KÜRE ÜZERİNDE 3 BOYUTLU GEZGİN SATICI PROBLEMİ ÇÖZÜMÜNDE YAPAY ATOM ALGORİTMASI OPTİMİZASYONUNUN PARALEL PROGRAMLAMA İLE UYGULAMASI

2018
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

Ayşe Nur ALTINTAŞ TANKÜL

KÜRE ÜZERİNDE 3 BOYUTLU GEZGİN SATICI PROBLEMİ ÇÖZÜMÜNDE YAPAY ATOM ALGORİTMASI OPTİMİZASYONUNUN PARALEL PROGRAMLAMA İLE UYGULAMASI

Ayşe Nur ALTINTAŞ TANKÜL

Karabük Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümünde
Yüksek Lisans Tezi
Olarak Hazırlanmıştır

KARABÜK
Temmuz 2018

Ayşe Nur ALTINTAŞ TANKÜL tarafından hazırlanan "KÜRE ÜZERİNDE 3 BOYUTLU GEZGİN SATICI PROBLEMİ ÇÖZÜMÜNDE YAPAY ATOM ALGORİTMASI OPTİMİZASYONUNUN PARALEL PROGRAMLAMA İLE UYGULAMASI" başlıklı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Dr. Öğr. Üyesi Burhan SELÇUK
Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. 13/07/2018

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Doç. Dr. Ergin YILMAZ (ZBEÜ)	
Dr. Öğr. Üyesi Ümit ATİLA (KBÜ)	
Dr. Öğr. Üyesi Burhan SELÇUK (KBÜ)	

..../..../2018

KBÜ Fen Bilimleri Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Prof. Dr. Filiz ERSÖZ
Fen Bilimleri Enstitüsü Müdür V.

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Ayşe Nur ALTINTAŞ TANKÜL

ÖZET

Yüksek Lisans Tezi

KÜRE ÜZERİNDE 3 BOYUTLU GEZGİN SATICI PROBLEMİ ÇÖZÜMÜNDE YAPAY ATOM ALGORİTMASI OPTİMİZASYONUNUN PARALEL PROGRAMLAMA İLE UYGULAMASI

Ayşe Nur ALTINTAŞ TANKÜL

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğr. Üyesi Burhan SELÇUK
Temmuz 2018, 98 Sayfa

Optimizasyon algoritmaları, verilen problem için her zaman en iyi sonucu verebilecek durumda olmasalar bile, en kısa sürede kabul edilebilir sonuçlar verir. Optimizasyon problemlerini çözmek için popüler hale gelen klasik matematiksel yöntemler ve meta-sezgisel yöntemler vardır. Meta-sezgisel yöntemler fizik tabanlı, sosyal tabanlı, biyolojik tabanlı, kimya tabanlı, spor tabanlı, sürü tabanlı, matematik tabanlı ve aynı zamanda hibrit tabanlı gibi birçok türde kategorize edilebilir. Bu çalışmada, optimizasyon teknikleri olan sürü tabanlı Karınca Kolonisi Algoritmasının ve Yapay Arı Kolonisi Algoritmasının, biyolojik temelli olan Genetik Algoritmanın ve kimya tabanlı yöntemlerden olan Yapay Atom Algoritmasının, performansları, sıralı ve paralel programlama yöntemiyle karşılaştırılarak küre üzerinde 3 boyutlu Gezgin Satıcı Problemi (GSP) çözülmüştür. Paralel hesaplama, sonuçların daha hızlı elde edilmesi için aynı işlemin (parçalanmış ve uyarlanmış) eş

zamanlı olarak çoklu işlemciler üzerinde yürütülmesidir. Bu fikir, problemlerin hesaplanmasının küçük parçalara bölünmesi ve bu parçaların aynı anda çalıştırılması üzerine kuruludur. GSP, verilen harita üzerinde tüm şehirleri ziyaret etmenin en düşük maliyetli yolunu bulan ve başlangıç noktasına dönen rota planlama problemlerinden biridir, en iyi rotayı planlamayı amaçlar. Algoritmaların rota mesafesi ve bu rotanın hesaplanma süresi açısından performansları incelenecektir. Uygulamayı gerçekleştirmek ve deney sonuçlarını gözlemlemek için bir ara yüz tasarlanacaktır.

Anahtar Kelimeler : Yapay atom algoritması, optimizasyon, 3 boyutlu gezgin satıcı problemi, paralel programlama.

Bilim Kodu **:** 924. 1.172

ABSTRACT

M. Sc. Thesis

APPLICATION OF ARTIFICIAL ATOM ALGORITHM OPTIMIZATION TO 3-DIMENSIONAL TRAVELING SALESMAN PROBLEM SOLUTION ON SPHERE BY PARALLEL PROGRAMMING

Ayşe Nur ALTINTAŞ TANKÜL

Karabuk University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Thesis Advisor:
Asst. Prof. Dr. Burhan SELÇUK
July 2018, 98 Page

Optimization algorithms yield acceptable results in the shortest time, even if they cannot always guarantee the best result in the given problem. There are classical mathematical methods and meta-heuristic methods that have become very popular for solving optimization problems. Meta-heuristic methods can be categorized in many types such as physics based, social based, biological based, chemistry based, sport based, swarm based, mathematics based and hybrid based. In this study, the performances of the Ant Colony System Algorithm and Artificial Bee Colony Algorithm that are swarm based, Genetic Algorithm that is biological based and Artificial Atom Algorithm that is chemistry-based methods, which are optimization techniques, are compared in a sequential and parallel programming way in order to solve the 3 dimensional Traveling Salesman Problem (TSP). Parallel computation is the simultaneous execution of the same task (fragmented and adapted) on

multiprocessors to get results faster. This idea is based on the division of the computation of problems into small pieces of work and running them simultaneously. TSP is one of the route planning problems that finds the lowest cost path of visiting all the cities on the giving map and returns to starting point, it was aimed to plan the best route. The performance of algorithms in terms of the route distance and the calculation time of this route will be examined. An interface will be designed to apply the implementation and observe the experimental results.

Key Words : Artificial atom algorithm, optimization, 3-dimensional travelling salesman problem, parallel programming.

Science Code : 924. 1.172

TEŞEKKÜR

Tez çalışmam sırasında bilgi, birikim ve tecrübeleri ile bana yol gösterici ve destek olan değerli danışman hocam sayın Dr. Öğr. Üyesi Burhan SELÇUK'a, sonsuz teşekkür ve saygılarımı sunarım.

Fırat Üniversitesi'nden Dr. Ayşe ERDOĞAN YILDIRIM'a yardımından ötürü teşekkür ederim.

Çalışmalarım boyunca maddi ve manevi destekleriyle beni hiçbir zaman yalnız bırakmayan eşim Mehmet TANKÜL, babam Adnan ALTINTAŞ, annem Elif ALTINTAŞ ve kardeşlerime de sonsuz teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

Sayfa

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR

SİMGELER

u : Küre yüzeyi üzerinde sabit boylam çizgileri

v : Küre yüzeyi üzerinde sabit boylam çizgileri

x : x Kordinatı

y : y Kordinatı

z : z Kordinatı

d : Mesafe

$\vec{V}$: Vektör

q_0 : Seçilme olasılığı

r : Seçilme olasılığı

G : Graf

V : Graf Düğümü

E : Graf Kenarı

φ : Azimutal açı

θ : Zenit açı

r : Kürenin yarıçapı (Bölüm 2)

α : Birim kürenin merkezinden iki noktaya olan vektörler arasındaki küçük açı (Bölüm 2)

α : İyonik Oran (Bölüm 6)

β : Kovalent Oran

n : Atom Sayısı

r : Elektron sayısı (Bölüm 6)

A : Atom

E : Elektron

f : Amaç fonksiyonu

P : Seçilme olasılığı

τ : Feromon miktarı

k : Karınca sayısı

θ : Ters döndürme işlemi

$\Diamond$: Ters döndürme işlem dizilerinin yapımı

$\oplus$: İki ters çevirme işleminin arka arkaya yapımı

$\odot$: Verilen olasılığa göre işlemin gerçekleştirilmesi

KISALTMALAR

2D : 2 Boyutlu

3D : 3 Boyutlu

A^3 : Yapay Atom Algoritması (Artificial Atom Algorithm)

ABC : Yapay Arı Kolonisi (Artificial Bee Colony)

ACO : Karınca Kolonisi Optimizasyonu (Ant Colony Optimization)

AIS : Yapay Bağışıklık Sistemi (Artificial Immune System)

ASO : Anarşik Toplum Optimizasyon (Anarchic Society Optimization)

BA : Yarasa Algoritması (Bat Algorithm)

BB-BC : Büyük-Patlama Büyük Çöküş (Big Bang–Big Crunch)

BFO : Bakteriyel Besin Arama Optimizasyonu Algoritması (Bacterial Foraging OptimizationAlgorithm)

CEA : Kültürel Evrim Algoritması (Cultural Evolution algorithm)

CPU : İşlemci (Central Processing Unit)

CS : Guguk Kuşu Arama (Cuckoo Search)

CSO : Kedi Sürüsü Optimizasyonu (Cat Swarm Optimization)

CX : Çevrim Çaprazlama (Cycle crossover (CX))

D : Boyut

DE : Diferansiyel Evrim (Differential Evolution)

EA : Seçim Algoritması (Election Algorithm)

ECO : Seçim Kampanyası Optimizasyon (Election Campaign Optimization)

EP : Evrimsel Programlama (Evolutionary Programming)

FPA : Çiçek Tozlaşma Algoritması (Flower Pollination Algorithm)

GA : Genetik Algoritma (Genetic Algorithm)

GPU : Grafik işlem birimi (Graphics Processing Unit)

GSP : Gezgin Satıcı Problemi

GWO : Gri Kurt Optimizasyonu (Grey Wolf Optimization)

HS : Armoni Arama (Harmony Search)

IA : İdeoloji Algoritması (Ideology Algorithm)

ICA : Emperyalist Rekabet Algoritması (Imperialist Competitive Algorithm)

İA : İyonik Alan

KA : Kovalent Alan

LCA : Lig Şampiyonası Algoritması (League Championship Algorithm)

MPI : Message Passing Interface

MPMD : Çoklu Program Çoklu Veri (Multiple Program Multiple Data)

OBX : Sıra Tabanlı Çaprazlama (Order-based crossover)

OpenMP : Open Multi-Processing

OX : Sıra Çaprazlama (Order crossover)

PA^3 : Paralel Yapay Atom Algoritması (Artificial Atom Algorithm)

