasyon aşaması değiştirilerek kullanılmıştır.

AAA'nın temel versiyonunda kullanılan kesme kuvveti kontrol parametresi önerilen yöntemde kullanılmamış ve kaldırılmıştır. Temel versiyonda, kesme kuvveti kontrol parametresi, sürekli değerler alan karar değişkenleri kullanarak yeni aday çözümler üretmek için kullanılmaktadır. Önerilen yöntemde karar değişkenleri ikili

değerler aldığından dolayı, yeni aday çözümler üretmek için kesme kuvveti kontrol parametresine ihtiyaç duyulmamıştır.



Şekil 4.13. PI-AAA yönteminin genel akış şeması

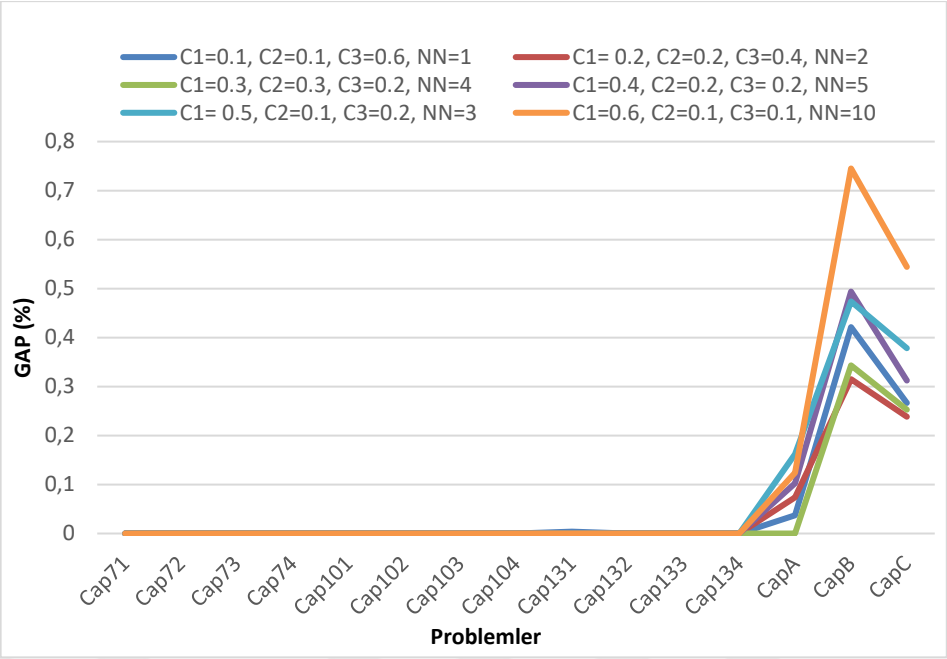
Geliştirilen yöntemin daha iyi anlaşılması adına, PI-AAA'nın genel akış şeması, Şekil 4.13'de verilmiştir. Temel AAA'daki sürekli değerler ile çalışan bütün kısımlar, ikili optimizasyon problemlerini çözmek için, ikili değerler ile çalışacak şekilde uyarlanmıştır.

4.3.2. PI-AAA Yönteminin Parametre Analizi

PI-AAA'nın, yakın zamanda önerilen algoritmalar ile karşılaştırılmasından önce, kontrol parametrelerinin uygun değerlerini belirleyebilmek için, yönteme özgü C_1, C_2, C_3 ve NNS kontrol parametrelerinin analizleri yapılmıştır. Yapılan analiz çalıştırmalarında, popülasyon büyüklüğü sayısı 40 ve durdurma kriteri olarak MaxFEs değeri 80,000 olarak kullanılmıştır. AAA'nın temel versiyonunda önerildiği üzere, adaptasyon parametresi 0.3 ve enerji kaybı parametresi ise 0.5 olarak alınmıştır. Temel versiyonda bulunan kesme kuvveti kontrol parametresi, önerilen yöntemden kaldırıldığından dolayı kullanılmamıştır.

Analizi yapılacak olan C_1, C_2 ve C_3 kontrol parametrelerinin değerlerinin toplamı 0.8 olmalıdır. Ayrıca C_1, C_2 ve C_3 kontrol parametrelerinin minimum değerleri 0.1 olarak seçilmiştir. Eğer C_1 parametresinin değerini 0.2 seçecek olursak, C_2 parametresinin en yüksek değeri 0.5 olabilecektir. Çünkü her bir parametrenin en az 0.1 değerine sahip olması gerekmektedir. Eğer C_1 parametresinin değerini 0.2, C_2 parametresinin değerini de 0.5 den yüksek bir değer seçersek, C_3 parametresinin değeri 0.1 den düşük olacaktır. Eğer C_1 ve C_2 parametrelerinin değerini 0.1 olarak seçersek, komşu çözümlerin, yeni aday çözümler üretmedeki etki oranı 0.6 olacaktır. Eğer komşu çözüm sayısı 1 olarak seçilmiş ise, rastgele seçilecek bir komşunun aday çözüm üretme etki oranı 0.6 olacak, eğer komşu çözüm sayısı 2 seçilmiş ise, her bir komşu çözümün etki oranı 0.3 olacaktır. Bu nedenle, komşu çözümlerin sayısının da analiz edilmesi gerekmektedir. Fakat komşu çözüm sayısı olarak bütün ihtimalleri analiz edemeyeceğimizden dolayı, komşu çözüm sayısı parametresi 1, 2, 3, 4, 5, 10, 20 ve 39 değerleri ile analiz edilmiştir.

PI-AAA'nın bütün parametre ihtimalleri dikkate alındığında, her bir problem için 168 farklı durum ortaya çıkmaktadır. Deneyler, 15 adet probleme sahip UFLP test seti üzerinde gerçekleştirilmiştir. Her bir durum 30 defa bağımsız olarak çalıştırılmıştır. Böylece PI-AAA yöntemi, parametre analizi için 75,600 (168x15x30) defa çalıştırılmıştır. Elde edilen sonuçların tümünün gösterimi mümkün olmadığından dolayı, 168 farklı parametre ihtimali indirgenerek Şekil 4.14' de verilmiştir.



Şekil 4.14. PI-AAA yönteminin parametre analiz sonuçları

Şekil 4.14' de görüldüğü üzere, PI-AAA yöntemi, C_1 parametresinin değeri 0.3, C_2 parametresinin değeri 0.3, C_3 parametresinin değeri 0.2 ve NNS parametresinin değeri 4 olarak seçildiğinde en iyi sonuçlar elde edilmiştir. Bu parametreler ile, 15 farklı problem için 30 defa bağımsız olarak çalıştırıldığında, 450 çalıştırmanın 403'ünde optimum sonuca ulaşılmıştır. Az boyutlu, orta boyutlu, büyük boyutlu problemlerin hepsinde ve çok büyük boyutlu problemlerden CapA probleminde, bütün çalıştırmalarda optimum sonuç elde edilmiştir. CapB probleminde 30 çalıştırmanın 12 sinde, CapC probleminde ise 1 tanesinde optimum sonuca ulaşılmıştır.

PI-AAA'nın, yakın zamanda önerilen algoritmalar ile yapılan kıyaslamalarında, bu bölümde yapılan analiz sonucu belirlenen parametreler kullanılmıştır.

4.3.3. Deneysel Sonuçlar ve Karşılaştırmalar

PI-AAA yönteminin performansının daha iyi değerlendirilebilmesi için karşılaştırmalar iki farklı deney seti üzerinden gerçekleştirilmiştir. İlk karşılaştırma için OR-Library'de (Beasley, 1990a) yayınlanmış, 15 farklı probleme sahip olan UFLP test seti kullanılmış ve önerilen PI-AAA yöntemi, BAAA yönteminin 2 farklı versiyonu, GA yöntemin 3 farklı versiyonu, BPSO, EDA ve UMDA yöntemi ile kıyaslanmıştır. PI-AAA yöntemi ve karşılaştırılan diğer yöntemler, Mathworks şirketinin ticari bir ürünü olan MATLAB ortamında kodlanmıştır. UFLP test seti kullanılarak gerçekleştirilen diğer

karşılaştırmada ise PI-AAA yöntemi, sonuçları literatürdeki çalışmalardan toplanan CPSO, ABCbin, DisDE, binDE, DisABC ve binABC yöntemleri ile kıyaslanmıştır.

İkinci kıyaslama için CEC2015 (Chen ve ark., 2014) test seti kullanılmış ve PI-AAA yönteminin performansının değerlendirilebilmesi için SBHS, BLDE, BHTPSO-QI, GBABC, BQIGSA ve SabDE yöntemleri ile kıyaslanmıştır. PI-AAA yöntemi Mathworks şirketinin ticari bir ürünü olan MATLAB ortamında kodlanmış, karşılaştırılan diğer algoritmaların sonuçları ise doğrudan Banitalebi ve ark. (2016) tarafından yapılan çalışmadan (SabDE) alınmıştır. Ayrıca bütün algoritmalar için çalıştırma ortamı olarak, Intel® Core™ i7-3820 3.60GHz işlemcili, 16GB belleğe sahip ve Windows 10 Pro 64-bit işletim sistemi ile çalışan bir kişisel bilgisayar kullanılmıştır.

4.3.3.1 UFLP Test Seti Kullanılarak Yapılan Karşılaştırmalar

Algoritmaların adil bir şekilde kıyaslanabilmesi için, yöntemlerin ortak kontrol parametreleri birbirine eşit olarak seçilmiştir. Bu bağlamda, popülasyon büyüklüğü 40 ve sonlandırma koşulu olarak, MaxFEs değeri 80,000 olarak seçilmiştir. MaxFEs değerine paralel olarak, gerekli yöntemler için sonlandırma koşulu olarak, maksimum iterasyon sayısı 2,000 olarak kullanılmıştır. GA'nın 3 farklı versiyonu (tek noktadan çaprazlama, iki noktadan çaprazlama ve çok noktadan çaprazlama) için, Holland (1992a) tarafından yapılan çalışmada olduğu üzere, çaprazlama oranı 0.8 ve mutasyon oranı 0.01 olarak seçilmiştir. Ebeveyn seçiminde turnuva seçim yöntemi kullanılmış, turnuva boyutu 10 olarak seçilmiştir. GA için 3 farklı çaprazlama operatörü kullanılmış ve bunların her birinin sonuçları PI-AAA ile ayrı ayrı kıyaslanmıştır. BAAA'nın iki versiyonu için, ilgili çalışmada (Zhang ve ark., 2016) olduğu üzere, kesme kuvveti kontrol parametresi 2, enerji kaybı kontrol parametresi 0.3 ve adaptasyon oranı kontrol parametresi 0.5 olarak kullanılmıştır. Yine aynı çalışmada (Zhang ve ark., 2016) kullanıldığı gibi, eğrinin değişim eğilimi (changing trend of the curve) (τ) kontrol parametresi, *Tanh(x)* lojistik fonksiyonu için 1.5 ve *Sig(x)* lojistik fonksiyonu için 2 olarak kullanılmıştır. BPSO algoritması için, parçacık hızının üst limiti (V_{max}) değeri 6, parçacık hızının alt limiti (V_{min}) değeri -6 ve pozitif ivme sabitleri (c_1, c_2) 2 olarak seçilmiştir.

PI-AAA yöntemi için ise, AAA'nın temel versiyonunda (Uymaz ve ark., 2015b) olduğu üzere, enerji kaybı kontrol parametresi 0.3 ve adaptasyon oranı kontrol parametresi 0.5 olarak kullanılmıştır. Önerilen PI-AAA yöntemine özgü kontrol parametreleri önceki bölümde yapılan parametre analizinden elde edildiği üzere, C_1

parametresinin değeri 0.3, C_2 parametresinin değeri 0.3, C_3 parametresinin değeri 0.2 ve NNS parametresinin değeri 4 olarak seçilmiştir.

Bu koşullar altında, tüm algoritmalar her bir problem için 30 kez bağımsız olarak çalıştırılmış ve elde edilen sonuçlardan *ortalama değer, standart sapma, optimum değeri bulma sayısı* ve *Gap* değeri olarak Çizelge 4.12 – 4.15'de sunulmuştur. Çizelgelerdeki *optimum değeri bulma sayısı* satırları, 30 farklı çalıştırmadan kaç tane çalıştırmada optimum sonuca erişildiğini ifade etmektedir. Çizelgelerdeki *Gap* satırları, yöntemin 30 defa çalıştırma sonucunda elde ettiği ortalama değer ile problemin optimum değeri arasındaki oransal fazlalığı ifade etmekte ve Denklem 4.2 kullanılarak hesaplanmaktadır.

Yapılan karşılaştırmanın daha anlaşılır ve görünür hale gelmesi için, her bir problem için en iyi algoritmanın değerleri kalın yazı tipinde verilmiştir. Elde edilen sonuçların daha iyi incelenebilmesi adına, problemler boyutlarına göre *az boyutlu, orta boyutlu, büyük boyutlu ve çok büyük boyutlu* olmak üzere 4 farklı grupta incelenmiş ve Çizelge 4.12-4.15'de gösterilmiştir.

Çizelge 4.12. PI-AAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, az boyutlu Cap71, Cap72, Cap73 ve Cap74 problemleri üzerinde karşılaştırılması

Yöntemler		Cap71	Cap72	Cap73	Cap74
GA-SP	Ortalama	**932615.750**	**977799.400**	1011314.476	**1034976.975**
	Gap	**0.00000**	**0.00000**	0.06659	**0.00000**
	Opt.Say.	**30**	**30**	19	**30**
	Std.Sap.	**0.000**	**0.000**	899.650	**0.000**
GA-TP	Ortalama	**932615.750**	**977799.400**	1011130.923	**1034976.975**
	Gap	**0.00000**	**0.00000**	0.04843	**0.00000**
	Opt.Say.	**30**	**30**	22	**30**
	Std.Sap.	**0.000**	**0.000**	825.576	**0.000**
GA-UP	Ortalama	**932615.750**	**977799.400**	1011069.739	**1034976.975**
	Gap	**0.00000**	**0.00000**	0.04238	**0.00000**
	Opt.Say.	**30**	**30**	23	**30**
	Std.Sap.	**0.000**	**0.000**	789.612	**0.000**
BAAA-Tanh	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
BAAA-Sig	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
BPSO	Ortalama	**932615.750**	**977799.400**	1010886.187	1035068.312
	Gap	**0.00000**	**0.00000**	0.02422	0.00882
	Opt.Say.	**30**	**30**	26	29
	Std.Sap.	**0.000**	**0.000**	634.625	500.272
EDA	Ortalama	933825.019	978689.053	1011579.868	1036909.799
	Gap	0.12966	0.09099	0.09285	0.18675
	Opt.Say.	20	16	13	18
	Std.Sap.	2352.759	1389.409	1292.617	3109.633
UMDA	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
PI-AAA	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**

Çizelge 4.13. PI-AAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, orta boyutlu Cap101, Cap102, Cap103 ve Cap104 problemleri üzerinde karşılaştırılması

Yöntemler		Cap101	Cap102	Cap103	Cap104
GA-SP	Ortalama	797193.286	**854704.200**	894351.782	**928941.750**
	Gap	0.06839	**0.00000**	0.06374	**0.00000**
	Opt.Say.	11	**30**	6	**30**
	Std.Sap.	421.655	**0.000**	505.036	**0.000**
GA-TP	Ortalama	797164.610	**854704.200**	894329.179	**928941.750**
	Gap	0.06479	**0.00000**	0.06121	**0.00000**
	Opt.Say.	12	**30**	10	**30**
	Std.Sap.	428.658	**0.000**	540.160	**0.000**
GA-UP	Ortalama	797107.258	**854704.200**	894427.382	**928941.750**
	Gap	0.05759	**0.00000**	0.07220	**0.00000**
	Opt.Say.	14	**30**	9	**30**
	Std.Sap.	436.524	**0.000**	522.784	**0.000**
BAAA-Tanh	Ortalama	796677.114	**854704.200**	**893782.113**	**928941.750**
	Gap	0.00360	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	29	**30**	**30**	**30**
	Std.Sap.	157.066	**0.000**	**0.000**	**0.000**
BAAA-Sig	Ortalama	**796648.438**	**854704.200**	**893782.113**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
BPSO	Ortalama	796992.553	854788.703	894223.572	929318.098
	Gap	0.04320	0.00989	0.04939	0.04051
	Opt.Say.	18	28	14	28
	Std.Sap.	428.658	321.588	521.237	1432.239
EDA	Ortalama	798133.621	856513.326	895530.313	930854.017
	Gap	0.18643	0.21167	0.19560	0.20585
	Opt.Say.	8	4	6	19
	Std.Sap.	1430.188	1612.638	1703.268	2982.045
UMDA	Ortalama	**796648.438**	**854704.200**	**893782.113**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
PI-AAA	Ortalama	**796648.438**	**854704.200**	**893782.113**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**

Çizelge 4.14. PI-AAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, büyük boyutlu Cap131, Cap132, Cap133 ve Cap134 problemleri üzerinde karşılaştırılması

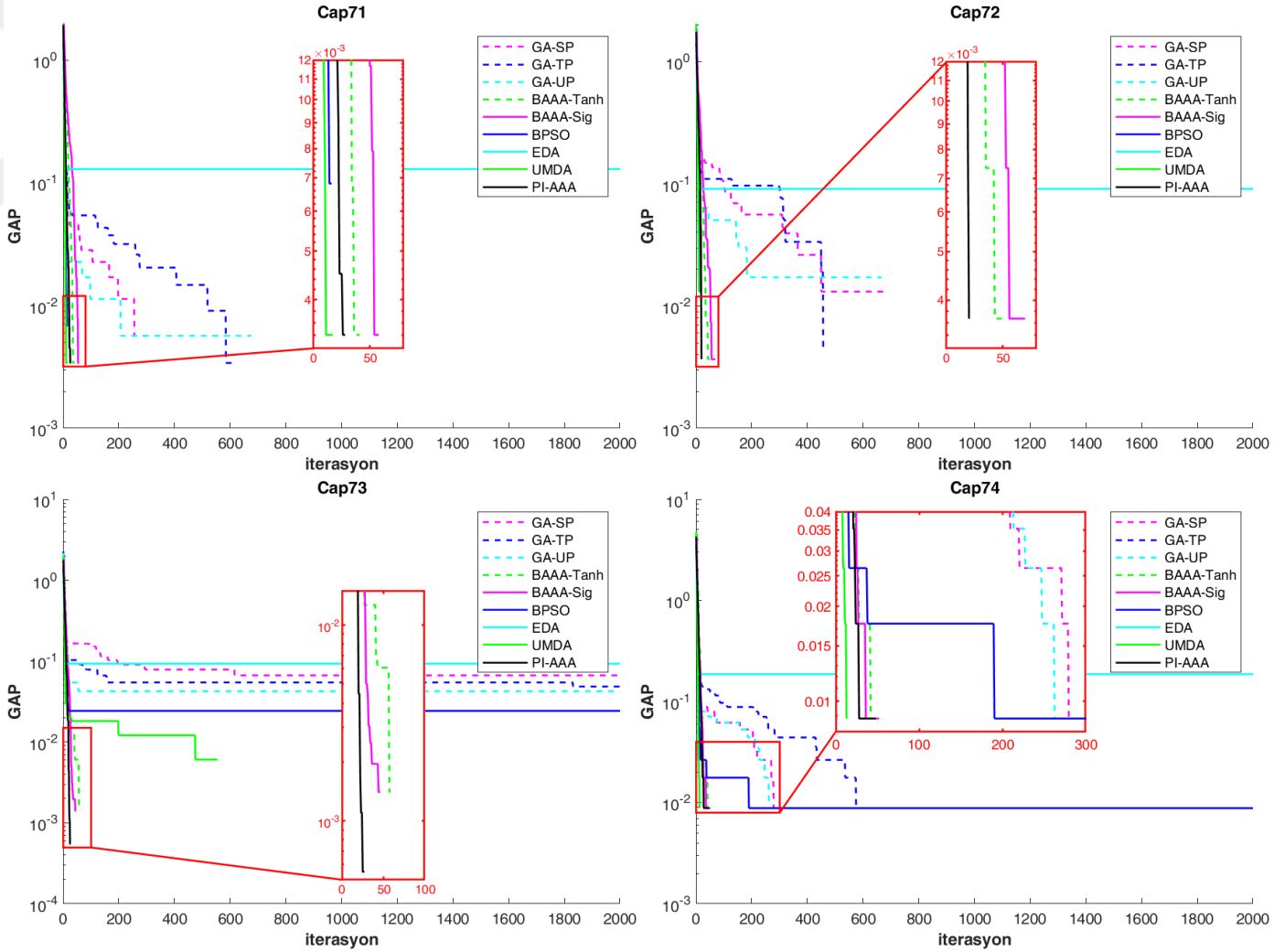
Yöntemler		Cap131	Cap132	Cap133	Cap134
GA-SP	Ortalama	793980.104	**851495.325**	893891.911	**928941.750**
	Gap	0.06813	**0.00000**	0.09128	**0.00000**
	Opt.Say.	16	**30**	10	**30**
	Std.Sap.	720.877	**0.000**	685.076	**0.000**
GA-TP	Ortalama	794012.905	**851495.325**	893740.954	**928941.750**
	Gap	0.07226	**0.00000**	0.07438	**0.00000**
	Opt.Say.	14	**30**	12	**30**
	Std.Sap.	690.560	**0.000**	655.920	**0.000**
GA-UP	Ortalama	793865.023	851517.200	893808.891	**928941.750**
	Gap	0.05362	0.00257	0.08198	**0.00000**
	Opt.Say.	15	29	9	**30**
	Std.Sap.	433.467	119.817	628.654	**0.000**
BAAA-Tanh	Ortalama	793525.591	**851495.325**	893333.515	**928941.750**
	Gap	0.01084	**0.00000**	0.02875	**0.00000**
	Opt.Say.	27	**30**	16	**30**
	Std.Sap.	262.498	**0.000**	324.451	**0.000**
BAAA-Sig	Ortalama	**793439.563**	**851495.325**	**893076.713**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
BPSO	Ortalama	794797.761	851991.551	893816.653	930756.565
	Gap	0.17118	0.05828	0.08285	0.19536
	Opt.Say.	10	21	10	18
	Std.Sap.	1505.749	1055.238	690.192	2594.211
EDA	Ortalama	798808.514	857160.840	897995.392	937037.233
	Gap	0.67667	0.66536	0.55076	0.87147
	Opt.Say.	0	1	1	0
	Std.Sap.	3332.123	2707.176	4801.182	6427.803
UMDA	Ortalama	793726.325	**851495.325**	893544.376	**928941.750**
	Gap	0.03614	**0.00000**	0.05237	**0.00000**
	Opt.Say.	20	**30**	12	**30**
	Std.Sap.	412.476	**0.000**	432.010	**0.000**
PI-AAA	Ortalama	**793439.563**	**851495.325**	**893076.713**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**

Çizelge 4.15. PI-AAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, çok büyük boyutlu CapA, CapB ve CapC problemleri üzerinde karşılaştırılması

Yöntemler		CapA	CapB	CapC
GA-SP	Ortalama	17164354.456	13054858.045	11586692.969
	Gap	0.04605	0.58391	0.70486
	Opt.Say.	24	9	2
	Std.Sap.	22451.206	66658.649	51848.248
GA-TP	Ortalama	17205089.145	13063527.186	11577797.524
	Gap	0.28348	0.65071	0.62755
	Opt.Say.	24	11	0
	Std.Sap.	139690.216	89122.485	46346.052
GA-UP	Ortalama	17166811.915	13107633.077	11578600.532
	Gap	0.06037	0.99053	0.63453
	Opt.Say.	24	3	0
	Std.Sap.	35181.974	79714.021	57031.219
BAAA-Tanh	Ortalama	17471223.794	13153617.764	11676427.752
	Gap	1.83470	1.34483	1.48479
	Opt.Say.	3	0	0
	Std.Sap.	225123.921	73978.543	101438.607
BAAA-Sig	Ortalama	17210900.533	13093705.559	11583462.068
	Gap	0.31735	0.88322	0.67678
	Opt.Say.	16	1	1
	Std.Sap.	90743.456	62168.803	45788.678
BPSO	Ortalama	17446511.870	13161205.473	11692212.797
	Gap	1.69066	1.40329	1.62198
	Opt.Say.	8	5	1
	Std.Sap.	319855.431	135326.728	115156.444
EDA	Ortalama	18942134.946	13743260.211	12109315.872
	Gap	10.40821	5.88785	5.24720
	Opt.Say.	0	0	0
	Std.Sap.	889476.329	368999.009	215429.705
UMDA	Ortalama	17181138.265	13058325.139	11560567.454
	Gap	0.14387	0.61063	0.47779
	Opt.Say.	23	7	0
	Std.Sap.	65899.979	60924.430	41444.853
PI-AAA	Ortalama	**17156454.478**	**13023608.901**	**11534648.849**
	Gap	**0.00000**	**0.34315**	**0.25253**
	Opt.Say.	**30**	**12**	**1**
	Std.Sap.	**0.000**	**50495.975**	**22746.218**