PABC : Paralel Yapay Arı Kolonisi (Artificial Bee Colony)

PACO : Paralel Karınca Kolonisi Optimizasyonu (Ant Colony Optimization)

PBX : Konum Tabanlı Çaprazlama (Position-based crossover)

PGA : Paralel Genetik Algoritma (Genetic Algorithm)

PMX : Kısmen Eşlemeli Çaprazlama (Partially-mapped crossover)

PSO : Parçacık Sürü Optimizasyonu (Particle Swarm Optimization)

SA : Benzetimli Tavlama (Simulated Annealing)

SCO : Toplum ve Medeniyet Optimizasyon (Society and Civilization Optimization)

SEOA : Sosyal Duygusal Optimizasyon Algoritması (Social Emotional Optimization Algorithm)

SLC : Futbol Ligi Rekabet Algoritması (Soccer League Competition Algorithm)

SPMD : Tek Program Çoklu Veri (Single Program Multiple Data)

TLBO : Öğretme Öğrenme Tabanlı Optimizasyon (Teaching–Learning-Based Optimization)

BÖLÜM 1

GİRİŞ

Dünyamızdaki en temel ilkelerden biri, en uygun yani optimal durumun araştırılmasıdır. Fizikteki atomların, elektronlarının enerjisini en aza indirgemek için bağlar oluşturmaya çalıştığı mikrokozmandan başlar, donma işlemi sırasında moleküller katı cisimler oluşturduğunda enerji için en uygun kristal yapılar bulmaya çalışırlar [1]. Elbette ki bu süreçler sadece fizik kanunlarından kaynaklanır. Aynı şey, türlerin çevrelerine daha iyi adaptasyonuna neden olan, en uygun hayatta kalma ilkesi için de geçerlidir [1]. Burada, yerel optimumlar, çevredeki diğer tüm hayvanlara hakim olan, iyi adapte olmuş bir türlerdir.

İnsanlık var olduğu sürece, birçok alanda mükemmellik için gayret gösterir. En az gayretle maksimum mutluluk derecesine ulaşmak ister [1]. Ekonomide kar ve satışlar en üst düzeye çıkarılmalı ve maliyetler mümkün olduğunca düşük olmalıdır. Bu nedenle, optimizasyon, günlük yaşamı da içine alan bilimlerin en eskilerinden biridir.

Mühendislikte optimizasyon, verilen koşullar altında mümkün olan en iyi sonuca ulaşmaktır. Tasarımda, inşaatta, bakımda ve buna benzer alanlarda mühendislerin karar almaları gerekir. Bu tür tüm kararların amacı, çabayı en aza indirmek ve kazancı maksimize etmektir. Çaba veya kazanç genellikle bazı tasarım değişkenlerinin bir fonksiyonu olarak ifade edilebilir. Dolayısıyla, optimizasyon bir fonksiyonun maksimum veya minimum değerini veren koşulları bulma işlemidir [2]. Örnek olarak minimizasyon; maliyet, mesafe, çapraz uzunluğu, ağırlık, işleme süresi, malzeme, enerji tüketim, nesne sayısı, maksimizasyon; kâr, değer, çıktı, dönüş, verim, faydalılık, verimlilik, kapasite, nesne sayısı gösterilebilir.

Genel ve soyut bir şey varsa, her zaman onunla ilgilenen bir matematik disiplini vardır. Optimizasyon, uygulanan matematiği ve analitik çözümü, en iyi duruma getirmeye odaklanan bir dalıdır. Optimizasyonun amacı, $F = \{f1, f2, .., fn\}$ kriterler kümesine göre olası en iyi öğeleri bulmaktır. Bu kriterler, objektif fonksiyonlar olarak adlandırılan matematiksel fonksiyonlar olarak ifade edilir [1].

Genel bir optimizasyon probleminin matematiksel tanımı [3]:

- Optimize edilecek fonksiyon $f(x)$ olarak kabul edilirse,
- Kısıtlayıcı fonksiyonlar $f(x) = 0$ ve $g(x) \leq 0$
- Tasarım değişkenleri $x_l \leq x \leq x_u$

Optimize edilecek fonksiyon uygunluk fonksiyonu veya hedef fonksiyon olarak tanımlanmaktadır. Hedef fonksiyondaki değişkenler tasarım değişkenleri, fonksiyonun sağlanmasında uyulması gereken kısıtlayıcıları sağlayan fonksiyonlar ise kısıtlayıcı fonksiyon olarak adlandırılmaktadır. Burada amaç, verilen problemde istenilen sonuca göre hedef fonksiyonun minimum veya maksimum yapılmasıdır [3].

Genellikle, optimizasyon algoritmaları iki temel sınıfa ayrılabilir: deterministik ve olasılıksal algoritmalar. Belirli bir problemin olası çözümleri ve yararları arasında net bir ilişki varsa, çoğunlukla deterministik algoritmalar kullanılır [2]. Deterministik algoritmalarda matematiksel fonksiyonların çözümünde sabit parametreler kullanılır, veriler önceden bilinmektedir ve sistemi başlattığınızda tam olarak ne olacağını bilirsiniz. Bir çözüm adayı ile onun "uygunluğu" arasındaki ilişki o kadar belirgin ya da çok karmaşık değilse ya da arama alanının boyutları çok yüksek olursa, bir problemi deterministik olarak çözmek zorlaşır. Bu tarz bir problemi deterministik olarak çözmeyi denemek, nispeten küçük problemler için bile, arama alanının kapsamlı bir şekilde numaralandırılmasına neden olabilir.

Deterministik algoritmaların zorlandığı bu noktada probabilistik algoritmalar devreye girer. Optimizasyonda en önemli araştırma alanlarından biri haline gelen probabilistik algoritmalarda ise şans unsuru söz konusudur [1]. Probabilistik algoritmalarda tanımlanan parametreler rastgele değerler alır ve doğrusal olmayan

problemlerin çözümünde kullanılmaktadır [3]. Bu algoritöalar daha kısa bir çalışma zamanı içinde, mümkün olan en iyi seviye olmasa bile kabul edilebilir bir sonuç elde eder ve bazen sezgisel bazen de meta-sezgisel olarak adlandırılır. Deterministik algoritmalar genellikle çözüm adaylarının işlem sırasını tanımlamak için sezgisel yöntemler kullanır. Diğer taraftan, olasılıkçı yöntemler yalnızca sezgisel tarafından seçilen daha sonraki hesaplamalarda arama alanının unsurlarını göz önüne alabilir [1]. Optimizasyonda kullanılan sezgisellikler, olası çözümlerden hangisinin daha sonra inceleneceğine karar vermeye yardımcı olan işlevlerdir, bazen sezgisel bazen de meta-sezgisel olarak adlandırılır [1].

Sezgisellik, algoritmanın hangi çözüm adayının bir sonraki adımda test edileceğini veya bir sonraki adayın nasıl üretilebileceğini belirlemek için algoritma tarafından toplanan bilgileri kullanan optimizasyon algoritmasının bir parçasıdır [4].

Meta-sezgisel yöntem, optimizasyon problemine, özellikle yetersiz veya eksik bilgiyle veya sınırlı hesaplama kapasitesinde, yeterince iyi bir çözüm sağlayabilen sezgisel bir yöntem bulmak, üretmek veya seçmek için tasarlanmış daha üst düzey bir prosedür veya sezgisel yöntemdir [4]. Bu algoritmaların en önemli özelliklerinden biri yerel aramalar sırasında bellekte önceki bulduğu değerleri saklar. Bu bellekte tutma işlemini gerçekleştirirken, aynı zamanda komşuları tarafından alınan ölçümleri de dikkate alır [3]. Arama uzayında daha verimli çözümler sunabileceği için meta-sezgisel yöntemler son yıllarda yaygın bir şekilde kullanılmıştır.

Gezgin Satıcı Problemi (GSP) bir satış elemanı ve şehirlerden oluşmaktadır. Satış elemanı şehirlerin her birini ziyaret etmek ve başladığı şehre dönmek zorundadır. Problemindeki amaç, seyahat eden satış elemanının seyahatinin toplam uzunluğunu en aza indirmektir. GSP NP-Zor bir problemdir, bu da demektir ki tam çözümü hesaplayacak olan algoritma N'de üstel olan bir süre gerektirir [5]. Bu yüzden GSP optimizasyon algoritmalarının kullanıldığı problemlerden birisidir. GSP 2D ve 3D olarak tasarlanabilmektedir.

Optimizasyon algoritmaları her ne kadar deterministik algoritmalara göre daha hızlı olsa da problemlerin boyutu arttıkça çalışma zamanı da artmaktadır. Bu noktada tek

bir işlemci tarafından yürütülen seri hesaplamalar yerine paralel programlama tercih edilebilir. Paralel programlamada, birden fazla işlemci yardımı ile algoritmayı daha hızlı çalıştırabilmek amaçlanmaktadır [6]. Paralel programlamanın çeşitli uygulama şekilleri vardır. Bunlar çalışmanın diğer bölümlerinde detaylı bir şekilde ele alınacaktır.

1.1. TEZ ÇALIŞMASININ AMACI VE KAPSAMI

Bu tez çalışmasının amacı, günümüzde, ulaşım ve lojistik uygulamaları, malzeme akış sistem tasarımı, araç rotalama problemleri, uçaklar için havaalanı rotalaması, elektronik devre tasarımı, matbaa zamanlama, bilgisayar kablolama gibi daha birçok kullanım alanı olan GSP'nin küre üzerinde tasarlanmış modelinde optimizasyon algoritması olan Yapay Atom Algortması(A^3)'nın uygulamasını gerçekleştirmek, bunu Karınca Kolonisi Algoritması(ACO), Yapay Arı Kolonisi Algoritması (ABC) ve Genetik Algoritma (GA) ile karşılaştırmaktır.

Optimizasyon algoritmaları deterministik algoritmalardan daha hızlı olsa da, problemlerin boyutu arttıkça çalışma süresi de artar. Tek bir işlemci tarafından gerçekleştirilen seri hesaplamalar yerine bu noktada paralel programlama tercih edilebilir. Paralel programlamada, birden fazla işlemcinin yardımıyla algoritmanın daha hızlı çalıştırılması amaçlanmaktadır. Paralel programlama çeşitli uygulama biçimlerine sahiptir.

Bu çalışmada, A^3, GSP'ye paralel olarak uygulanmış ve literatürde bulunan diğer algoritmalar ile karşılaştırılmıştır. Öncelikle GSP, A^3, GA, ACO ve ABC 'nin tanımı verilmiş ve daha sonra bu algoritmaların GSP'de uygulanması açıklanmıştır ve birbiriyle karşılaştırmaları yapılmıştır.

1.2. TEZ ÇALIŞMASININ ORGANİZASYONU

İkinci bölümde GSP tanımı yapılacak, karmaşıklığı, sınıflandırması, uygulama alanları ve matematiksel modeli açıklanacaktır.

Üçüncü bölümde Paralel Programlama terminolojisi, paralellik türleri, paralel bilgisayar bellek mimarileri, programlama modelleri ve paralel uygulamalar açıklanacaktır.

Dördüncü bölümde meta-sezgisel algoritmalar üzerinde durulacak ve özelliklerine göre sınıflandırılan sezgisel yöntemler açıklanacaktır.

Beşinci bölümde Yapay Atom Algoritması, temel kavramlar ve tanımlar, kovalent bağ, iyonik bağ, elektron etkileri, A^3'ün GSP'ye uygulanması ve A^3'ün GSP'ye paralel olarak uygulanmasının adımları açıklanacaktır.

Altıncı bölümde Genetik Algoritma, Temel kavramlar ve genel algoritma, başlangıç popülasyonu, doğal seçilim, çaprazlama, mutasyon, GA'nın GSP'ye uygulanması ve GA'nın GSP'ye parallel uygulanması açıklanacaktır.

Yedinci bölümde Karınca Kolonisi Algoritması, parametreleri, geçiş kuralı, feromen güncellemesi, lokal feromen güncellemesi, global feromen güncellemesi ve ACO'nun GSP'ye paralel olarak uygulanması açıklanacaktır.

Sekizinci bölümde Yapay Arı Kolonisi Algoritması, parametreler ve algoritma tanımı, besin kaynaklarının oluşturulması, işçi arı aşaması, gözcü arı aşaması, kâşif arı aşaması, ABC'nin GSP'ye uygulanması ve ABC'nin GSP'ye parallel uygulanması açıklanacaktır.

Dokuzuncu bölümde yapılan 3D GSP, A^3, ACO, GA ve ABC ile uygulanacak ve performansları değerlendirilecektir.

Son bölümde ise yapılan uygulama hakkında genel bir değerlendirmenin yapılacağı sonuç kısmı yer alacaktır.

BÖLÜM 2

GEZGİN SATICI PROBLEMİ

2.1. TANIMI

Gezgin Satıcı Problemi, basit ama çok zor çözülebilen klasik bir kombinasyonel optimizasyon problemidir. Verilen şehirler ve her şehir çifti arasındaki mesafe göz önüne alındığında, problem, her şehri bir kere ziyaret eden ve başlangıç noktasına dönen mümkün olan en kısa rotayı bulmaktır [6].

Şehirlerin köşelerle temsil edildiği ve mesafelerin ağırlıklı kenarlarla gösterildiği tam ağırlıklı yönlendirilmemiş grafikte G (V, E), GSP belirli bir köşeden başlayıp, diğer tüm köşeleri bir kez ziyaret eden ve başlangıç tepe noktasında biten en aza indirgenmiş Hamilton döngüsünü bulmaktır [7].

2.2. KARMAŞIKLIĞI

Gezgin satıcı problemi, NP sınıfından karmaşıklığı olan bir problemdir ve tam olarak polinom zamanında çözüm elde edilemez.

Düğümlerin sayısı arttıkça, sorunu çözmek için harcanan zaman katlanarak artar. GSP'deki düğüm sayısının artmasıyla paralel olarak, çözüm zamanı katlanarak Çizelge 2.1'de gösterilmektedir.

Çizelge 2.1. Hamilton Döngülerinin değerlendirilmesi [8].

Düğüm Sayısı	Döngü Sayısı (n-1)!	Gerekli Zaman
12	39.916.800	0.004 saniye
13	479.001.600	0.05 saniye
14	6.227.020.800	1 saniye
15	87.178.291.200	9 saniye
16	1.307.647.368.000	2 dakika
17	2.1 * 1013	35 dakika
18	3.6 * 1014	10 saat
19	6.4 * 1015	7.5 gün
20	1.2 * 1017	140 gün
21	2.4 * 1018	7.5 yıl
22	5.1 * 1019	160 yıl
23	1.1 * 1021	3.500 yıl
24	2.6 * 1022	82.000 yıl
25	6.2 * 1023	2 milyon yıl

Eğer başlangıç şehri problemde veriliyorsa, muhtemel Hamilton yollarının sayısı kalan $(n-1)$ adet şehrin yer değişmesine yani $(n-1)!$'e eşit olmaktadır. Yani kesin çözüm, tüm permütasyonları denemek ve kaba kuvvet aramayı kullanarak en düşük maliyetli yolu seçmek olacaktır. Bu yöntem, az sayıdaki şehir için en iyi çözümü garanti eder, ancak 20 şehir için bile pratik olmaz. Problemin çözümü basit olmasına rağmen sonuca ulaşmak için tüm çözüm uzayının taranması uygulanabilir bir yaklaşım değildir [7]. Bu yüzden GSP problemlerini çözmek için sezgisel ve meta-sezgisel tekniklerin kullanılması daha verimli bir yoldur.

2.3. MATEMATİKSEL MODELİ

Gezgin satıcı problemine ait doğrusal programlama modeli Eşitlik 2.1'de gösterilmiştir [9]:

$$
\begin{aligned}
\text{Min} \quad & \sum_{i=0}^{n} \sum_{i=0}^{n} c_{ij} x_{ij} & & \\
& 0 \leq x_{ij} \leq 1 & & i, j = 0, 1, \dots, n \\
\text{s. t.} \quad & x_{ij} \text{ tam sayı} & & i, j = 0, 1, \dots, n \\
& \sum_{i=0, i \neq j}^{n} x_{ij} = 1 & & j = 0, 1, \dots, n \\
& \sum_{j=0, i \neq j}^{n} x_{ij} = 1 & & i = 0, 1, \dots, n \\
& u_i - u_j + n x_{ij} \leq n - 1 & & 1 \leq i \neq j \leq n
\end{aligned}
\tag{2.1}
$$

GSP'nin matematiksel modelinde gösterilen i şehrinden j şehrine olan uzaklık c_{ij} olarak gösterilmiştir. x_{ij} değeri ise i şehrinden j şehrine gidilip gidilmediğini tutan değişkendir, eğer gidilmişse 1, gidilmemişse 0 değerini almaktadır. GSP modelinde istenen c_{ij}, x_{ij} çarpımlarının toplamını minimize etmektir. Modelde n, şehir sayısını ifade ederken, u parametresi ise eksiksiz tam bir tur yapılması için koşulları sağlamak üzere modele eklenen yapsay bir değişkendir.

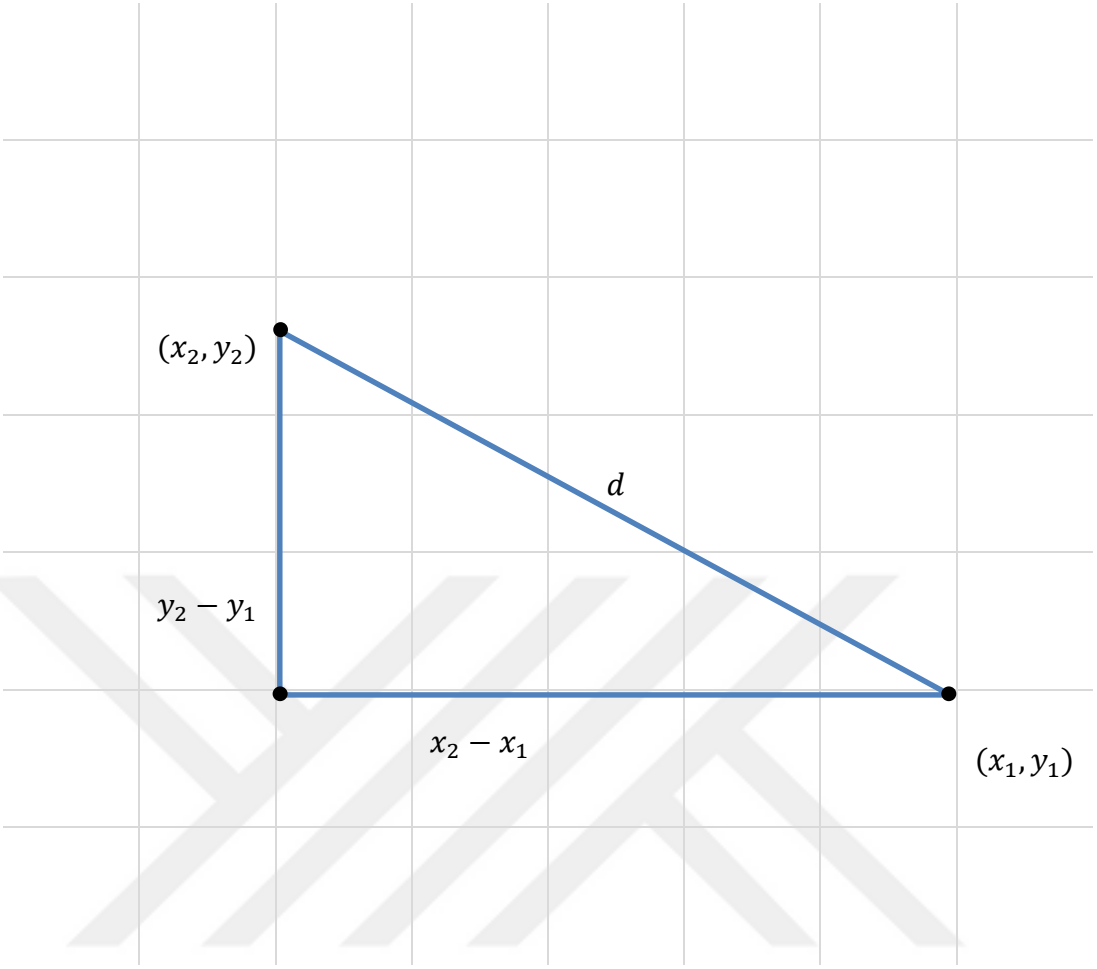
2.4. 2D GSP

GSP genellikle 2D Öklid düzlemi üzerinde bir problem olarak tanımlanır.

Şehirlerin pozisyonları tam olarak bilinir ve şehirler arasındaki mesafeler Öklid mesafeleridir [10]. Problemin çözümü ise optimal turu belirlemekten geçer.