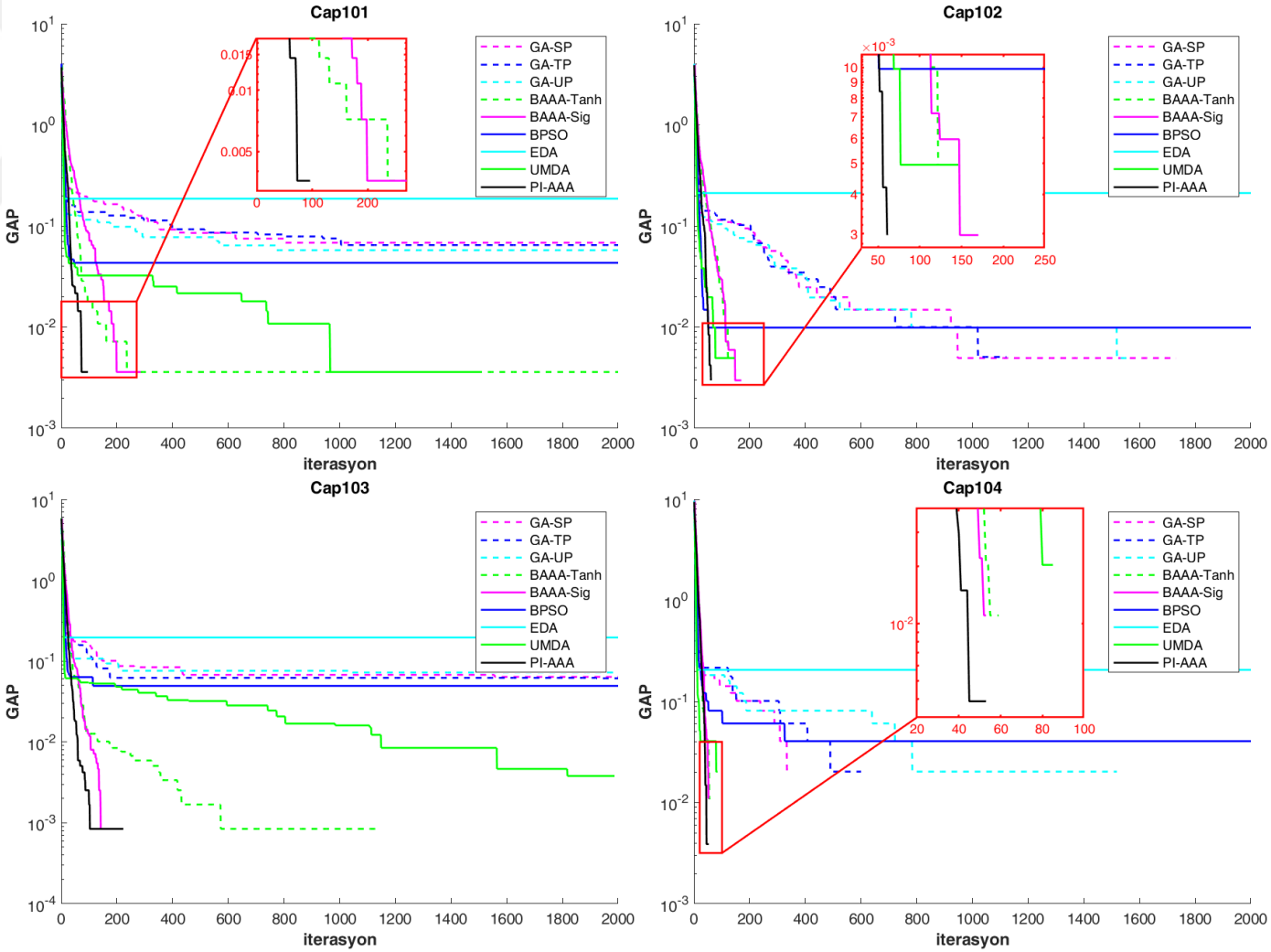
Çizelge 4.12-4.15'de görüldüğü üzere, EDA yöntemi herhangi bir problemde optimal değere ulaşmamıştır. EDA haricindeki tüm algoritmalar, her çalıştırmada Cap71 ve Cap72 problemlerinde optimum çözüme ulaşmışlardır. BPSO algoritması bu problemler haricinde hiçbir problemde bütün çalıştırmalarda optimum sonuca erişememiştir. UMDA yöntemi, *az boyutlu* ve *orta boyutlu* problemlerde 30 çalıştırmanın hepsinde optimal değerler elde etmesine rağmen, *çok büyük boyutlu* problemlerde 30 çalıştırmanın hepsinde optimal değerlere ulaşamamaktadır. Genetik algoritmanın 3 versiyonu da, her çalışmada Cap74, Cap102, Cap104, Cap132 ve Cap134 problemlerinde optimum çözümlere ulaşmıştır. Fakat problemlerin geri kalanında, GA versiyonları her çalışmada optimum çözümleri elde edememiştir. Bu nedenle, GA versiyonları ve BPSO'nun performansları, problem karakteristiklerine ve ilklendirmeye karşı gürbüzlük (robustness) açısından karşılaştırılan algoritmalardan nispeten daha kötüdür. *Az boyutlu, orta boyutlu* ve *büyük boyutlu* problemlerin sonuçlarına göre, BAAA-Tanh yöntemi, Cap101, Cap131 ve Cap133 problemleri dışındaki tüm problemlere 30 çalıştırmanın hepsinde optimum çözümler üretebilmiştir. BAAA-Sig yöntemi ise *çok büyük boyutlu* problemler haricindeki bütün problemlerde *optimum değeri bulma sayısı* olarak 30 değerini elde etmiştir. Bu nedenle sigmoid fonksiyonunun, bu test problemleri üzerinde, sürekli çözümleri ikili değerlere transfer etmek için daha uygun bir transfer fonksiyonu olduğu görülmektedir. UFLP'nin çözümünde, BAAA-Sig yöntemi ile önerilen algoritma karşılaştırıldığında, popülasyona dayalı etki faktörünün ve ikili olarak yapılandırılmış çözüm uzayı üzerinde çalışmanın, transfer fonksiyonlarının kullanımından daha faydalı ve başarılı olduğu görülmektedir. CapA problemlerinde, sadece önerilen algoritma her çalışmada optimum çözümü üretmiştir. CapB ve CapC problemlerinde ise PI-AAA, diğer yöntemlere göre çok daha düşük GAP değerleri elde etmiştir. Ayrıca, problem farklılıklarını ve ilklendirme koşullarını (rastgele başlatma) düşündüğümüzde, çizelgelerdeki standart sapma değerleri temel alındığında, PI-AAA yöntemi, diğer algoritmalara göre daha gürbüzdür (robust).

Yöntemleri kıyaslamak için kullanılan bir başka karşılaştırma metriği ise algoritmaların yakınsama karakteristikleridir. Karşılaştırılan algoritmaların çözüme ulaşma hızlarının daha iyi değerlendirilebilmesi için her bir problem için üretilen yakınsama eğrileri Şekil 4.15-4.18'de sunulmuştur.



Şekil 4.15. PI-AAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, az boyutlu Cap71, Cap72, Cap73 ve Cap74 problemleri için ürettikleri yakınsama eğrileri

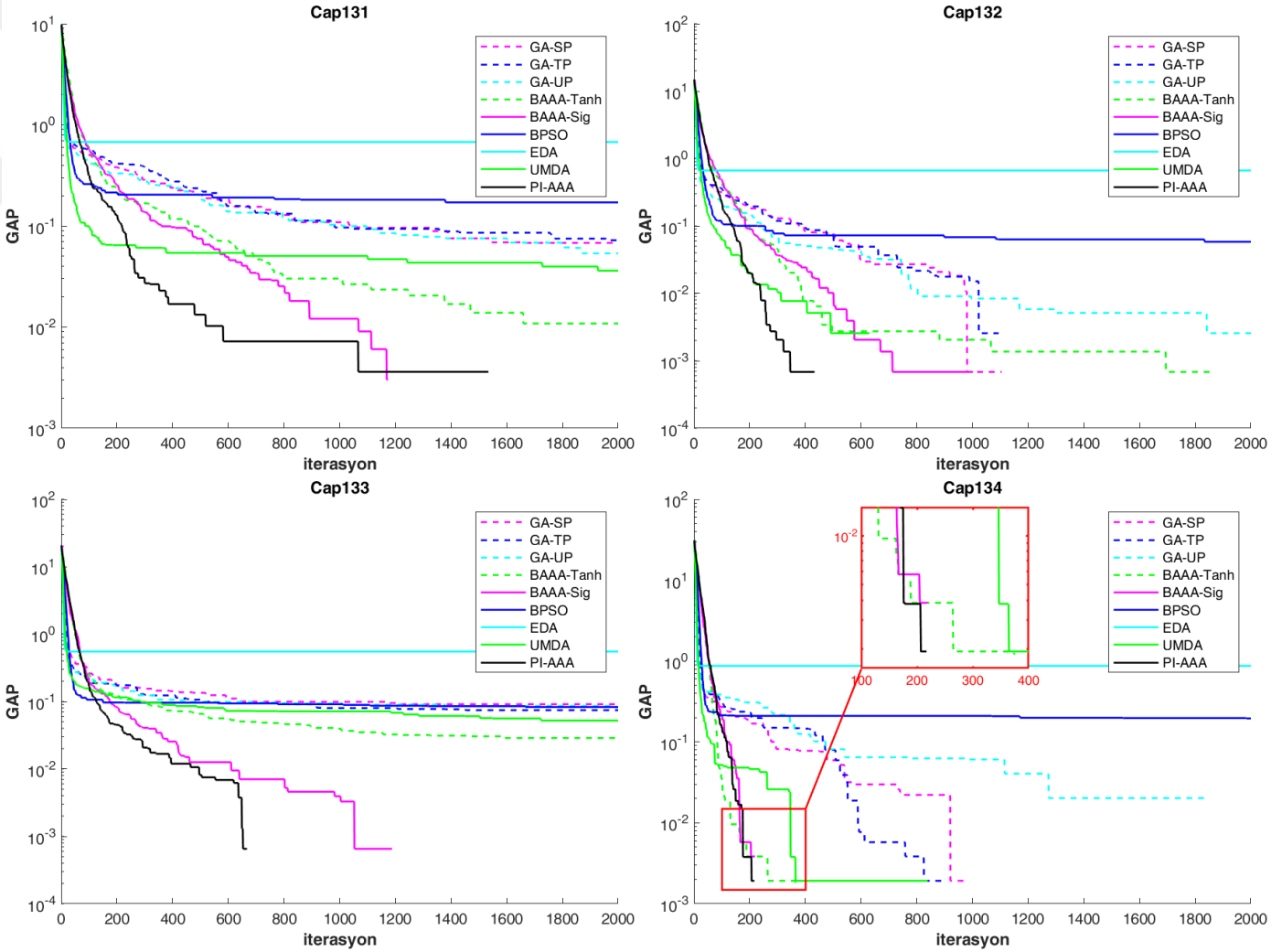
Şekil 4.15 incelendiğinde, PI-AAA, UMDA, BPSO, BAAA-Sig ve BAAA-Tanh yöntemleri Cap71 ve Cap72 problemlerinde ilk iterasyonlarda optimal değerlere hızla ulaştığı görülmektedir. Cap71 ve Cap72 problemlerinde, BPSO algoritması optimum değere diğer algoritmalardan daha hızlı yakınsamıştır. Genetik algoritmanın versiyonları (GA-SP, GA-TP ve GA-UP) Cap71, Cap72 ve Cap74 problemlerinde yaklaşık 500 iterasyon sonunda problemlerin optimum değerlerini bulabilmekte, fakat Cap73 probleminde ilk iterasyonlarda çözümleri iyileştirebilmesine rağmen, ilerleyen iterasyonlarda daha iyi çözümler üretememekte ve optimum karar değişkenlerini bulamamaktadır. EDA yöntemi, *az boyutlu* problemlerin hepsinde tüm iterasyon boyunca en uygun değeri elde edememektedir. Genetik algoritmanın versiyonlarında olduğu gibi, BPSO yöntemi de Cap73 probleminde optimum sonuca erişememiş, UMDA yöntemi ise yaklaşık 600 iterasyon çalıştıktan sonra optimum karar değişkenlerini bulabilmiştir. Cap74 probleminde, PI-AAA, UMDA, BAAA-Sig ve BAAA-Tanh yöntemleri, iterasyonların başında optimal değerlere erişmiştir. Önerilen PI-AAA yöntemi *az boyutlu* problemlerin hepsinde iterasyonların başlarında optimal karar değişkenlerini bulmayı başarmıştır.



Şekil 4.16. PI-AAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, orta boyutlu Cap101, Cap102, Cap103 ve Cap104 problemleri için ürettikleri yakınsama eğrileri

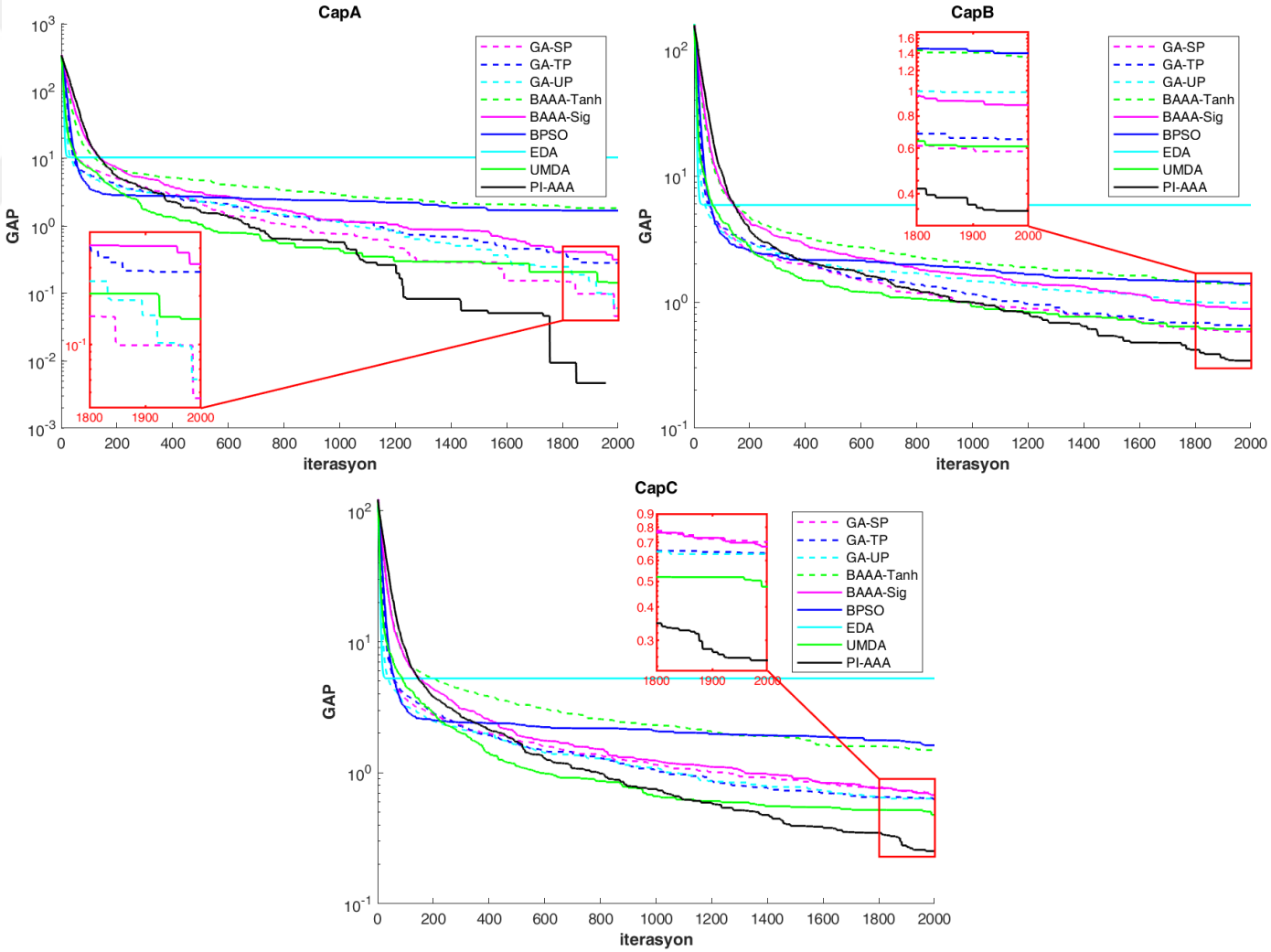
Şekil 4.16 incelendiğinde, Cap101, Cap102, Cap103 ve Cap104 problemlerinde, EDA ve BPSO yöntemlerinin, yerel minimum noktalarına yakınsadıkları ve en uygun değere ulaşmadıkları görülmektedir. GA versiyonları (GA-SP, GA-TP ve GA-UP), Cap101 ve Cap103 problemlerinde, iterasyon süreci boyunca optimal sonuca ulaşamamış, Cap102 ve Cap104 problemlerinde ise, toplam iterasyonun yaklaşık olarak orta kısımlarında optimum çözüme ulaşabilmişlerdir.