Verilen iki nokta arasındaki Öklit mesafesi, onları birbirine bağlayan çizgi parçasının uzunluğudur [11]. Noktalar (x_1, y_1) ve (x_2, y_2) 2 boyutlu uzaydaysa aralarındaki Öklid uzaklığı Eşitlik 2.2'deki gibi hesaplanır.

$$
d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2.2}
$$



Şekil 2.1. 2D düzlemde iki nokta arasındaki Öklid mesafesi.

2.5. KÜRE ÜZERİNDE 3D GSP

3D GSP problemi denildiğinde akla ilk gelen 3D Öklit düzlemi üzerinde olan GSP problemidir. Bu şekilde düşündüğümüz zaman 2 şehir arasındaki mesafeyi hesaplamak için Eşitlik 2.3 kullanılmaktadır [12].

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2} \tag{2.3}$$

Küre üzerinde 3D GSP, Öklid düzlemi üzerinde çalışılan GSP probleminden biraz daha farklıdır. Küre üzerinde GSP'de, iki şehir arasındaki uzaklık, küre yüzeyinde bulunan iki nokta arasındaki yayın uzunluğu hesaplanarak bulunur. Küre üzerinde

GSP, Genetik Algoritma [13,14], Karınca Kolonisini Algoritması [15], Parçacık Sürü Optimizasyon Algoritması [16], hibrid Ateşböceği Sürü Optimizasyonu [17], Çiçek Tozlaşma Algoritması [18] ve Gugukkuşu Arama Algoritması [19] kullanılarak daha önce uygulanmıştır.

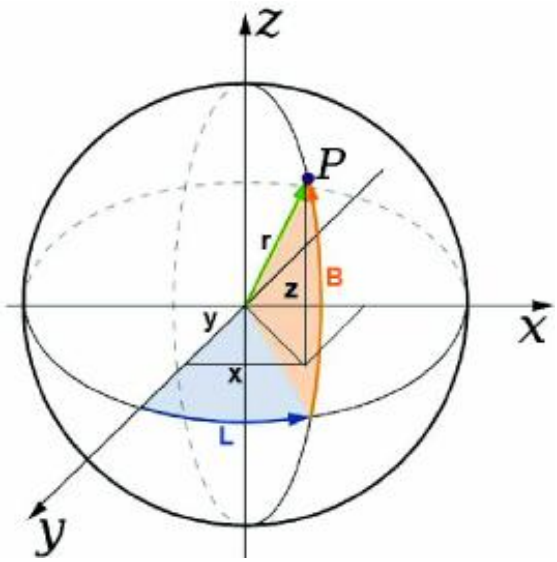
2.5.1. Küre

Daire sabit bir noktaya (merkez) sabit bir mesafede hareket eden bir nokta tarafından oluşturulur [20].

Bir küre, kürenin yüzeyindeki tüm noktaların kürenin merkezinden sabit bir uzaklığı olan çemberin üç boyutlu bir formudur [13].

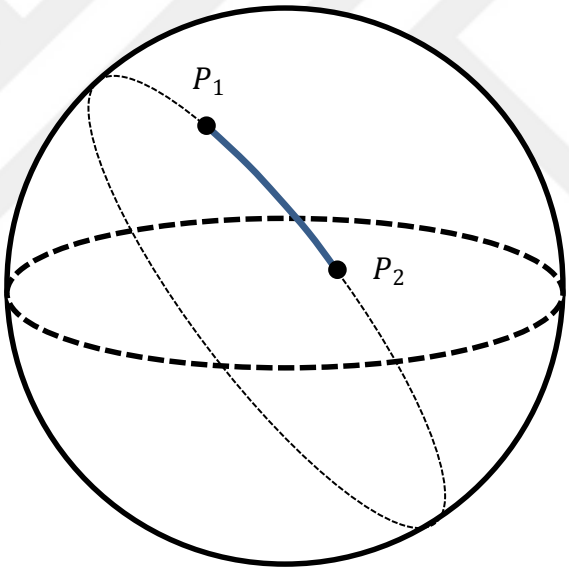
Kartezyen koordinatlarda, koordinat üzerinde ortalanmış olan yarıçap r'ye sahip küresel bir yüzey, Eşitlik 2.4 denklemi sağlayan noktaların (x, y, z) tanımlanmasını sağlar [21].

$$x^2 + y^2 + z^2 = r^2 \tag{2.4}$$



Şekil 2.2. Bir noktanın üç boyutlu kartezyen (x, y, z) ve küresel (r yarıçapı, B enlem ve L boylam açısı) koordinatları [22].

Küre dediğimiz zaman akla ilk olarak gelenlerden birisi de dünyadır. Fakat dünya tam olarak bir küre değildir. Bir greyfurtun yapısı, dünyanın şeklini tanımlamak için küreden daha iyi bir analojidir, çünkü üstten ve alttan daha yassıdır. Dünyanın Kuzey ve Güney Kutbu boyunca ölçülen çapı, Ekvatordaki iki noktadan ölçülen çapından yaklaşık 40 km daha azdır. Dünyanın yarıçapı genellikle 6370 kilometre olarak verilir [20]. Dünya üzerindeki iki nokta arasındaki çizginin temel yönlerini modelleme amacıyla, Dünya'yı 6400 kilometre yarıçapında gerçek bir küre olarak ele alırsak, kavramsal olarak basit bir çerçeveden makul sonuçlar elde edebiliriz. Bir küre ya da diğer eğimli yüzey üzerindeki iki nokta arasındaki en kısa çizgi orthodrome olarak adlandırılır [23] ve bu çizgi, yerel olarak boyu en aza indirgeyen bir eğrinin, yani büyük çemberin (jeodezik) bir parçasıdır.

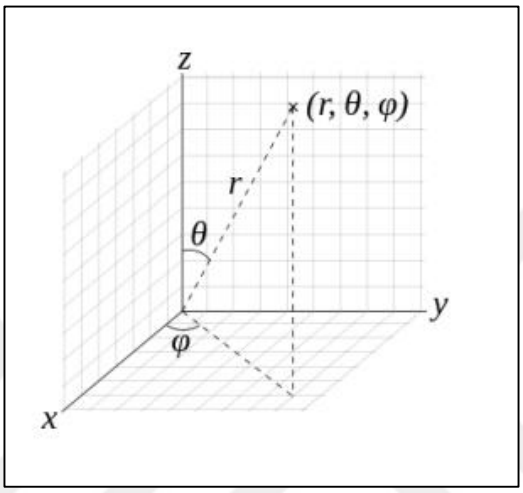


Şekil 2.3. Küre üzerinde P_1 ve P_2 noktasından geçen büyük çemberin gösterimi.

2.5.2. Bir Noktanın Küresel Koordinatları

Küresel koordinatlar, aynı zamanda küresel kutupsal koordinatlar olarak adlandırılır, bir küre veya küremsi cisim üzerindeki konumları tanımlamak için doğal olan eğrisel koordinatlar sistemidir [24].

φ, verilen noktanın *x-y* düzleminde, $0 \leq \varphi < 2\pi$ aralığında, *x* eksenine olan açısıdır ve azimutal açı olarak adlandırılır (boylam olarak anıldığında λ olarak gösterilir). θ, $0 \leq \theta < \pi$ aralığında, verilen noktanın pozitif *z* eksenine olan kutup açısıdır (zenit olarak da bilinen açı, enlem δ ile 90° arasındaki farka eşittir $\theta = 90° - \delta$). ρ ise kürenin merkezinden noktaya olan vektörün uzunluğudur ve yarıçap *r*'ye eşittir [24]. Bu değerler, kürenin üzerindeki nokta için matematikte yaygın olarak kullanılmaktadır.



Şekil 2.4. Yaygın olarak kullanılan küresel koordinatlar *(r, θ, φ)* [21].

(ρ, θ, φ) aşağıdaki şekilde Kartezyen koordinatlarda verilir [24]:

$$\begin{bmatrix} r \\ \theta \\ \varphi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \arccos(z/r) \\ \arctan(y/x) \end{bmatrix}, 0 \leq \theta \leq \pi, 0 \leq \varphi < 2\pi \tag{2.5}$$

veya tersine:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\,sin\theta\,cos\varphi \\ r\,sin\theta\,sin\varphi \\ r\,cos\theta \end{bmatrix} \tag{2.6}$$

Eğri (veya düzlem) Öklid yüzeyleri iki boyutlu nesnelerdir ve bir yüzeydeki konumlar *u* ve *v* parametreleri ile tanımlanabilir. Yüzey üzerindeki bir koordinat konumu, parametrik vektör fonksiyonuyla temsil edilebilir [21]. Burada *x, y ve z* için

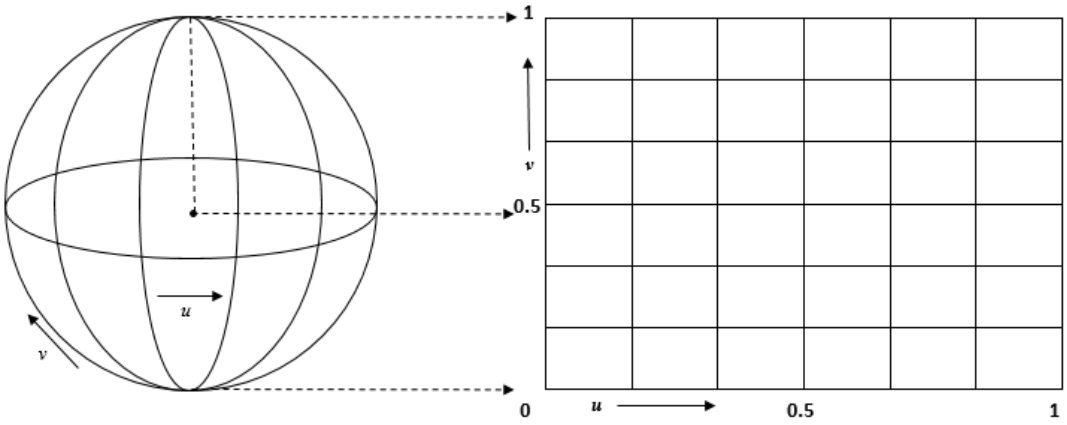
Kartezyen koordinat değerleri, *u* ve *v* parametrelerinin fonksiyonları olarak Eşitlik 2.7'teki gibi ifade edilir.

$$P(u) = (x(u,v), y(u,v), z(u,v)) \tag{2.7}$$

Kartezyen koordinatların her biri artık iki yüzey parametresinin *u* ve *v* fonksiyonudur. Merkezi orijininde ve yarıçapı *r* olan küresel bir yüzey, Eşitlik 2.6'teki vektör denklemleri kullanılarak Eşitlik 2.8'daki gibi tanımlanabilir [21]:

$$\begin{aligned} x(u,v) &= r\ sin(\pi v)cos(2\pi u) \\ y(u,v) &= r\ sin(\pi u)\ sin(2\pi v) \\ z(u,v) &= r\ cos(\pi v) \end{aligned} \tag{2.8}$$

u parametresi, yüzey üzerinde sabit boylam çizgileri tanımlarken *v* parametresi, sabit enlem çizgilerini tanımlar. Küresel yüzey üzerindeki *u, v* parametrelerinin koordinat pozisyonları Şekil 2.5'te gösterilmiştir.



Şekil 2.5. Küresel yüzey üzerindeki *u, v* parametrelerinin gösterimi.

Örnek olarak, *u* ve *v* parametrelerinin farklı değerleri için *x, y, z* koordinatları Eşitlik 2.8'ya göre Çizelge 1'de verilmiştir [15]. Hesaplamaların basitleştirilmesi için yarıçapı 1 olan birim küre olan kullanılmıştır.

Çizelge 2.2. *u* ve *v* parametrelerinin farklı değerleri için küresel yüzey koordinatları [15].

u	*v*	*x*	*y*	*z*
0	**0**	0	0	1
0	**0.5**	1	0	6.123233e−17
0	**1**	1.224646e−16	0	−1
0.5	**0**	0	0	1
0.5	**0.5**	−1	1.224646e−16	6.123233e−17
0.5	**1**	−1.224646e−16	1.499759e−32	−1
1	**0**	0	0	1
1	**0.5**	1	−2.449293e−16	6.123233e−17
1	**1**	1.224646e−16	−2.999519e−32	−1

2.5.3. Birim Küre Üzerinde Bulunan İki Nokta Arasındaki En Kısa Mesafenin Bulunması

Küre üzerinde bulunan iki nokta *(P_1, P_2)* arasındaki en kısa mesafe, bu iki noktadan geçen büyük çember üzerinde bulunan kısa yayın (Şekil 2.3) uzunluğu kadardır. Birim kürenin merkezinden iki noktaya olan vektörler arasındaki küçük açı alfanın (α) radyan cinsinden değeri, bu yayın uzunluğu hesaplanırken kullanılabilir. İki vektörün skaler çarpımı Eşitlik 2.9'de verilmiştir.

$$\overrightarrow{V1} \,.\, \overrightarrow{V2} = \left|\overrightarrow{V1}\right| . \left|\overrightarrow{V2}\right| . \cos\alpha \tag{2.9}$$

Eşitlik 2.9'de verilen $\overrightarrow{V1} \,.\, \overrightarrow{V2}$ değeri Eşitlik 2.10'deki gibi hesaplanır. Birim kürede vektörlerin uzunlukları 1 birim olduğundan Eşitlik 2.9'de α eşitliğin sol tarafında yalnız bırakıldığında Eşitlik 2.11 elde edilir.

$$\overrightarrow{V1} \,.\, \overrightarrow{V2} = P_{1x}P_{2x}P_{1y}P_{2y}P_{1z}P_{2z} \tag{2.10}$$

Eşitlik 2.11'da elde edilen α değeri iki nokta arasındaki en kısa mesafeyi ifade eder.

$$\alpha = \arccos(\overrightarrow{V1}.\overrightarrow{V2}) \tag{2.11}$$

2.4. UYGULAMALARI

GSP'nin kolay anlaşılır olması ve gerçek dünyadaki birçok pratik uygulamaya uyarlanabilir olması, üzerinde yoğun olarak çalışılmasının nedenlerindendir. GSP birçok gerçek hayat problemine uygulanmakla birlikte en çok yol ve rota planlama gibi konularda kullanılmaktadır [8,25].

GSP'nin kullanım alanları şu şekilde sıralanabilir:

1. Gaz türbini motorlarının yenilenmesi
2. X-ışını kristalografisi
3. Bilgisayar kablolaması
4. Depodaki sipariş toplama problemi
5. Araç yönlendirmesi
6. PCB üretiminde maskeleme
7. Baskı planlaması problemi
8. Okul otobüsü yönlendirme problemi
9. Mürettebat çizelgeleme problemi
10. Görüşme planlaması sorunu
11. Sıcak haddeleme planlama problemi
12. Görev planlama problemi
13. Küresel navigasyon uydu sistemi ölçüm ağları tasarımı
14. GSM operatörlerinin baz istasyonlarının yerleşim yerlerinin belirlenmesi
15. Birçok ulaşım ve lojistik uygulamaları
16. Malzeme akış sistem tasarımı
17. Araç rotalama problemleri
18. Uçaklar için havaalanı rotalaması,
19. Elektronik devre tasarımı
20. Sipariş toplama
21. En ucuz ya da hızlı hat ayarlama
22. Limancılık, havacılık, postacılık güzergâh ayarlama
23. Robot hareket planlama

3D GSP ise insansız hava araçlarının rota planlamasında [26] , dünya üzerinde jet ve uçak gibi her türlü hava aracının yaptığı uçuşların rotalamasında kullanılmaktadır [16]. Bunun yanısıra atom, molekül, protein gibi kimya, fizik ve biyoloji gibi alanlarda kullanılan bazı yapılar da küre ile ifade edilmektedir, dolayısıyla, küresel GSP çözümü, parça toplama, parça yerleştirme üzerine kurulu robot yolu planlama çalışmaları, rota planlama gibi mikro ve makro sistemler için doğrudan uygulamaları bulabilir [13].

BÖLÜM 3

PARALEL HESAPLAMA

Çoğu insan, ne anlama geldiğini anlayamasa bile, seri hesaplamaya aşinadır. İnsanların günlük yazıp çalıştırdıkları çoğu program seri programlardır. Bir seri program, tek bir bilgisayarda, tipik olarak tek bir işlemci üzerinde çalışır. Programdaki talimatlar birbiri ardına seri olarak yürütülür ve bir seferde sadece bir komut çalıştırılır [27]. Paralel hesaplama, bir programdaki birçok komutun aynı anda paralel olarak çalışmasına izin veren bir hesaplama şeklidir. Bunu başarmak için, bir programın bağımsız bölümlere ayrılması gerekir, böylece her işlemci diğer programlayıcılarla eş zamanlı olarak programın bir kısmını çalıştırabilir [27]. Yani paralel hesaplama, birçok problemin eş zamanlı olarak gerçekleştirildiği, büyük problemlerin genellikle daha küçük olanlara bölünebileceği prensibiyle çalışan bir hesaplama şeklidir ve bu küçük olan problemler daha sonra eş zamanlı (paralel olarak) olarak çözülür [28].

Paralel hesaplamanın en büyük avantajı makul bir zamanda çözülemeyen problemleri çözebilir olmasıdır. Bunun yanında tek CPU (Central Processing Unit) yani tek işlemci ile çözülemeyen büyüklükteki problemlerin çözülebilmesi paralel hesaplamanın tercih edilmesini sağlar.

3.1. TERMİNOLOJİ

Düğüm: Genellikle kendi işletim sistemini kendi örneğini çalıştıran bir bilgisayar sisteminin ayrı bir birimi [29].

- Stampede 6400 düğümleri vardır

İşlemci (CPU): Ortak bir belleği ve yerel diski paylaşan yonga [29].

-Stampede, düğüm başına iki Sandy Bridge işlemcisine sahiptir

Çekirdek: Bir işlem yongasını destekleyebilen bir bilgisayar yongasındaki bir işlem birimi [29].

- Stampede, işlemci başına 8 çekirdeğe veya düğüm başına 16 çekirdeğe sahiptir.

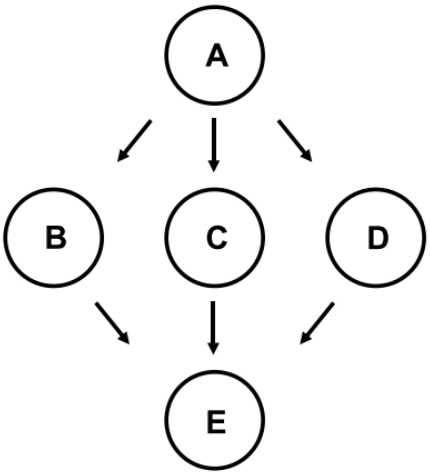
Yardımcı işlemci: Birincil işlemcinin özelliklerini desteklemek için tasarlanmış bir mikroişlemci [29].

- Stampede, her işlemcideki her işlemciye 61 çekirdekle tek bir Phi işlemci içerir.

Küme: Tek bir kaynak olarak işlev gören düğümler topluluğu [29].

3.2. PARALELLİK TÜRLERİ

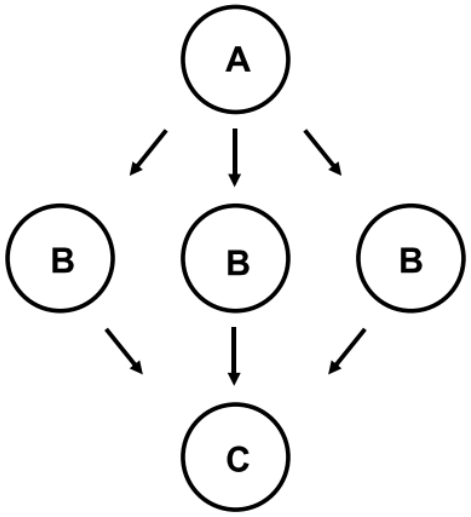
Fonksiyonel Paralelizm: Her süreç farklı bir "işlev" gerçekleştirir veya bağımsız olan farklı kod bölümlerini yürütür [29].



Şekil 3.1. Fonksiyonel paralelizm şeması [29].

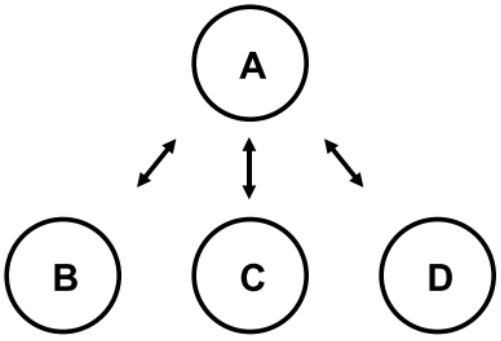
Veri Paralelliği: Her süreç aynı işi birbirinden farklı ve bağımsız veri parçaları üzerinde yapar. Genellikle fonksiyonel paralellikten daha ölçeklenebilirdir. OpenMP (Open Multi-Processing) ile yüksek düzeyde veya MPI (Message Passing Interface)

veya hibrit programlama gibi bir mesaj iletme kütüphanesi kullanılarak daha düşük bir seviyede programlanabilir [29].