PI-AAA ve BAAA-Sig yöntemleri ise, orta boyutlu problemlerin tümünde iterasyonların erken evresinde optimal değerlere ulaşmışlardır.



Şekil 4.17. PI-AAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, büyük boyutlu Cap131, Cap132, Cap133 ve Cap134 problemleri için ürettikleri yakınsama eğrileri

Şekil 4.17 incelendiğinde, PI-AAA, UMDA, BAAA-Tan ve BAAA-Sig yöntemlerinin büyük boyutlu problemlerdeki (Cap131, Cap132, Cap133 ve Cap134) yakınsama hızları genel olarak diğer yöntemlerin yakınsama hızından daha iyidir. EDA ve BPSO yöntemlerinin optimal değerlere ulaşmada zorlandıkları görülmektedir. Genetik algoritma versiyonlarının yakınsama hızları çok yavaştır. Cap133 probleminde, sadece PI-AAA ve BAAA-Sig yöntemleri en uygun değere ulaşmıştır.



Şekil 4.18. PI-AAA yöntemi ile ikili olarak yapılandırılmış optimizasyon algoritmalarının, çok büyük boyutlu CapA, CapB ve CapC problemleri için ürettikleri yakınsama eğrileri

Şekil 4.18 incelendiğinde, CapB ve CapC problemlerinde hiçbir yöntemin optimal değere ulaşamadığı görülmektedir. CapA probleminde, sadece PI-AAA yöntemi, diğer yöntemlere göre daha hızlı bir yakınsama gerçekleştirerek optimal değere ulaşmıştır. CapB ve CapC probleminde, EDA yöntemi hariç tüm yöntemler benzer bir yakınsama karakterine sahip olup, önerilen PI-AAA yöntemi optimum değere en yakın çözüm üreten yöntem olmuştur.

UFLP test setinde yer alan 15 problemin çözümlerine genel olarak bakıldığında, PI-AAA yönteminin yakınsama karakteristiğinin karşılaştırılan diğer algoritmalara göre çok daha iyi olduğu görülmektedir.

UFLP test seti ile yapılan ikinci karşılaştırmada ise, CPSO, ABCbin, DisDE, binDE, DisABC ve binABC'nin literatürden elde edilen deneysel sonuçları, PI-AAA yöntemi ile *Gap* değerleri açısından karşılaştırılmış ve bu Gap değerlerine göre bir sıralama gerçekleştirilmiştir. CPSO, DisDE ve binDE yöntemlerinin sonuçları Kashan ve ark. (2013) tarafından yapılan çalışmadan, DisABC ve binABC yöntemlerinin sonuçları Kiran ve Gunduz (2013) tarafından yapılan çalışmadan ve ABCbin yönteminin sonuçları ise Kiran (2015a) tarafından yapılan çalışmadan direkt olarak alınmış ve Çizelge 4.16'da verilmiştir.

Çizelge 4.16. PI-AAA yöntemi ile yakın zamanda önerilmiş, ikili olarak yapılandırılmış optimizasyon algoritmalarının, UFLP test seti üzerinde karşılaştırılması

Problemler	CPSO		ABCbin		DisDE		binDE		DisABC		binABC		PI-AAA	
	Gap	Sıra	Gap	Sıra	Gap	Sıra	Gap	Sıra	Gap	Sıra	Gap	Sıra	Gap	Sıra
Cap71	5.0E-02	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap72	7.0E-02	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap73	6.0E-02	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap74	7.0E-02	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap101	1.4E-01	3	**0.0E+00**	**1**	7.2E-03	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap102	1.5E-01	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap103	1.6E-01	4	5.1E-03	3	8.4E-04	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap104	1.8E-01	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap131	7.5E-01	5	2.0E-01	3	**0.0E+00**	**1**	3.6E-03	2	6.2E-01	4	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap132	7.8E-01	5	2.0E-02	3	**0.0E+00**	**1**	5.0E-03	2	9.5E-02	4	**0.0E+00**	**1**	**0.0E+00**	**1**
Cap133	7.3E-01	7	7.5E-02	5	1.5E-02	3	1.4E-02	2	3.1E-02	4	1.2E-01	6	**0.0E+00**	**1**
Cap134	8.9E-01	2	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**	**0.0E+00**	**1**
CapA	2.2E+01	7	3.2E+00	6	3.7E-02	2	1.3E+00	4	1.5E-01	3	3.0E+00	5	**0.0E+00**	**1**
CapB	1.1E+01	7	2.8E+00	5	**6.7E-02**	**1**	1.5E+00	3	3.3E+00	6	2.5E+00	4	3.4E-01	2
CapC	9.7E+00	7	2.0E+00	4	**5.8E-02**	**1**	1.6E+00	3	4.7E+00	6	2.6E+00	5	2.5E-01	2
TR		59		36		20		25		36		31		**17**

Yapılan karşılaştırmanın daha anlaşılır ve görünür hale gelmesi için, Çizelge 4.16'da her bir problem için en iyi algoritmanın değerleri kalın yazı tipinde verilmiştir. Çizelgedeki TR (Toplam Sıra-Total Rank) ismindeki en son satır, toplam sıralamayı temsil etmekte ve algoritmanın her bir problemdeki sıralaması toplanarak elde edilmektedir.

Çizelge 4.16 incelendiğinde, PI-AAA yöntemi, 15 adet problemden 13 tanesinde birinci, kalan 2 problemde ise ikinci olmuştur. Sıralama toplamları açısından baktığımızda ise 17 toplam sıralama değeri ile birinci sırada yer almaktadır. 11 problemde birinci, 3 problemde ikici ve 1 problemde üçüncü olan DisDE algoritması ise sıralama toplamları açısından bakıldığında, 20 toplam sıralama değeri ile ikinci olmuştur. Karşılaştırmada en kötü performans ise, 59 toplam sıralama değeri ile hiçbir problemin çözümünde ilk sırada yer alamayan CPSO algoritmasına aittir.

Elde edilen verilerden açıkça görülmektedir ki, PI-AAA yöntemi, yakın zamanda önerilmiş ikili olarak yapılandırılmış optimizasyon algoritmaları ile kıyaslandığında, kaliteli sonuçlar üretebilen, rekabetçi ve problemin boyutu, karakteristiği ve ilklendirme şartlarına göre gürbüz (robust) bir algoritmadır.

PI-AAA yönteminin başarısını daha da kanıtlamak adına ikinci bir problem seti kullanılarak bir başka kıyaslama daha gerçekleştirilmiştir.

4.3.3.2 CEC2015 Problem Seti Kullanılarak Yapılan Karşılaştırmalar

Geliştirilen PI-AAA yöntemi için yapılan bir diğer deneysel çalışma CEC2015 (Chen ve ark., 2014) problem seti üzerinde gerçekleştirilmiştir. PI-AAA yöntemi bu problemler ile 30 kez çalıştırılmış, 30 farklı çalıştırmanın ortalama ve standart sapma değerleri hesaplanmış ve elde edilen sonuçlar yakın zamanda önerilmiş algoritmaların sonuçları ile karşılaştırılmıştır. SBHS (Kong ve ark., 2015b), BLDE (Chen ve ark., 2015), BHTPSO-QI (Beheshti ve ark., 2015), GBABC (Ozturk ve ark., 2015), BQIGSA (Nezamabadi-pour, 2015) ve SabDE (Banitalebi ve ark., 2016) algoritmalarının sonuçları kıyaslanmak amacı ile doğrudan Banitalebi ve ark. (2016) tarafından yapılan çalışmadan elde edilmiştir.

Algoritmaların adil bir şekilde karşılaştırılabilmesi için, PI-AAA yönteminin ortak kontrol parametreleri Banitalebi ve ark. (2016) tarafından yapılan çalışmaya uygun olarak seçilmiştir. Bu bağlamda, popülasyon büyüklüğü değeri 50 ve durdurma kriteri olarak MaxFEs değeri 100,000 olarak kullanılmıştır. Karar değişkenleri sürekli değerler

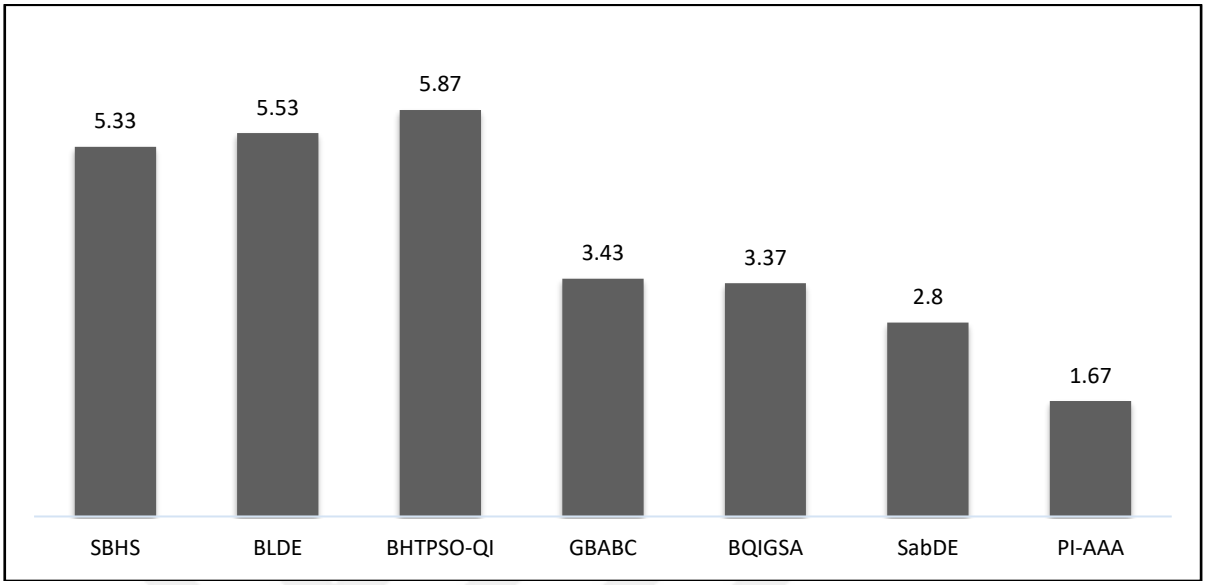
alan problemin boyutu (D) 10 olarak seçilmiş ve bu karar değişkenlerini ikili değerler ile ifade etmek amacı ile her boyut için 50 bit kullanılmıştır. Böylece popülasyondaki her bir birey ikili değerler alabilen 500 adet boyuta sahip olacaktır. Her bir bireyin uygunluk değerlerinin hesaplanabilmesi için 500 bitlik bu ikili değerlerin sürekli değerlere dönüştürülmesi gerekmektedir. Bu dönüşüm işlemi Denklem 3.13 kullanılarak hesaplanmaktadır. AAA'nın (Uymaz ve ark., 2015b) temel çalışmasında olduğu gibi, enerji kaybı parametresi değeri 0.3 ve adaptasyon parametresi değeri 0.5 olarak kullanılmıştır. PI-AAA yönteminde, algler ayrık çözüm uzayında çalıştığından dolayı temel AAA'da var olan kesme kuvveti parametresi kullanılmamaktadır. PI-AAA yöntemine özgü kontrol parametreleri önceki bölümlerde yapılan parametre analizinden elde edildiği üzere, C_1 parametresinin değeri 0.3, C_2 parametresinin değeri 0.3, C_3 parametresinin değeri 0.2 ve NNS parametresinin değeri 4 olarak seçilmiştir. Bu koşullar altında, elde edilen sonuçlar *ortalama değer* ve *standart sapma* değerleri ile Çizelge 4.17'de, Friedman (Derrac ve ark., 2011) sıra test sonuçları ise Şekil 4.19'da sunulmuştur. Çizelgede, her bir problem için en iyi sonucu üreten yöntemin değerleri kalın yazı tipinde verilmiştir.

Çizelge 4.17. PI-AAA yöntemi ile yakın zamanda önerilen algoritmaların CEC2015 test seti üzerinde karşılaştırılması

		SBHS	BLDE	BHTPSO-QI	GBABC	BQIGSA	SabDE	PI-AAA
F1	Ortalama	3.700E+08	1.087E+11	7.510E+08	2.729E+07	8.419E+07	3.093E+08	**1.804E+07**
	Std.Sap	2.271E+08	2.351E+11	2.203E+09	2.267E+07	7.354E+07	1.168E+08	**2.147E+07**
F2	Ortalama	4.493E+09	4.747E+09	5.132E+09	2.864E+09	7.834E+09	2.541E+09	**1.511E+04**
	Std.Sap	1.790E+06	3.209E+08	4.685E+08	2.374E+09	6.527E+09	5.008E+09	**5.039E+03**
F3	Ortalama	3.203E+02	3.201E+02	3.203E+02	3.202E+02	3.202E+02	3.200E+02	**3.063E+02**
	Std.Sap	7.013E-02	1.093E-02	7.426E-02	2.641E+02	2.641E+02	2.044E-02	**1.167E+00**
F4	Ortalama	4.478E+02	4.378E+02	4.478E+02	4.358E+02	4.389E+02	**4.116E+02**	5.145E+02
	Std.Sap	5.171E+00	7.754E+00	1.122E+01	3.599E+02	3.621E+02	**7.606E+00**	7.329E+01
F5	Ortalama	2.592E+03	1.934E+03	1.804E+03	1.108E+03	1.542E+03	9.330E+02	**5.004E+02**
	Std.Sap	1.959E+02	3.088E+02	3.926E+02	9.736E+02	1.275E+03	9.464E+01	**1.278E-01**
F6	Ortalama	5.312E+04	2.484E+06	6.799E+06	7.442E+06	5.582E+05	4.625E+04	**6.003E+02**
	Std.Sap	1.512E+04	7.369E+06	1.186E+07	1.321E+07	6.055E+05	4.076E+04	**5.926E-02**
F7	Ortalama	8.368E+02	8.472E+02	8.510E+02	7.589E+02	7.392E+02	7.752E+02	**7.003E+02**
	Std.Sap	7.981E+00	1.289E+01	1.694E+01	4.668E+02	4.324E+02	4.155E+03	**2.685E-01**
F8	Ortalama	3.344E+08	7.104E+09	5.407E+07	3.949E+07	2.948E+06	2.395E+07	**8.048E+02**
	Std.Sap	1.810E+09	5.651E+09	8.312E+07	2.442E+08	2.469E+06	5.432E+07	**2.012E+00**
F9	Ortalama	1.189E+03	1.190E+03	1.209E+03	1.017E+03	1.048E+03	1.177E+03	**9.033E+02**
	Std.Sap	5.210E+00	1.736E+01	2.050E+01	8.397E+02	8.643E+02	4.102E+01	**2.456E-01**
F10	Ortalama	9.038E+08	1.326E+10	1.320E+09	9.909E+05	4.426E+04	2.416E+07	**1.348E+04**
	Std.Sap	6.596E+08	1.724E+10	2.073E+09	3.659E+06	4.477E+04	8.862E+07	**1.390E+04**
F11	Ortalama	1.737E+03	1.599E+03	1.632E+03	1.159E+03	1.172E+03	1.114E+03	**1.105E+03**
	Std.Sap	1.661E+01	3.313E+01	3.371E+01	9.557E+02	9.669E+02	1.131E+01	**8.877E-01**
F12	Ortalama	1.445E+03	1.440E+03	1.437E+03	1.264E+03	1.255E+03	**1.224E+03**	1.252E+03
	Std.Sap	5.683E-01	3.247E+00	7.552E+00	1.044E+03	1.035E+03	**1.302E+00**	3.082E+01
F13	Ortalama	1.464E+10	2.882E+10	5.365E+08	**1.446E+03**	1.452E+03	2.815E+09	1.625E+03
	Std.Sap	9.749E+09	2.073E+10	2.294E+09	**1.193E+03**	1.197E+03	4.158E+09	3.952E+00
F14	Ortalama	2.736E+03	4.267E+03	5.005E+03	2.162E+03	3.356E+03	1.727E+03	**1.598E+03**
	Std.Sap	2.220E+02	2.569E+03	2.703E+03	2.091E+03	2.869E+03	4.411E+02	**4.199E+00**
F15	Ortalama	1.700E+03	1.700E+03	1.734E+03	2.012E+03	**1.530E+03**	1.700E+03	1.581E+03
	Std.Sap	1.552E-04	2.622E-05	1.851E+02	1.659E+03	**1.262E+03**	2.177E-05	1.223E+02

Sonuçlar incelendiğinde, PI-AAA yöntemi tek modlu (tek tepeli) fonksiyonların (unimodal functions) her ikisinde de (F1 ve F2) diğer yöntemlerden daha başarılıdır. F4 fonksiyonu hariç, basit çok tepeli fonksiyonlarda (simple multimodal functions) (F3-F9) PI-AAA yöntemi yine en iyi sonuçları üreterek ilk sırada yer almayı başarmıştır. F4 fonksiyonunda ise SabDE yöntemi PI-AAA yönteminden daha iyidir. 3 adet hibrit fonksiyonlardan (F10-F12) 2 tanesinde PI-AAA yöntemi ilk sırada yer alırken F12 fonksiyonunda ise SabDE yöntemi çok ufak bir fark ile PI-AAA yönteminin önüne geçmiştir. Son 3 fonksiyon olan kompozit fonksiyon türünde (F13-F15) ise 3 farklı yöntem başarılı sonuçlar üretmişlerdir. GBABC, PI-AAA ve BQIGSA yöntemleri, sırasıyla F13, F14 ve F15 fonksiyonlarında en iyi sonuçları elde etmişlerdir. CEC2015

problem seti sonuçlarını genel olarak incelediğimizde ise PI-AAA yöntemi 15 fonksiyondan 11 tanesinde, en iyi olmayı başarmıştır.



Şekil 4.19. Friedman sıra testi (rank test) sonuçlarına göre yöntemlerin karşılaştırılması

Şekil 4.19, deneysel çalışma için Friedman sıra testinin (rank test) sonuçlarını göstermektedir. Friedman'ın sıralama sonuçlarına göre, PI-AAA yöntemi ilk sırada yer almakta ve deneysel çalışmada diğer yöntemlerden daha başarılı olduğu görülmektedir.

4.4. Tez Kapsamında Geliştirilen AAA Tabanlı Yeni İkili Yaklaşımların Karşılaştırılması

Bu tez kapsamında, AAA yöntemi temel alınarak binAAA, SAAA ve PI-AAA isimleri ile 3 (üç) adet ikilileştirme yöntemi geliştirilmiştir. binAAA yöntemi yeni bir ikilileştirme operatörünü, SAAA yöntemi stigmerjik davranışı ve PI-AAA yöntemi popülasyon etkisini kullanmaktadır. Bu yöntemler önceki başlıklarda detaylı bir şekilde anlatılmış, literatürden elde edilen ve yeniden kodlanarak çalıştırılan algoritmalar ile kıyaslamaları gerçekleştirilmiştir. Bu bölümde ise bu tez kapsamında geliştirilmiş olan 3 adet yöntemin UFLP test seti üzerinde karşılaştırılması yapılmıştır.

Yöntemlerin adil bir şekilde karşılaştırılabilmesi için, yöntemlerin ortak kontrol parametreleri birbirlerine eşit olarak seçilmiştir. Bu bağlamda bütün yöntemlerde, popülasyon büyüklüğü 40 ve durdurma kriteri olarak MaxFEs değeri 80,000 olarak kullanılmıştır. AAA'nın (Uymaz ve ark., 2015b) temel çalışmasında olduğu gibi, enerji kaybı kontrol parametresi 0.3 ve adaptasyon oranı kontrol parametresi 0.5 olarak kullanılmıştır. AAA'nın temel versiyonunda kullanılan kesme kuvveti kontrol parametresi geliştirilen yöntemlerde kullanılmamıştır. binAAA yöntemine özgü NR kontrol parametresinin değeri 0.33 olarak kullanılmıştır. SAAA yöntemine özgü UMSP kontrol parametresinin değeri 0.5 ve DSP kontrol parametresinin değeri 0.66 olarak kullanılmıştır. PI-AAA yöntemine özgü kontrol parametrelerinin değerleri, önceki bölümde yapılan parametre analizinden elde edildiği üzere, C_1 parametresi için 0.3, C_2 parametresi için 0.3, C_3 parametresi için 0.2 ve NNS parametresi için 4 olarak seçilmiştir.

Her bir problem için tüm algoritmalar 30 kez bağımsız olarak çalıştırılmış ve elde edilen sonuçlardan *ortalama değer, standart sapma, optimum değeri bulma sayısı* ve *GAP* değeri olarak Çizelge 4.18 – 4.21'de sunulmuştur. Çizelgelerdeki *GAP* değerleri Denklem 4.2 kullanılarak hesaplanmaktadır:

Yapılan karşılaştırmanın daha anlaşılır ve görünür hale gelmesi adına, her bir problem için en iyi algoritmanın değerleri kalın yazı tipinde verilmiştir. Sonuçların daha iyi incelenebilmesi için, problemler boyutlarına göre *az boyutlu, orta boyutlu, büyük boyutlu ve çok büyük boyutlu* olmak üzere 4 farklı grupta incelenmiş ve Çizelge 4.18 – 4.21'de gösterilmiştir.

Çizelge 4.18. Tez kapsamında geliştirilen yöntemlerin, az boyutlu Cap71, Cap72, Cap73 ve Cap74 problemleri üzerinde karşılaştırılması

Yöntem		Cap71	Cap72	Cap73	Cap74
binAAA	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
SAAA	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
PI-AAA	Ortalama	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**

Çizelge 4.19. Tez kapsamında geliştirilen yöntemlerin, orta boyutlu Cap101, Cap102, Cap103 ve Cap104 problemleri üzerinde karşılaştırılması

Yöntem		Cap101	Cap102	Cap103	Cap104
binAAA	Ortalama	**796648.438**	**854704.200**	**893782.113**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
SAAA	Ortalama	**796648.438**	**854704.200**	**893782.113**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
PI-AAA	Ortalama	**796648.438**	**854704.200**	**893782.113**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**

Çizelge 4.20. Tez kapsamında geliştirilen yöntemlerin, büyük boyutlu Cap131, Cap132, Cap133 ve Cap134 problemleri üzerinde karşılaştırılması

Yöntem		Cap131	Cap132	Cap133	Cap134
binAAA	Ortalama	**793439.563**	**851495.325**	893082.539	**928941.750**
	Gap	**0.00000**	**0.00000**	0.00065	**0.00000**
	Opt.Say.	**30**	**30**	29	**30**
	Std.Sap.	**0.000**	**0.000**	31.914	**0.000**
SAAA	Ortalama	**793439.563**	**851495.325**	**893076.713**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**
PI-AAA	Ortalama	**793439.563**	**851495.325**	**893076.713**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Opt.Say.	**30**	**30**	**30**	**30**
	Std.Sap.	**0.000**	**0.000**	**0.000**	**0.000**

Çizelge 4.21. Tez kapsamında geliştirilen yöntemlerin, çok büyük boyutlu CapA, CapB ve CapC problemleri üzerinde karşılaştırılması

Yöntem		CapA	CapB	CapC
binAAA	Ortalama	17196684.161	13066644.044	11555763.878
	Gap	0.23449	0.67472	0.43604
	Opt.Say.	21	3	3
	Std.Sap.	78259.572	57572.956	43963.880
SAAA	Ortalama	**17156454.478**	**13011234.616**	11539496.443
	Gap	**0.00000**	**0.24781**	0.29466
	Opt.Say.	**30**	**15**	1
	Std.Sap.	**0.000**	**39224.744**	29766.311
PI-AAA	Ortalama	**17156454.478**	13023608.901	**11534648.849**
	Gap	**0.00000**	0.34315	**0.25253**
	Opt.Say.	**30**	12	**1**
	Std.Sap.	**0.000**	50495.975	**22746.218**

binAAA yöntemi *az boyutlu, orta boyutlu v*e *büyük boyutlu* problemlerden *Cap133* problemi hariç bütün problemlerde, 30 çalıştırmanın hepsinde en uygun çözümü bulmuştur. *Cap133* probleminde ise 29 çalıştırmada optimum değere erişmiş olup sadece 1 çalıştırmada erişememiştir. SAAA ve PI-AAA yöntemleri *az boyutlu, orta boyutlu v*e *büyük boyutlu* problemlerin tümünde 30 çalıştırmanın hepsinde en uygun çözümü bulmuştur. *Çok büyük boyutlu* problemler haricinde, önerilen algoritmalar çözüm kalitesi ve sağlamlık açısından benzer performans göstermektedirler. Yöntemlerin birbirleri ile

aralarındaki performans farklılıkları *çok büyük boyutlu* problemlerde ortaya çıkmaktadır. *Çok büyük boyutlu* problemlerden CapA probleminin sonuçları incelendiğinde, binAAA yöntemi 30 çalıştırmanın 21 tanesinde optimum sonuca ulaşmış ve 0.23449 Gap değerini elde etmiştir. SAAA ve PI-AAA yöntemleri ise 30 çalıştırmanın hepsinde en uygun çözüme erişmişlerdir. CapB probleminde, binAAA yöntemi 30 çalıştırmanın 3 tanesinde optimum sonuca ulaşmış ve 0.67472 Gap değerini elde etmiştir. PI-AAA yöntemi 30 çalıştırmanın 12 tanesinde optimum sonuca ulaşmış ve 0.34315 Gap değerini elde ederek CapB probleminin çözümünde ikinci en iyi Gap değerine sahip olmuştur. Geliştirilen yöntemler arasında CapB probleminde en iyi Gap değerine sahip yöntem SAAA olmuştur. SAAA yöntemi 0.24781 Gap değeri ile 30 çalıştırmanın 15 tanesinde optimum sonuca ulaşmıştır. CapC probleminde binAAA yöntemi 30 çalıştırmanın 3 tanesinde optimum sonuca ulaşmış ve 0.43604 Gap değerini elde etmiştir. Fakat SAAA ve PI-AAA yöntemleri 30 çalıştırmanın sadece 1 tanesinde optimum sonuca ulaşmalarına rağmen Gap değeri açısında bakıldığında binAAA yönteminden daha düşük değerler elde etmişlerdir. Bu sonuç *standart sapma* değerlerine bakıldığında da açıkça görülmektedir. binAAA yöntemi 3 defa optimum sonuca erişmiş fakat kalan 27 çalıştırmada SAAA ve PI-AAA yöntemlerine kıyasla CapC probleminin optimum değerinden çok uzak sonuçlar elde etmiştir. Bu da, binAAA yönteminin *standart sapma* değerlerinin SAAA ve PI-AAA yöntemlerine göre çok yüksek çıkmasına neden olmuştur. CapC probleminde en iyi sonuca 0.25253 Gap değeri ile PI-AAA yöntemi ulaşmıştır.

Genel olarak bakıldığında, CapA probleminde SAAA ve PI-AAA yöntemleri, CapB probleminde SAAA yöntemi ve CapC probleminde ise PI-AAA yöntemi en iyi sonuca ulaşmıştır. binAAA yöntemi ise önerilen 3 yöntem arasında en kötü performansı sergilemiştir.