Şekil 3.2. Veri paralelliği şeması [29].

Görev Paralelliği: Aynı veya farklı veri kümeleri üzerinde tamamen farklı hesaplamalar yapılabilen paralel bir programın karakteristiğidir. Bu, aynı hesaplamanın aynı veya farklı veri kümelerinde yürütüldüğü veri paralelliği ile çelişir. Görev paralellikleri genellikle bir problemin boyutu ile ölçeklendirilmez [29].



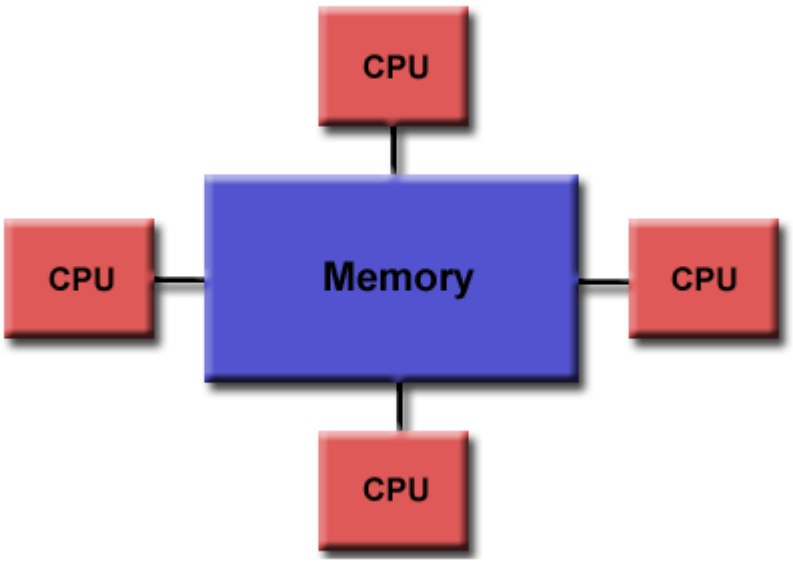
Şekil 3.3. Görev paralelliği şeması [29].

Boru Hattı Paralelliği: Her Sahne bir çözümün bir parçası üzerinde çalışır. Bir kademenin çıktısı bir sonraki kademenin girişidir. (Not: Bu paralellik tipi, her aşamada tamamlanması için aynı miktarda zaman alırsa en iyi şekilde çalışır) [29].

3.3. PARALEL BİLGİSAYAR BELLEK MİMARİLERİ

Modern bilgisayarları sınıflandırmanın faydalı bir yolu hafıza modelidir. Paralel bellek mimarilerine göre bilgisayarları 3 başlıkta inceleyebiliriz: paylaşımlı bellek mimarisi, dağıtık bellek mimarisi, hibrit bellek mimarisi [29].

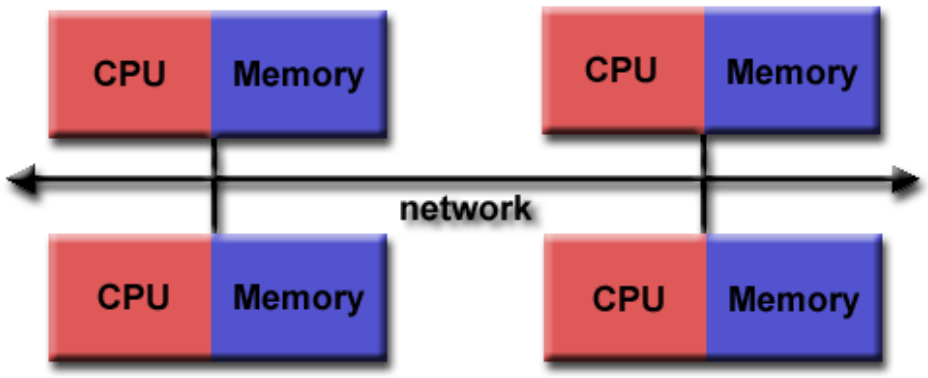
Paylaşımlı bellek: Paralel bilgisayarlar çok çeşitlidir, ancak çoğunlukla işlemcilerin genel adres alanı olarak tüm belleğe erişme kabiliyetleri ortaktır. Birden çok işlemci bağımsız olarak çalışabilir, ancak aynı bellek kaynaklarını paylaşabilir. Bir işlemci tarafından gerçekleştirilen bellek konumundaki değişiklikler, diğer tüm işlemciler tarafından görülebilir. Görevler arasında veri paylaşımı, belleğin CPU'lara yakınlığı nedeniyle hem hızlı hem de tek tiptir. Genel adres alanı, belleğe kullanıcı dostu bir programlama bakış açısı sağlar fakat tek bir adres alanı olduğu için veri trafiği fazladır ve veri erişiminde senkronizasyonun sağlanması gerekir [30].



Şekil 3.4. Paylaşımlı bellekli paralel bilgisayar mimarisi [30].

Dağıtık Bellek: Paylaşılan bellek sistemleri gibi, dağıtık bellek sistemleri de geniş çeşitlilik gösterebilir, ancak ortak bir özelliği paylaşırlar. Dağıtılmış bellek sistemleri, işlemci içi belleği bağlamak için bir iletişim ağı gerektirir. İşlemcilerin kendi yerel belleği vardır. Bir işlemcideki bellek adresleri başka bir işlemci ile

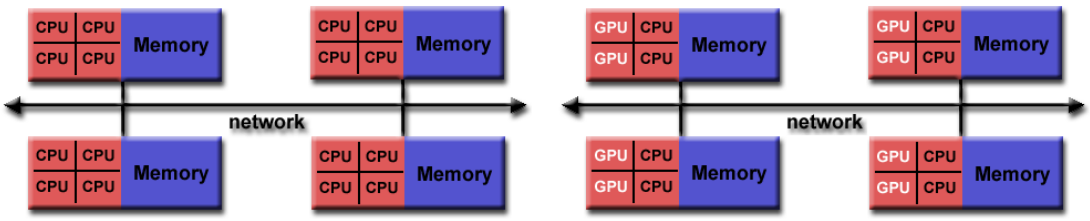
eşleşmez, dolayısıyla tüm işlemcilerde genel adres alanı kavramı yoktur. Her işlemcinin kendi yerel belleği olduğundan, bağımsız olarak çalışır. Yerel hafızasına yaptığı değişiklikler, diğer işlemcilerin hafızasını etkilemez. Bu nedenle, önbellek tutarlılığı kavramı geçerli değildir. Bir işlemcinin başka bir işlemcideki verilere erişmesi gerektiğinde, genellikle verilerin nasıl ve ne zaman iletildiğini açık bir şekilde tanımlamak programcının görevidir. Görevler arasında eşitleme aynı şekilde programcının sorumluluğundadır. Veri aktarımı için kullanılan ağ, Ethernet kadar basit olsa da, büyük ölçüde değişir. Her işlemci, kendi hafızasına müdahale olmadan ve global önbellek tutarlılığını korumaya çalışmakla uğraşmaksızın veriye hızla erişebilir ama var olan veri yapılarını, genel belleğe dayalı olarak bu bellek kuruluşuna eşlemek zor olabilir ve uzak bir düğümde bulunan verilere erişmek, düğümün yerel verilerine ulaşmaktan daha uzun sürer [30].



Şekil 3.5. Dağıtık bellekli paralel bilgisayar mimarisi [30].

Hibrit Bellek: Bugün dünyanın en büyük ve en hızlı bilgisayarları hem paylaşımlı hem de dağıtık bellek mimarilerini kullanmaktadır. Paylaşılan hafıza bileşeni, bir paylaşımlı bellek makinesi ve / veya grafik işlem birimi (GPU) olabilir. Dağıtılmış bellek bileşeni, sadece kendi belleğini bilen, başka bir makinedeki bellekle ilgili olmayan, çoklu paylaşımlı bellek / GPU makinelerinin ağ bağlantısıdır. Bu nedenle, ağ iletişimi, verileri bir makineden diğerine taşımak için gereklidir. Mevcut eğilimler, bu tür bellek mimarisinin hüküm sürmeye devam edeceğini ve gelecekte hesaplamanın üst seviyelerinde kullanımının artacağını gösteriyor. Artan

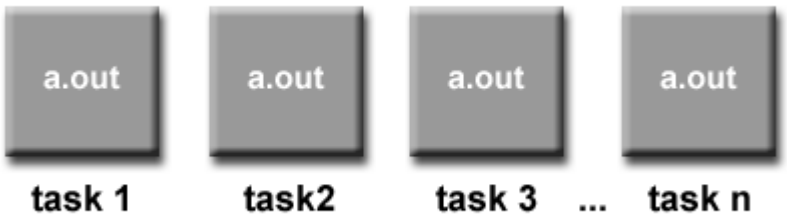
ölçeklenebilirlik, hibrit bellek mimarisi için önemli bir avantajdır fakat artan programcı karmaşıklığı önemli bir dezavantajdır [30].



Şekil 3.6. Hibrit bellekli paralel bilgisayar mimarisi [30].

3.4. PROGRAMLAMA MODELLERİ

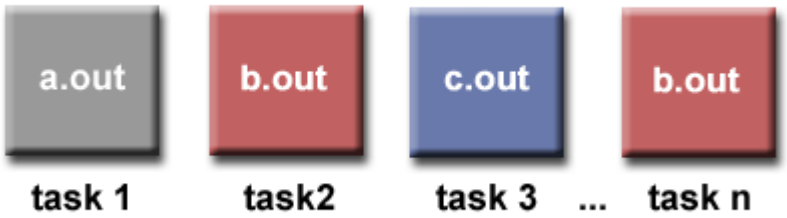
Tek Program Çoklu Veri (SPMD - Single Program Multiple Data): SPMD, daha önce bahsedilen paralel programlama modellerinin herhangi bir kombinasyonu üzerine kurulabilen "yüksek seviye" programlama modelidir. Tüm görevler aynı programın kopyasını aynı anda yürütür. Tüm görevler farklı veriler kullanabilir. SPMD programları genellikle, farklı görevlerin dallara ayrılmasına veya yalnızca yürütmek üzere tasarlandıkları programın bölümlerine koşullu olarak uygulanmasına izin vermek için programlanmış gerekli mantığa sahiptir. Yani, görevler mutlaka tüm programı yürütmek zorunda değildir. İleti geçişi veya karma programlama kullanan SPMD modeli, çok düğümlü kümeler için muhtemelen en yaygın kullanılan paralel programlama modelidir [30].