5. SONUÇLAR VE ÖNERİLER

Son yıllarda optimizasyon problemlerinin çözümü için birçok yeni algoritma önerilmiştir. Bu algoritmalar genellikle doğadaki canlıların bireysel davranış şekillerinden, içgüdüsel hareketlerinden ve hatta birbirleri ile aralarındaki akıllı etkileşimlerinden (sürü zekâsı) esinlenilerek geliştirilmişlerdir. Önerilen algoritmalar genellikle belirli bir problem çeşidine veya karakteristiğine sahip problemleri çözecek şekilde tasarlanmaktadır. Daha sonra yapılan çeşitli uygulamalar neticesinde yönteme farklı karakteristikteki problemleri de çözebilme yeteneği kazandırılmaktadır. Örneğin ayrık optimizasyon problemlerinin çözümü için önerilen bir algoritma, farklı teknikler kullanılarak sürekli optimizasyon problemlerini de çözebilecek hale dönüştürülebilmektedir. Bunun tersine, karar değişkenleri sürekli değerler alan optimizasyon problemlerinin çözümü için önerilen bir algoritmaya, ikilileştirme yöntemleri uygulanarak karar değişkenleri ikili değerler alan optimizasyon problemlerini de çözme yeteneği kazandırılabilmektedir. Bu sayede yöntemin kendine has yeteneklerinin, problem karakteristiğinden bağımsız bir şekilde birçok optimizasyon problemlerine uygulanması sağlanabilmektedir.

Bu tez kapsamında, karar değişkenleri sürekli değerler alan problemler için önerilmiş bir algoritmanın, yeni ve özgün yöntemler geliştirilerek, karar değişkenleri ikili değerler alan problemleri de çözebilmesinin sağlanması amaçlanmıştır. Literatürdeki doğa esinli yöntemler incelendikten sonra, optimizasyon algoritması olarak yakın zamanda önerilmiş olan Yapay Alg Algoritması (AAA) tercih edilmiştir. Yapay Alg Algoritması, karar değişkenleri sürekli değerler alan problemlerin çözümü için 2015 yılında gerçekleştirilen doktora çalışmasında, doğada var olan mikroalglerin davranışlarından ilham alınarak geliştirilmiş ve literatüre kazandırılmış bir sürü zekâsı ilhamlı yöntemdir. AAA, geliştirilmesinden bugüne kadar geçen kısa zaman diliminde literatürde etkin bir yer edinmiştir.

Tez kapsamında geliştirilen ilk yöntemde, popülasyondaki alg kolonileri ikili değerler ile ilklendirilerek yeni aday çözümlerin elde edilebilmesi için helisel hareket fazı, ikili değerler ile çalışabilecek şekilde yeniden uyarlanmıştır. Helisel hareket fazında, seçilen komşu çözümün rastgele belirlenmiş üç adet boyutunun değerleri, belirli bir olasılığa bağlı olarak mantıksal değil (NOT) işlemine tabi tutularak aday çözüme kopyalanmaktadır. AAA'nın sürekli versiyonundaki adaptasyon sürecindeki işlemler,

sürekli çözüm uzayında tanımlanmış değerler üzerinde çalışmaktadır. Bu hali ile ikili değerler alan problemlerin çözümünde kullanılamadığından dolayı yöntemin Adaptasyon Süreci de yeniden uyarlanmıştır. AAA'nın temel versiyonunda, adaptasyon parametresi, her bir iterasyon için adaptasyon sürecinin işletilip işletilmemesine karar vermek için kullanılmaktadır. Geliştirilen binAAA yönteminde ise adaptasyon parametresi hem adaptasyon sürecinin işletilip işletilmemesine karar vermek için, hem de bu süreçte etkilenecek boyutların belirlenmesi için kullanılmıştır.

Geliştirilen binAAA yöntemi UFLP test seti üzerinde çalıştırılmış ve elde edilen sonuçlar binABC, BPSO, GA, DisABC, IBPSO ve ABCbin algoritmalarının sonuçlarıyla karşılaştırılmıştır. binAAA yöntemi ikili optimizasyon problemini çözme konusunda diğer algoritmalarla karşılaştırılabilir bir performans göstermiştir. Literatürde daha önce önerilmiş olan ikili AAA sürümü ve geliştirilen yöntem göz önüne alındığında, geliştirilen algoritma diğerinden daha iyi sonuçlar üretebilmiştir. Bunun nedeni, geliştirilen yöntemin ayrık çözüm uzayında çalışması, hem yerel hem de global aramada yetenekli bir arama stratejisine sahip olmasıdır. binAAA yöntemi, PSO tabanlı diğer yöntemler ile karşılaştırıldığında diğer yöntemler ile eşit veya daha iyi sonuçlar üretmektedir. binAAA ve ABC tabanlı algoritmalar düşük boyutlu problemlerde nispeten eşit performansa sahip olmakla birlikte önerilen algoritma CapA, CapB ve CapC gibi yüksek boyutlu problemleri çözmede ABC versiyonlarından daha iyi performans sergilemiştir. Genetik algoritmalar CapA ve CapB problemlerinde binAAA'dan kısmen daha iyi görünmekle birlikte CapC ve daha düşük boyutlu problemleri çözmede genetik algoritmaların sonuçları birçok algoritmadan daha kötüdür. Problemlerin çözümünde yerel arama yeteneğinin yanı sıra etkili küresel arama özelliğine de ihtiyaç duyulduğundan dolayı problemlerin boyutsallığı çok önem arz etmektedir. Ayrıca yerel ve global arama yeteneklerinin belirli bir dengede çalışması beklenmektedir. Genetik algoritmaların, global aramada karşılaştırılan diğer yöntemlere göre daha yetenekli olduğu görülmektedir. Ancak GA varyasyonları global ve yerel arama özelliğini farklı boyutlar için dengeleyememektedir. Bu da keşfedilen çözümler etrafında yoğunlaşamamasına (intensification) ve düşük boyutlarda binAAA yönteminden daha kötü sonuçlar üretmesine neden olmaktadır.

binAAA yöntemi, bu tezde saf ikili optimizasyon problemlerini çözmek için kullanılmıştır. Deneysel sonuçlar ve karşılaştırmalar incelendiğinde, geliştirilen algoritmanın, çalışmada ele alınan ikili optimizasyon problemlerindeki diğer algoritmalara kıyasla daha iyi veya eşit bir performans sergilediği görülmektedir.

Yakın gelecekte, binAAA algoritmasının parametrelerinin performansa etkilerinin analiz edilmesi ve özellik seçimi, sırt çantası problemi ve diğer ikili optimizasyon problemleri için uygulanması planlanmaktadır. Böylece yöntemin etkinliği daha da kanıtlanmış olacaktır.

İkili optimizasyon problemlerinin çözümü için, AAA algoritması temel alınarak önerilen ikinci yöntem iki farklı güncelleme mekanizması barındırmaktadır. Bu mekanizmanın ilki, aday çözüm üretmek için lojik tabanlı XOR operatörü kullanmakta, ikinci mekanizmada ise ilk mekanizmadan edinilen bilgi kullanılarak stigmerjik (stigmergic) davranış temelinde yeni çözümler üretilmektedir. SAAA yöntemi önceki binAAA yönteminde de olduğu üzere ikili değerler ile ilklendirilmektedir. Helisel hareket fazı lojik özel veya (xor) operatörü kullanılarak gerçekleştirilmiş ve "Güncelleme Mekanizması-1" olarak isimlendirilmiştir. Temel AAA algoritmasında olduğu üzere hareket rastgele belirlenen 3 (üç) adet boyut üzerinden gerçekleşmektedir. Güncelleme Mekanizması-1 adımının çalışmasından etkilenen ve yönteme eklenen iki yeni değişken, Güncelleme Mekanizması-2 adımında kullanılmakta ve yönteme stigmerjik davranış kazandırılmış olunmaktadır. Yeni aday çözümler, iki farklı güncelleme mekanizmasından da elde edilebilmektedir. Geliştirilen SAAA yönteminde de, binAAA yöntemindeki ile aynı değişiklik yapılarak, adaptasyon parametresi hem adaptasyon sürecinin işletilip işletilmemesine karar vermek için, hem de bu süreçte etkilenecek boyutların belirlenmesi için kullanılmıştır.

SAAA yönteminin performansı hem UFLP test seti hem de nümerik fonksiyonlar üzerinde araştırılmıştır. SAAA yöntemi UFLP üzerinde, BAAA yönteminin 2 farklı versiyonu, GA yöntemin 3 farklı versiyonu ve BPSO yöntemi ile kıyaslanmıştır. İkinci karşılaştırma için ise CEC2015 test seti kullanılmış ve geliştirilen yöntemin performansının değerlendirilebilmesi için SBHS, HS, BLDE, BHTPSO-QI, GBABC, BQIGSA ve SabDE yöntemleri ile kıyaslanmıştır. Karşılaştırmalarda 2 farklı test seti ve farklı karakteristiklere sahip birçok algoritma kullanılarak yöntemin gürbüzlüğü ortaya konulmaya çalışılmıştır. Bütün sonuçlar genel olarak incelendiğinde, geliştirilen algoritma sadece düşük boyutlu problemlerde değil, aynı zamanda yüksek boyutlu problemler için de dengeli bir keşif ve sömürü kabiliyeti sunmaktadır. Ayrıca SAAA yönteminin, çözüm kalitesi, yakınsama özellikleri ve sağlamlık açısından, araştırmada ele alınan ikili optimizasyon problemlerini çözmede etkili ve verimli bir algoritma olduğu gösterilmiştir.

Literatürden derlenecek olan farklı ikili optimizasyon problemlerini SAAA yöntemi ile çözmek yakın zamanda planlanan çalışmalar arasındadır. Ayrıca etkinliğini kanıtlamış ABC, GSA ve DE gibi algoritmalara da stigmerjik davranış şeklinin adapte edilmesi ve hatta stigmerjik davranışın her bir boyut için ayrı olarak uygulanması planlanmaktadır.

Tez çalışmasında üçüncü ve son olarak, AAA yönteminin ikili varyantı olan PI-AAA yöntemi geliştirilmiştir. Tez kapsamında geliştirilen diğer yöntemler ile paralel olarak PI-AAA yönteminde de ilklendirme ikili değerler ile yapılmaktadır. İkili hale getirme işlemi için bir popülasyon etkisi (population influence) yaklaşımı sunulmuş ve bu yaklaşım AAA'nın çalışmasıyla bütünleştirilmiştir. PI-AAA yönteminde, aday çözümler üretebilmek için, mevcut çözüm, en iyi çözüm ve alg kolonilerinden rastgele seçilen komşu çözümler kullanılmaktadır. Bu çözümler ile yapılan olasılık hesaplamaları neticesinde aday çözümler üretilmektedir. Olasılık değerleri PI-AAA algoritmasına yönteme özgü kontrol parametresi olarak tanımlanmıştır. Yöntemin performansı için kontrol parametrelerinin değerlerinin önemli olması nedeniyle, parametrelerin etkileri analiz edilmiş ve bu parametreler için en uygun değerin belirlenmesi amaçlanmıştır. Temel AAA'da yapılan bir diğer değişiklik ise, algoritmanın adaptasyon aşamasında yapılmıştır. Temel AAA'nın adaptasyon aşamasında sürekli değerler kullanılmakta ve sonuç olarak yine sürekli değerler üretilmektedir. Bu nedenle, adaptasyon aşaması yeniden düzenlenerek, karar değişkenleri ikili değerler alan bir popülasyon ile çalışabilmesi için uyarlanmıştır. Temel versiyonda, kesme kuvveti kontrol parametresi, sürekli değerler alan karar değişkenleri kullanarak yeni aday çözümler üretmek için kullanılmaktadır. PI-AAA yönteminde karar değişkenleri ikili değerler aldığından dolayı, yeni aday çözümler üretmek için kesme kuvveti kontrol parametresine ihtiyaç duyulmamıştır. Bu nedenle AAA'nın temel versiyonunda kullanılan kesme kuvveti kontrol parametresi PI-AAA yönteminde kullanılmamış ve kaldırılmıştır.

PI-AAA yönteminin performansı, ilk olarak UFLP üzerinde incelenmiştir. Literatürden 15 adet test problemi elde edilmiş ve bu problemler kullanılarak yöntemin parametre analizleri yapılmıştır. Analiz sonucunda elde edilen parametre değerleri kullanılarak PI-AAA yöntemi, ABC, GA, PSO, EDA vb. gibi modern evrimsel hesaplama ve sürü zekâsı algoritmaları ile karşılaştırılmıştır. PI-AAA yönteminin etkinliğini kanıtlamak adına yapılan ikinci karşılaştırma CEC2015 problem seti üzerinde gerçekleştirilmiştir. PI-AAA yönteminde, UFLP test seti için yapılan parametre analizinden elde edilen en iyi parametre değerleri kullanılmıştır. Elde edilen sonuçlar,

PI-AAA yönteminin karşılaştırmalarda daha iyi veya eşit performans gösterdiğini ve geliştirilen yaklaşımın rekabetçi bir ikili optimizasyon algoritması olduğunu göstermektedir.

Bu tez kapsamında geliştirilen PI-AAA yönteminin, gelecekte özellik seçimi ve sırt çantası problemleri gibi farklı ikili optimizasyon problemlerini çözebilecek şekilde uyarlanması planlanmaktadır. Geliştirilen popülasyon etki mekanizması bir algoritmanın özelliklerine bağlı olmadığından, bu mekanizmayı diğer popülasyon bazlı akıllı optimizasyon algoritmaları ile de entegre edilmesi düşünülmektedir. Ayrıca mevcut çözüm, en iyi çözüm ve rastgele seçilen komşu çözümlere ek olarak farklı karakteristiklere sahip çözümlerin de aday çözümler üretebilmek için kullanılması düşünülmektedir.

Sonuç olarak, tez kapsamında, AAA yöntemi temel alınarak binAAA, SAAA ve PI-AAA isimleri ile 3 (üç) adet ikilileştirme yöntemi geliştirilmiştir. Deneysel çalışmalardan elde edilen sonuçlar incelendiğinde, geliştirilen yöntemlerin çözüm kalitesi, standart sapma ve yakınsama özellikleri açısından ikili optimizasyon problemlerini çözmede alternatif, rekabetçi ve sağlam oldukları görülmektedir. Bu bağlamda gerçekleştirilen bu tez çalışması kapsamında, literatüre ikili optimizasyon alanında bir katkı sağlanmıştır.

KAYNAKLAR

Abbass, H. A., 2001, MBO: Marriage in honey bees optimization-A haplometrosis polygynous swarming approach, *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, 207-214.

Akay, B., 2009, Nümerik optimizasyon problemlerinde yapay arı kolonisi (artificial bee colony) algoritmasının performans analizi, Doktora Tezi, *Erciyes Üniversitesi Fen Bilimleri Enstitüsü*.

Al-Sultan, K. ve Al-Fawzan, M., 1999, A tabu search approach to the uncapacitated facility location problem, *Annals of Operations Research*, 86, 91-103.

Atkinson, J. ve Campos, D., 2016, Improving BCI-based emotion recognition by combining EEG feature selection and kernel classifiers, *Expert Systems with Applications*, 47, 35-41.

Azad, M., Kalam, A., Rocha, A. M. A. ve Fernandes, E. M. d. G., 2013, A simplified binary artificial fish swarm algorithm for uncapacitated facility location problems, *World Congress on Engineering 2013, WCE 2013*, 31-36.

Azad, M. A. K., Rocha, A. M. A. ve Fernandes, E. M., 2012, Solving multidimensional 0–1 knapsack problem with an artificial fish swarm algorithm, *International Conference on Computational Science and Its Applications*, 72-86.

Babaoğlu, İ., 2016, Utilization of Bat Algorithm for Solving Uncapacitated Facility Location Problem, In: Intelligent and Evolutionary Systems, Eds: Springer, p. 199-208.

Banitalebi, A., Abd Aziz, M. I. ve Aziz, Z. A., 2016, A self-adaptive binary differential evolution algorithm for large scale binary optimization problems, *Information Sciences*, 367, 487-511.

Bansal, J. C. ve Deep, K., 2012, A Modified Binary Particle Swarm Optimization for Knapsack Problems, *Applied Mathematics and Computation*, 218 (22), 11042-11061.

Beasley, J. E., 1990a, Or-Library - Distributing Test Problems by Electronic Mail, *Journal of the Operational Research Society*, 41 (11), 1069-1072.

Beasley, J. E., 1990b, A Lagrangian Heuristic for Set-Covering Problems, *Naval Research Logistics*, 37 (1), 151-164.

Beheshti, Z., Shamsuddin, S. M. ve Hasan, S., 2015, Memetic binary particle swarm optimization for discrete optimization problems, *Information Sciences*, 299, 58-84.

Beşkirli, M., Koç, İ., Haklı, H. ve Kodaz, H., 2018, A new optimization algorithm for solving wind turbine placement problem: Binary artificial algae algorithm, *Renewable Energy*, 121, 301-308.

Cebeci, S., 2012, Alt Küme Bulma Tabanlı Ayrık Optimizasyon Problemleri İçin Ayrık Parçacık Sürü Optimizasyonu Modelleri, Yüksek Lisans Tezi, *Gebze Yüksek Teknoloji Enstitüsü Mühendislik Ve Fen Bilimleri Enstitüsü*.

Cha, S.-H., Yoon, S. ve Tappert, C. C., 2005, Enhancing binary feature vector similarity measures.

Chen, Q., Liu, B., Zhang, Q., Liang, J., Suganthan, P. ve Qu, B., 2014, Problem definition and evaluation criteria for CEC 2015 special session and competition on bound constrained single-objective computationally expensive numerical optimization, *Computational Intelligence Laboratory, Zhengzhou University, China and Nanyang Technological University, Singapore, Tech. Rep*.

Chen, Y., Xie, W. ve Zou, X., 2015, A binary differential evolution algorithm learning from explored solutions, *Neurocomputing*, 149, 1038-1047.

Choi, S.-S., Cha, S.-H. ve Tappert, C. C., 2010, A survey of binary similarity and distance measures, *Journal of Systemics, Cybernetics and Informatics*, 8 (1), 43-48.

Cinar, A. C. ve Kiran, M. S., 2018, Similarity and logic gate-based tree-seed algorithms for binary optimization, *Computers & Industrial Engineering*, 115, 631-646.

Crina, G. ve Ajith, A., 2006, Stigmergic optimization: Inspiration, technologies and perspectives, In: Stigmergic optimization, Eds: Springer, p. 1-24.

Çelik, Y., 2013, Optimizasyon Problemlerinde Bal Arıları Evlilik Optimizasyonu Algoritmasının (Marriage In Honey Bee Optimization-MBO) Performansının Geliştirilmesi, Doktora Tezi, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*.

Deng, C., Zhao, B., Yang, Y., Peng, H. ve Wei, Q., 2011, Novel binary encoding differential evolution algorithm, *International Conference in Swarm Intelligence*, 416-423.

Derrac, J., Garcia, S., Molina, D. ve Herrera, F., 2011, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm and Evolutionary Computation*, 1 (1), 3-18.

Dorigo, M., 1992, Optimization, learning and natural algorithms, *PhD Thesis, Politecnico di Milano*.

Dorigo, M., Maniezzo, V. ve Colorni, A., 1996, Ant system: Optimization by a colony of cooperating agents, *Ieee Transactions on Systems Man and Cybernetics Part B-Cybernetics*, 26 (1), 29-41.

Drias, H., Sadeg, S. ve Yahi, S., 2005, Cooperative bees swarm for solving the maximum weighted satisfiability problem, *International Work-Conference on Artificial Neural Networks*, 318-325.

Eberhart, R. ve Kennedy, J., 1995, A new optimizer using particle swarm theory, *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, 39-43.

Engelbrecht, A. P. ve Pampara, G., 2007, Binary differential evolution strategies, *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 1942-1947.

Freville, A. ve Plateau, G., 1994, An Efficient Preprocessing Procedure for the Multidimensional 0-1-Knapsack Problem, *Discrete Applied Mathematics*, 49 (1-3), 189-212.

Geem, Z. W., Kim, J. H. ve Loganathan, G. V., 2001, A new heuristic optimization algorithm: harmony search, *simulation*, 76 (2), 60-68.

Ghosh, D., 2003, Neighborhood search heuristics for the uncapacitated facility location problem, *European Journal of Operational Research*, 150 (1), 150-162.

Glover, F. ve Laguna, M., 1998, Tabu search, In: Handbook of combinatorial optimization, Eds: Springer, p. 2093-2229.

Gortázar, F., Duarte, A., Laguna, M. ve Martí, R., 2010, Black box scatter search for general classes of binary optimization problems, *Computers & Operations Research*, 37 (11), 1977-1986.

Graham, L. ve Wilcox, L., 2000, Algae–Prentice Hall, *Upper Saddle River, New Jersey*.

Gunasundari, S., Janakiraman, S. ve Meenambal, S., 2016, Velocity Bounded Boolean Particle Swarm Optimization for improved feature selection in liver and kidney disease diagnosis, *Expert Systems with Applications*, 56, 28-47.

Guner, A. R. ve Sevkli, M., 2008, A discrete particle swarm optimization algorithm for uncapacitated facility location problem, *Journal of Artificial Evolution and Applications*, 2008.

Haddar, B., Khemakhem, M., Hanafi, S., Wilbaut, C. ve Chabchoub, H., 2013, A new Hybrid Heuristic for the 0-1 Knapsack Sharing Problem, *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (Ieee-Iesm 2013)*, 12-18.

Haddar, B., Khemakhem, M., Hanafi, S. ve Wilbaut, C., 2015, A hybrid heuristic for the 0-1 Knapsack Sharing Problem, *Expert Systems with Applications*, 42 (10), 4653-4666.

Hancer, E., Xue, B., Karaboga, D. ve Zhang, M. J., 2015, A binary ABC algorithm based on advanced similarity scheme for feature selection, *Applied Soft Computing*, 36, 334-348.

Hansen, N., Müller, S. D. ve Koumoutsakos, P., 2003, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), *Evolutionary computation*, 11 (1), 1-18.

Hashemi, M. ve Meybodi, M. R., 2011, Univariate Marginal Distribution Algorithm in Combination with Extremal Optimization (EO, GEO), *Neural Information Processing, Pt Ii*, 7063, 220-+.

Holland, J., 1975, Adaptation in natural and artificial systems: an introductory analysis with application to biology, *Control and artificial intelligence*.

Holland, J. H., 1992a, Genetic Algorithms, *Scientific American*, 267 (1), 66-72.

Holland, J. H., 1992b, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT press, p.

James, R. J. W. ve Nakagawa, Y., 2005, Enumeration methods for repeatedly solving multidimensional knapsack sub-problems, *Ieice Transactions on Information and Systems*, E88d (10), 2329-2340.

Jaramillo, J. H., Bhadury, J. ve Batta, R., 2002, On the use of genetic algorithms to solve location problems, *Computers & Operations Research*, 29 (6), 761-779.

Jia, D., Duan, X. ve Khan, M. K., 2014, Binary Artificial Bee Colony optimization using bitwise operation, *Computers & Industrial Engineering*, 76, 360-365.

John, R., 1992, Koza:"Genetic Programming", on the programming of computers by means of natural selection, MIT Press.

Kamboj, V. K., 2016, A novel hybrid PSO-GWO approach for unit commitment problem, *Neural Computing & Applications*, 27 (6), 1643-1655.

Karaboga, D., 2005, An idea based on honey bee swarm for numerical optimization, *Technical report-tr06, Erciyes university, engineering faculty, computer …*.

Karaboga, D. ve Basturk, B., 2007, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *Journal of Global Optimization*, 39 (3), 459-471.

Karaboğa, D., 2014, Yapay zeka optimizasyon algoritmaları, Nobel Akademi Yayıncılık, p.

Kashan, M. H., Nahavandi, N. ve Kashan, A. H., 2012, DisABC: a new artificial bee colony algorithm for binary optimization, *Applied Soft Computing*, 12 (1), 342-352.

Kashan, M. H., Kashan, A. H. ve Nahavandi, N., 2013, A novel differential evolution algorithm for binary optimization, *Computational Optimization and Applications*, 55 (2), 481-513.

Kennedy, J. ve Eberhart, R., 1995, Particle swarm optimization, *1995 Ieee International Conference on Neural Networks Proceedings, Vols 1-6*, 1942-1948.

Kennedy, J. ve Eberhart, R. C., 1997, A discrete binary version of the particle swarm algorithm, *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, 4104-4108.

Khanesar, M. A., Teshnehlab, M. ve Shoorehdeli, M. A., 2007, A novel binary particle swarm optimization, *Control & Automation, 2007. MED'07. Mediterranean Conference on*, 1-6.

Khazaei, P., Dabbaghjamanesh, M., Kalantarzadeh, A. ve Mousavi, H., 2016, Applying the Modified TLBO Algorithm to Solve the Unit Commitment Problem, *2016 World Automation Congress (Wac)*.

Kiran, M. S. ve Gunduz, M., 2013, XOR-based artificial bee colony algorithm for binary optimization, *Turkish Journal of Electrical Engineering and Computer Sciences*, 21, 2307-2328.

Kiran, M. S., 2015a, The continuous artificial bee colony algorithm for binary optimization, *Applied Soft Computing*, 33, 15-23.

Kiran, M. S., 2015b, TSA: Tree-seed algorithm for continuous optimization, *Expert Systems with Applications*, 42 (19), 6686-6698.

Kıran, M. S., 2014, Optimizasyon Problemlerinin Çözümü İçin Yapay Arı Kolonisi Algoritması Tabanlı Yeni Yaklaşımlar, Doktora Tezi, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*.

Kirkpatrick, S., Gelatt, C. D. ve Vecchi, M. P., 1983, Optimization by simulated annealing, *science*, 220 (4598), 671-680.

Kong, M., Tian, P. ve Kao, Y., 2008, A new ant colony optimization algorithm for the multidimensional knapsack problem, *Computers & Operations Research*, 35 (8), 2672-2683.

Kong, X., Gao, L., Ouyang, H. ve Li, S., 2015a, Solving large-scale multidimensional knapsack problems with a new binary harmony search algorithm, *Computers & Operations Research*, 63, 7-22.

Kong, X., Gao, L., Ouyang, H. ve Li, S., 2015b, A simplified binary harmony search algorithm for large scale 0–1 knapsack problems, *Expert Systems with Applications*, 42 (12), 5337-5355.

Korkmaz, S., Babalik, A. ve Kiran, M. S., 2018, An artificial algae algorithm for solving binary optimization problems, *International Journal of Machine Learning and Cybernetics*, 9 (7), 1233-1247.

Korkmaz, S. ve Kiran, M. S., 2018, An artificial algae algorithm with stigmergic behavior for binary optimization, *Applied Soft Computing*, 64, 627-640.

Köse, U., 2017, Yapay zeka tabanlı optimizasyon algoritmaları geliştirilmesi, Doktora Tezi, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*.

Krarup, J. ve Pruzan, P. M., 1983, The Simple Plant Location Problem - Survey and Synthesis, *European Journal of Operational Research*, 12 (1), 36-81.

Kratica, J., Tošic, D., Filipović, V. ve Ljubić, I., 2001, Solving the simple plant location problem by genetic algorithm, *RAIRO-Operations Research*, 35 (1), 127-142.

Krause, J., Cordeiro, J., Parpinelli, R. S. ve Lopes, H. S., 2013, A survey of swarm algorithms applied to discrete optimization problems, In: Swarm Intelligence and Bio-Inspired Computation, Eds: Elsevier, p. 169-191.

Laalaoui, Y. ve M'Hallah, R., 2016, A binary multiple knapsack model for single machine scheduling with machine unavailability, *Computers & Operations Research*, 72, 71-82.

Lalami, M. E. ve El-Baz, D., 2012, GPU Implementation of the Branch and Bound method for knapsack problems, *2012 Ieee 26th International Parallel and Distributed Processing Symposium Workshops & Phd Forum (Ipdpsw)*, 1769-1777.

Larrañaga, P. ve Lozano, J. A., 2001, Estimation of distribution algorithms: A new tool for evolutionary computation, Springer Science & Business Media, p.

Lee, S., Soak, S., Oh, S., Pedrycz, W. ve Jeon, M., 2008, Modified binary particle swarm optimization, *Progress in Natural Science*, 18 (9), 1161-1166.

Li, S., Jiang, T., Chen, H. Q., Shen, D. M., Todo, Y. ve Gao, S. C., 2016, Discrete Chaotic Gravitational Search Algorithm for Unit Commitment Problem, *Intelligent Computing Theories and Application, Icic 2016, Pt Ii*, 9772, 757-769.

Marandi, A., Afshinmanesh, F., Shahabadi, M. ve Bahrami, F., 2006, Boolean particle swarm optimization and its application to the design of a dual-band dual-polarized planar antenna, *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 3212-3218.

Michel, L. ve Van Hentenryck, P., 2004, A simple tabu search for warehouse location, *European Journal of Operational Research*, 157 (3), 576-591.

Mirjalili, S. ve Lewis, A., 2013, S-shaped versus V-shaped transfer functions for binary particle swarm optimization, *Swarm and Evolutionary Computation*, 9, 1-14.

Mirjalili, S., Mirjalili, S. M. ve Lewis, A., 2014a, Grey Wolf Optimizer, *Advances in Engineering Software*, 69, 46-61.

Mirjalili, S., Wang, G.-G. ve Coelho, L. d. S., 2014b, Binary optimization using hybrid particle swarm optimization and gravitational search algorithm, *Neural Computing and Applications*, 25 (6), 1423-1435.

Mirjalili, S. ve Lewis, A., 2016, The Whale Optimization Algorithm, *Advances in Engineering Software*, 95, 51-67.

Monabbati, E. ve Kakhki, H. T., 2015, On a class of subadditive duals for the uncapacitated facility location problem, *Applied Mathematics and Computation*, 251, 118-131.

Moosavian, N., 2015, Soccer league competition algorithm for solving knapsack problems, *Swarm and Evolutionary Computation*, 20, 14-22.

Mühlenbein, H., 1997, The equation for response to selection and its use for prediction, *Evolutionary Computation*, 5 (3), 303-346.

Nakrani, S. ve Tovey, C., 2004, On honey bees and dynamic server allocation in internet hosting centers, *Adaptive Behavior*, 12 (3-4), 223-240.

Nezamabadi-pour, H., Rostami-Shahrbabaki, M. ve Maghfoori-Farsangi, M., 2008, Binary particle swarm optimization: challenges and new solutions, *CSI J Comput Sci Eng*, 6 (1-A), 21-32.

Nezamabadi-pour, H., 2015, A quantum-inspired gravitational search algorithm for binary encoded optimization problems, *Engineering Applications of Artificial Intelligence*, 40, 62-75.

Omran, M. G. H., Salman, A. ve Engelbrecht, A. P., 2005, Self-adaptive differential evolution, *Computational Intelligence and Security, Pt 1, Proceedings*, 3801, 192-199.

Ozturk, C., Hancer, E. ve Karaboga, D., 2015, A novel binary artificial bee colony algorithm based on genetic operators, *Information Sciences*, 297, 154-170.

Özkan, S., 2010, Gezgin Satıcı Probleminin Çözümüne Yönelik Algoritmik Yaklaşımlar, Yüksek Lisans Tezi, *Gazi Üniversitesi Fen Bilimleri Enstitüsü*.

Özkış, A., 2018, Çok Amaçlı Problemlerin Çözümü İçin Yeni Sezgisel Algoritma Geliştirilmesi, Doktora Tezi, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*.

Pampara, G., Franken, N. ve Engelbrecht, A. P., 2005, Combining particle swarm optimisation with angle modulation to solve binary problems, *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 89-96.

Pampara, G., Engelbrecht, A. P. ve Franken, N., 2006, Binary differential evolution, *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 1873-1879.

Pampará, G. ve Engelbrecht, A. P., 2011, Binary artificial bee colony optimization, *Swarm Intelligence (SIS), 2011 IEEE symposium on*, 1-8.

Passino, K. M., 2002, Biomimicry of bacterial foraging for distributed optimization and control, *IEEE control systems*, 22 (3), 52-67.

Patvardhan, C., Bansal, S. ve Srivastav, A., 2016, Parallel improved quantum inspired evolutionary algorithm to solve large size Quadratic Knapsack Problems, *Swarm and Evolutionary Computation*, 26, 175-190.

Pavez-Lazo, B. ve Soto-Cartes, J., 2011, A deterministic annular crossover genetic algorithm optimisation for the unit commitment problem, *Expert Systems with Applications*, 38 (6), 6523-6529.

Pavithr, R. S. ve Gursaran, 2016, Quantum Inspired Social Evolution (QSE) algorithm for 0-1 knapsack problem, *Swarm and Evolutionary Computation*, 29, 33-46.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S. ve Zaidi, M., 2006, The bees algorithm—a novel tool for complex optimisation problems, In: Intelligent Production Machines and Systems, Eds: Elsevier, p. 454-459.

pinkturtleemily, 2013, Congratulations. You have now saved 0.3 seconds of your life, https://imgur.com/gallery/BcPI0E1: [Ziyaret Tarihi: 13 Şubat 2019].

Rashedi, E., Nezamabadi-Pour, H. ve Saryazdi, S., 2009, GSA: a gravitational search algorithm, *Information Sciences*, 179 (13), 2232-2248.

Rashedi, E., Nezamabadi-Pour, H. ve Saryazdi, S., 2010, BGSA: binary gravitational search algorithm, *Natural Computing*, 9 (3), 727-745.

Sadri, J. ve Suen, C. Y., 2006, A Genetic Binary Particle Swarm Optimization model, *2006 Ieee Congress on Evolutionary Computation, Vols 1-6*, 656-+.

Saha, S., Kole, A. ve Dey, K., 2011, A Modified Continuous Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem, In: Information Technology and Mobile Communication, Eds: Springer, p. 305-311.

Salman, A. A., Omran, M. G. ve Ahmad, I., 2015, Adaptive probabilistic harmony search for binary optimization problems, *Memetic Computing*, 7 (4), 291-316.

Saravanan, B., Kumar, C. ve Kothari, D. P., 2016, A solution to unit commitment problem using fire works algorithm, *International Journal of Electrical Power & Energy Systems*, 77, 221-227.

Sevkli, M. ve Guner, A. R., 2006, A continuous particle swarm optimization algorithm for uncapacitated facility location problem, *Ant Colony Optimization and Swarm Intelligence, Proceedings*, 4150, 316-323.

Shang, L., Zhou, Z. ve Liu, X., 2016, Particle swarm optimization-based feature selection in sentiment classification, *Soft Computing*, 20 (10), 3821-3834.

Shunmugapriya, P. ve Kanmani, S., 2017, A hybrid algorithm using ant and bee colony optimization for feature selection and classification (AC-ABC Hybrid), *Swarm and Evolutionary Computation*, 36, 27-36.

Singhal, P. K., Naresh, R. ve Sharma, V., 2015, A modified binary artificial bee colony algorithm for ramp rate constrained unit commitment problem, *International Transactions on Electrical Energy Systems*, 25 (12), 3472-3491.

Storn, R. ve Price, K., 1997, Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, 11 (4), 341-359.

Sun, M., 2006, Solving the uncapacitated facility location problem using tabu search, *Computers & Operations Research*, 33 (9), 2563-2589.

Sun, Y. J., Wang, F., Wang, B., Chen, Q. F., Engerer, N. A. ve Mi, Z. Q., 2017, Correlation Feature Selection and Mutual Information Theory Based Quantitative Research on Meteorological Impact Factors of Module Temperature for Solar Photovoltaic Systems, *Energies*, 10 (1).

Tan, F., Fu, X. Z., Zhang, Y. P. ve Bourgeois, A. G., 2008, A genetic algorithm-based method for feature subset selection, *Soft Computing*, 12 (2), 111-120.

Taormina, R. ve Chau, K. W., 2015, Data-driven input variable selection for rainfall-runoff modeling using binary-coded particle swarm optimization and Extreme Learning Machines, *Journal of Hydrology*, 529, 1617-1632.

Teodorovic, D. ve Dell'Orco, M., 2005, Bee colony optimization–a cooperative learning approach to complex transportation problems, *Advanced OR and AI methods in transportation*, 51, 60.

Tohyama, H., Ida, K. ve Matsueda, J., 2011, A Genetic Algorithm for the Uncapacitated Facility Location Problem, *Electronics and Communications in Japan*, 94 (5), 47-54.

Trivedi, A., Srinivasan, D., Biswas, S. ve Reindl, T., 2015, Hybridizing genetic algorithm with differential evolution for solving the unit commitment scheduling problem, *Swarm and Evolutionary Computation*, 23, 50-64.

Türk Dil Kurumu, 2011, Güncel Türkçe sözlük.

Uymaz, S. A., 2015, Yeni bir biyolojik ilhamlı metasezgisel optimizasyon metodu: Yapay alg algoritması, Doktora Tezi, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*.

Uymaz, S. A., Tezel, G. ve Yel, E., 2015a, Artificial algae algorithm with multi-light source for numerical optimization and applications, *Biosystems*, 138, 25-38.

Uymaz, S. A., Tezel, G. ve Yel, E., 2015b, Artificial algae algorithm (AAA) for nonlinear global optimization, *Applied Soft Computing*, 31, 153-171.

Wang, L., Fu, X., Menhas, M. I. ve Fei, M., 2010a, A modified binary differential evolution algorithm, In: Life System Modeling and Intelligent Computing, Eds: Springer, p. 49-57.

Wang, L., Xu, Y., Mao, Y. ve Fei, M., 2010b, A discrete harmony search algorithm, In: Life System Modeling and Intelligent Computing, Eds: Springer, p. 37-43.

Wedde, H. F., Farooq, M. ve Zhang, Y., 2004, BeeHive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior, *International Workshop on Ant Colony Optimization and Swarm Intelligence*, 83-94.

Wilcoxon, F., 1945, Individual Comparisons by Ranking Methods, *Biometrics Bulletin*, 1 (6), 80-83.

Wolpert, D. H. ve Macready, W. G., 1997, No free lunch theorems for optimization, *IEEE transactions on evolutionary computation*, 1 (1), 67-82.

Wolpert, D. H. ve Macready, W. G., 2005, Coevolutionary free lunches.

Yaman, F., 2014, Optimizasyon Problemlerinin Çözümünde Hesaplama Maliyetinin Azaltılması, Doktora Tezi, *Ankara Üniversitesi Fen Bilimleri Enstitüsü*.

Yanasse, H. H. ve Soma, N. Y., 1987, A New Enumeration Scheme for the Knapsack-Problem, *Discrete Applied Mathematics*, 18 (2), 235-245.

Yang, Q., 2008, A comparative study of discrete differential evolution on binary constraint satisfaction problems, *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, 330-335.

Yang, X.-S., 2005, Engineering optimizations via nature-inspired virtual bee algorithms, *International Work-Conference on the Interplay Between Natural and Artificial Computation*, 317-323.

Yang, X.-S., 2009, Firefly algorithms for multimodal optimization, *International symposium on stochastic algorithms*, 169-178.

Yang, X.-S. ve Deb, S., 2009, Cuckoo search via Lévy flights, *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 210-214.

Yang, X.-S., 2010a, Nature-inspired metaheuristic algorithms, Luniver press, p.

Yang, X.-S., 2010b, Engineering optimization: an introduction with metaheuristic applications, John Wiley & Sons, p.

Yang, X.-S., 2012, Flower pollination algorithm for global optimization, *International conference on unconventional computing and natural computation*, 240-249.

Yang, X.-S., 2014, Nature-inspired optimization algorithms, Elsevier, p.

Yang, X. S., 2010c, Firefly algorithm, stochastic test functions and design optimisation, *International Journal of Bio-Inspired Computation*, 2 (2), 78-84.

Yang, X. S., 2010d, A New Metaheuristic Bat-Inspired Algorithm, *Nicso 2010: Nature Inspired Cooperative Strategies for Optimization*, 284, 65-74.

Yuan, X. H., Nie, H., So, A. J., Wang, L. ve Yuan, Y. B., 2009, An improved binary particle swarm optimization for unit commitment problem, *Expert Systems with Applications*, 36 (4), 8049-8055.

Zhang, X. D., Wu, C. Z., Li, J., Wang, X. Y., Yang, Z. J., Lee, J. M. ve Jung, K. H., 2016, Binary artificial algae algorithm for multidimensional knapsack problems, *Applied Soft Computing*, 43, 583-595.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Sedat KORKMAZ
Uyruğu	**:**	T.C.
Doğum Yeri ve Tarihi	**:**	Gaziantep - 1977
Telefon	**:**	+90 (542) 473 6173
Faks	**:**	
E-Posta	**:**	skorkmaz@ktun.edu.tr sedat_korkmaz@hotmail.com

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	:	Gaziantep Anadolu Lisesi, Şehitkamil, Gaziantep	1995
Üniversite	:	Selçuk Üniversitesi, Müh.-Mim. Fakültesi, Bilgisayar Müh., Selçuklu, Konya	2000
Yüksek Lisans	:	Selçuk Üniversitesi, Fen Bilimleri Ens., Bilgisayar Müh. ABD, Selçuklu, Konya	2004
Doktora	:	Konya Teknik Üni., Lisansüstü Eğitim Ens., Bilgisayar Müh. ABD, Selçuklu, Konya	2019

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2018-	Konya Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü	Öğretim Görevlisi
2008-2018	Selçuk Üniversitesi Bilgisayar Mühendisliği Bölümü	Öğretim Görevlisi
2000-2008	Selçuk Üniversitesi Bilgisayar Mühendisliği Bölümü	Araştırma Görevlisi

UZMANLIK ALANI

Optimizasyon Yöntemleri, Veri Tabanı Yönetimi, Yazılım, Meta-sezgisel Algoritmalar

YABANCI DİLLER

İngilizce

YAYINLAR

Korkmaz, S., Babalik, A. ve Kiran, M. S., 2018, An artificial algae algorithm for solving binary optimization problems, International Journal of Machine Learning and Cybernetics, 9 (7), 1233-1247. (Doktora tezinden yapılmıştır)

Korkmaz, S. ve Kiran, M. S., 2018, An artificial algae algorithm with stigmergic behavior for binary optimization, Applied soft computing, 64, 627-640. (Doktora tezinden yapılmıştır)