Şekil 3.7. Tek Program Çoklu Veri (SPMD - Single Program Multiple Data) [30].

Çoklu Program Çoklu Veri (MPMD - Multiple Program Multiple Data): SPMD gibi, MPMD aslında daha önce belirtilen paralel programlama modellerinin herhangi bir kombinasyonu üzerine kurulabilen bir "yüksek seviye" programlama modelidir.

Görevler aynı anda farklı programlar yürütebilir. Tüm görevler farklı verileri kullanabilir. MPMD uygulamaları, SPMD uygulamaları kadar yaygın değildir, ancak belirli problem türleri için daha uygun olabilir [30].



Şekil 3.8. Çoklu Program Çoklu Veri (MPMD - Multiple Program Multiple Data) [30].

3.5. PARALEL UYGULAMALAR

Bilim ve Mühendislik: Tarihsel olarak, paralel hesaplama, "bilgi işlemin en yüksek noktası" olarak kabul edilmiştir ve birçok bilim ve mühendislik alanında zor problemleri modellemek için kullanılmıştır:

1. Atmosfer, Toprak, Çevre
2. Fizik- uygulanan, nükleer, parçacık, yoğun madde, yüksek basınç, füzyon, fotonik
3. Biyoloji, Biyoteknoloji, Genetik
4. Kimya, Moleküler Bilimler
5. Jeoloji, Sismoloji
6. Makine Mühendisliği- protezden uzay aracına
7. Elektrik Mühendisliği, Devre Tasarımı, Mikroelektronik
8. Bilgisayar Bilimi, Matematik
9. Savunma, Silahlar

Endüstriyel ve ticari: Günümüzde ticari uygulamalar, daha hızlı bilgisayarların geliştirilmesinde eşit ya da daha büyük bir itici güç sağlamaktadır. Bu uygulamalar, çok miktarda verinin karmaşık yollarla işlenmesini gerektirir. Örneğin:

1. "Büyük Veri", veri tabanları, veri madenciliği
2. Yapay Zeka (AI)
3. Web arama motorları, web tabanlı iş hizmetleri
4. Tıbbi görüntüleme ve tanı
5. İlaç tasarımı
6. Finansal ve ekonomik modelleme
7. Ulusal ve çok uluslu şirketlerin yönetimi
8. Özellikle eğlence endüstrisinde gelişmiş grafikler ve sanal gerçeklik
9. Ağ bağlantılı video ve çoklu medya teknolojileri
10. Petrol arama

Küresel uygulamalar: Paralel hesaplama, çok çeşitli uygulamalarda dünya çapında yaygın bir şekilde kullanılmaktadır.

BÖLÜM 4

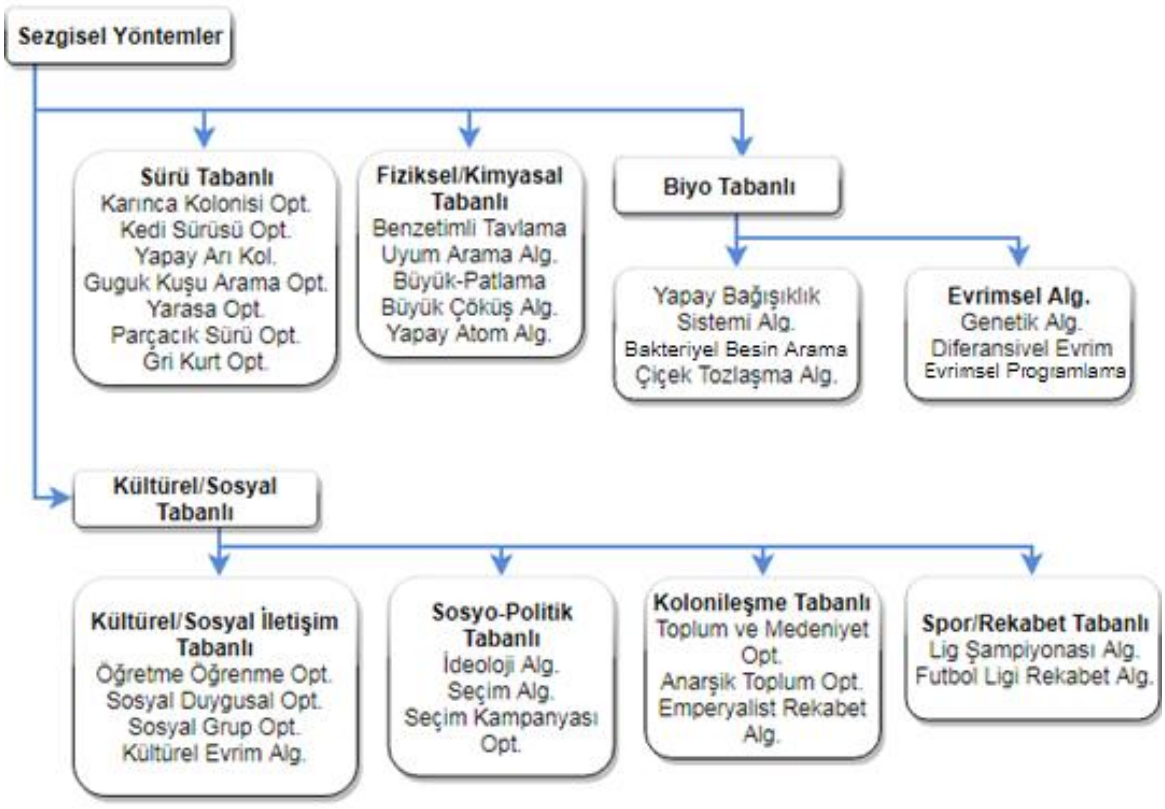
META-SEZGİSEL ALGORİTMALAR

Bilim, mühendislik, ekonomi ve iş dünyasında çok sayıda gerçek yaşam optimizasyonu problemi karmaşıktır ve çözülmesi zordur. Makul bir süre içinde tam olarak çözülemezler. Bu sınıf problemleri çözmek için yaklaşık algoritmaların kullanılması ana alternatiftir. Yaklaşık algoritmalar ayrıca iki sınıfa ayrılabilir: özel-sezgisel yaklaşım ve meta-sezgisel yaklaşım [31].

Özel sezgisel problemler bağımlıdır; belirli bir problem için tasarlanmıştır ve o probleme uygulanabilir, meta-sezgisel yaklaşım ise çok çeşitli optimizasyon problemleri için geçerli olan daha genel yaklaşık algoritmaları temsil eder. Herhangi bir optimizasyon problemini çözmek için uyarlanabilirler. Meta-sezgisel yaklaşımlar, zor olduğuna inanılan problemlerin örneklerini genellikle geniş çözüm arama alanını keşfederek çözer. Bu algoritmalar, alanın etkili boyutunu azaltarak ve bu alanı verimli bir şekilde araştırarak bunu gerçekleştirir [31]. Meta-sezgisellik üç ana amaca hizmet eder: Problemleri daha hızlı çözme, büyük problemleri çözme ve sağlam algoritmalar elde etme. Bunun yanı sıra, meta-sezgisel algoritmaların tasarımı ve uygulaması basittir ve çok esnektirler.

Birçok farklı doğadan ilham alınmış meta-sezgisel optimizasyon algoritmaları vardır. Bu algoritmalar farklı sınıflara ayrılabilir. Sürü zekâ algoritmaları Karınca Kolonisi Optimizasyon Algoritması (ACO) [32], Kedi Sürüsü Optimizasyon Algoritması [33], Yapay Arı Kolonisi Algoritması [34], Guguk Kuşu Arama Optimizasyonu Algoritması [35], Yarasa Optimizasyonu Algoritması [36], Parçacık Sürü Optimizasyonu Algoritması [37], Gri Kurt Optimizasyonu Algoritması [38]. Biyo tabanlı algoritmalar Yapay Bağışıklık Sistemi Algoritması [39], Bakteriyel Besin Arama Optimizasyonu Algoritması [40], Çiçek Tozlaşma Algoritması [41] ve bir alt kategori olarak evrimsel algoritmalar Genetik Algoritma [42], Diferansiyel Evrim

[43], Evrimsel Programlama [44] gibi algoritmalardır. Başka bir kategori de kültürel/ sosyal algoritmalardır; bunlar İdeoloji Algoritması [45], Seçim Algoritması [46], Seçim Kampanyası Optimizasyon Algoritması [47], Futbol Ligi Rekabet Algoritması [48], Öğretme Öğrenme Tabanlı Optimizasyonu [49], Anarşik Toplum Optimizasyonu [50],Toplum ve Medeniyet Optimizasyonu [51], Sosyal Duygusal Optimizasyon Algoritması [52], Lig Şampiyonası Algoritması [53], Emperyalist Rekabet Algoritması [54], Sosyal Grup Optimizasyonu [55], Kültürel Evrim Algoritması [56]. Doğadan ilham alan bir başka algoritma da Benzetimli Tavlama [57], Armoni Arama Algoritması [58], Büyük-Patlama Büyük Çöküş Algoritması [59] ve Yapay Atom Algoritması [60]gibi fiziksel / kimyasal sistem tabanlı algoritmalardır. Şekil 4.1'de sezgisel algoritmaların sınıflandırması verilmiştir.



Şekil 4.1. Sezgisel algoritmaların sınıflandırılması.

4.1. SÜRÜ TABANLI SEZGİSEL YÖNTEMLER

Bir dizi hayvan türünün sergilediği kolektif veya sürü şeklinde olan davranışından ilham alan sürü tabanlı algoritmalardır. Paylaşılan hedefe ulaşmak için bir sürü tarafından sergilenen toplu zekâ, sürü tabanlı algoritmaların arkasındaki temel fikri oluşturur. Bir ortamda çok sayıda homojen ajan, istenen hedefe ulaşmak için karşılıklı işbirliği yoluyla çalışır.

Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - ACO), Dorigo ve ark. [32] tarafından ortaya atılmış, ilham kaynağı gerçek karıncaların yiyecek arama davranışı olan bir yöntemdir. Yiyecek ararken, karıncalar başlangıçta yuvalarını çevreleyen alanı rastgele bir şekilde araştırırlar. Bir karınca yiyecek kaynağı bulur bulmaz, yiyeceğin miktarını ve kalitesini değerlendirir ve bir kısmını da yuvaya taşır. Geri dönüş sırasında karınca, yerde kimyasal bir feromon izi biriktirir. Yiyeceğin miktarına ve kalitesine bağlı olarak biriken feromon miktarı diğer karıncaları besin kaynağına yönlendirecektir. Feromon yollar aracılığıyla karıncalar arasındaki dolaylı iletişim, yuva ve besin kaynakları arasında en kısa yolların bulunmasını sağlar.

Yapay Arı Kolonisi (Artificial Bee Colony - ABC) algoritması ilk kez Derviş Karaboğa [61] tarafından önerilmiştir. Algoritma, besin kaynaklarını bulmak için bal arıları sürüsünün yiyecek kaynağı arama yönteminden esinlenmiştir. Bu kaynakları başarılı bir şekilde bulmak için bilgiyi paylaşan üç farklı bal arısı grubu vardır. İlk olarak, gıda kaynağını faydalanan işçi arı var. İkincisi, sürekli olarak bir gıda kaynağı arayan kâşif arı ve son olarak yuvada bekleyen gözcü arılar vardır.

Gri Kurt Optimizasyonu (Grey Wolf Optimization - GWO) algoritması gri kurtların kendilerine özgü yiyecek arama ve avlanma karakteristiğini taklit eder [38]. GWO, gri kurtların dört seviyeli sosyal hiyerarşisini, ilk olarak α, ikinci olarak da β, üçüncü δ ve son seviyesinde ω kurtları kabul etmiştir. α kurtlar, bütün kurtların yöneten lider kurtlardır. Ayrıca tüm avlanma sürecini kontrol etmek, avlanma, disiplini sürdürme, uyku ve uyanma gibi her türlü kararı tüm kurtlar için almakla yükümlüdür. α olmak için en iyi aday olan β kurtlarıdır, diğer kurtlardan geribildirim alır ve bu geri bildirimleri lidere iletir. Kurtların üçüncü seviyesi δ kurtlar kurtların güvenlik ve

bütünlüğünü sağlamaktan sorumlu olan ve son kurt olan ω kurtlarına hükmeden kurtlardır.

Guguk Kuşu Arama (Cuckoo Search) algoritması bazı kuş türlerinin ve meyve sineklerinin Le'vy uçuş davranışı ile bazı guguk kuşu türlerinin zorunlu parazit kuluçkaya yatma davranışının kombinasyonuna dayanmaktadır.

Kedi Sürüsü Optimizasyonu (Cat Swarm Optimization - CSO), kedilerin avlanma modelinden esinlenerek geliştirilmiştir [33]. Zamanlarının çoğunu dinlenmeye rağmen, kediler her zaman tetiktedir ve çevreleri ve çevrelerindeki nesneler hakkında meraklıdır. Bu davranış, kedilere avlarını bulma ve onları avlama konusunda yardımcı olur. Dinlenmeye ayırdıkları zamanla karşılaştırıldığında, kediler enerjilerini korumak için av peşinde koşmaya çok az zaman harcarlar. Modellenen algoritmada iki durum vardır. Bunlar; kedilerin dinlenme davranışını modelleyen "arama durumu" ve avlarını kovalama davranışını modelleyen "izleme durumu"dur. Kediler avlarını yakalayabilmek için en iyi konuma ulaşmaya çalışır.

Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO) yöntemi Kennedy ve Eberhart tarafından kuş ve balık sürülerinin toplanma gibi organizmaların sosyal davranışını temel alarak geliştirilmiştir [37]. PSO algoritması sadece optimizasyon için bir araç değil, aynı zamanda sosyal davranış ilkelerine dayanan insan ve yapay ajanların sosyo-bilişini temsil eden bir araçtır. Bir optimizasyon aracı olarak PSO, partikül denilen bireylerin zamanla pozisyonlarını (durumlarını) değiştirdikleri nüfus tabanlı bir arama prosedürü sağlar. Bir PSO sisteminde, parçacıklar çok boyutlu bir arama alanında uçarlar. Uçuş sırasında, her parçacık kendi konumunu kendi tecrübesine göre ve komşuların deneyimine göre, kendisi ve komşuları tarafından karşılaşılan en iyi konumdan yararlanır.

Yarasa Algoritması (Bat Algorithm BA), yarasaların ses ile yer/yön belirleme davranışına dayanmaktadır. Mikrobatlar sesin yankılanmasını özelliğini kullanarak avlarını bulabilir ve tamamen karanlıkta bile farklı böcek türlerini ayırt edebilirler. BA'da, yarasalar avlarını araştırıp bulurken yarasaların frekans, ses yüksekliği ve nabız atış hızı değişir. En iyiyi seçme işlemi, belirli durdurma kriterleri karşılanana

kadar devam eder. Bu aslında yarasa sürüsünün dinamik davranışını kontrol etmek için frekans ayarlama tekniğini kullanır ve keşif ve sömürü arasındaki denge BA'da algoritmaya bağlı parametreler ayarlanarak kontrol edilebilir.

4.2. FİZİKSEL/KİMYASAL TABANLI SEZGİSEL ALGORİTMALAR

Doğadaki fiziksel veya kimyasal olaylardan esinlenerek oluşturulmuş sezgisel algoritmalardır.

Benzetimli Tavlama (Simulated Annealing - SA), katı bir madde ile fiziksel tavlama işlemine olan benzerliğinden ötürü bu şekilde adlandırılmıştır [57]. Bu işlemde, kristalli katı ısıtılır ve daha sonra kararlı kristal örgü biçimine yani, minimum kafes enerji durumuna ulaşana kadar çok yavaş soğumaya bırakılır, böylece katı üzerinde kristal kusurları olmaz. Soğutma yeterince yavaşsa, nihai biçim, bu gibi üstün yapısal bütünlüğe sahip bir katı ile sonuçlanır. Benzetimli tavlama, bu tür bir termo-dinamik davranış ile ayrık bir optimizasyon problemi için küresel minimumlar arasındaki bağlantıyı kurar. Ayrıca, böyle bir bağlantıdan yararlanmak için algoritmik bir araç sağlar.

Armoni Arama (Harmony Search) algoritması caz müzisyenlerinin doğaçlama sürecinden esinlenmiştir[62]. Bu sürecin amacı mükemmel armoni durumunu araştırmaktır. Müzikteki bu armoni arayışı, bir optimizasyon sürecindeki optimizasyonu bulmak ile benzerdir. HS'de her müzisyen bir karar değişkenine karşılık gelir, enstrümanın perde aralığı ise karar değişkeninin değer aralığına karşılık gelir. Müziksel uyum, çözüm vektörüne karşılık gelir ve kitlenin estetiği, nesnel işlevlere karşılık gelir. Müziksel uyumun belli bir zaman sonra geliştirilmiş olacağı gibi, çözüm vektörü de yinelemeli olarak sonucu iyileştirir.

Büyük Patlama Büyük Çöküş (Big Bang–Big Crunch – BB-BC) algoritması evrenin evrim teorilerinden birine dayanan bir optimizasyon yöntemidir [59]. Büyük Patlama evresinde, enerji yayılımı düzensizlik yaratır ve rastlantısallık bu fazın ana özelliğidir; Big Çöküş fazında ise rasgele dağıtılmış parçacıklar bir sıraya sokulur. Bu teoriden esinlenen BB-BC, fazında rasgele noktalar üretir ve bu noktaları Büyük

Çöküş fazında bir kütle merkezi veya minimum maliyet yaklaşımıyla tek bir temsili noktaya küçültür.

Yapay Atom Algoritması (Artificial Atom Algorithm - A^3) Ali Karcı [60] tarafında ortaya atılan, atomların kimyasal bileşik oluşturma işlemlerden esinlenen bir meta-sezgisel optimizasyon algoritmasıdır. A^3, kimyasal iyonik bağ ve kimyasal kovalent bağ süreçlerinin modellenmesi ile geliştirilmiştir.

4.3. BİYO TABANLI SEZGİSEL ALGORİTMALAR

Biyo tabanlı sezgisel algoritmalar genellikle doğadaki oluşumları taklit ederek ortaya çıkartılmış algoritmalardır ve literatürde birçok farklı biyo tabanlı algoritma bulunmaktadır.

Yapay Bağışıklık Sistemi (Artificial Immune System) canlıların biyolojik bağışıklık sisteminin ilkelerinden esinlenilmiştir [39]. Doğada, bir hayvan antijenlere maruz kaldığında, organizmayı savunmak için etkili bir bağışıklık tepkisi gelişir. Bunu yapmak için, antijenler ile mücadele etmek için antikorlar üretilir. En iyi antikorlar kopyalanır, hipermutasyona uğratılır ve seçilir, diğer yandan popülasyonun çeşitliliğini arttırmak için rastgele antikorlar (kemik iliği tarafından üretilir) oluşturulur. Böylece, organizmaya yine bu antijen tarafından saldırılırsa, daha hızlı bir bağışıklık tepkisi gelişir. Bu adaptasyon becerisi, hipermutasyon yoluyla klonal seçim ve benzerlik olgunlaşması veya daha basit olarak klonal seçim olarak bilinir.

Bakteriyel Besin Arama Optimizasyonu Algoritması (Bacterial Foraging OptimizationAlgorithm - BFOA) Escherichia Coli (E. Coli) bakterilerinin besin arama sürecinden esinlenilerek ortaya çıkartılmış bir algoritmadır [40]. Bakteriler çevredeki kimyasal maddeleri algılar (besinler gibi) ve belirli sinyallere doğru hareket eder ve belirli sinyallerden uzaklaşır. BFOA'da ilk olarak tüm yapay bakteriler, çok boyutlu araştırma alanı içinde rastgele konumlara yerleştirilir ve daha sonra bu çözümlerin maliyetlerini veya uygunluğunu ölçerler ve bu uygunluğu kullanarak çözüm konumları arasında küresel optimum çözümü bulurlar. Daha sonra,

her bir bakteri üzerinde belirli algoritmaya bağlı işlemler gerçekleştirerek daha iyi çözümler üretilir.

Çiçek Tozlaşma Algoritması (Flower Pollination Algorithm) çiçeklerin tozlaşma sürecinden esinlenerek elde edilmiştir [41]. Çiçeklerin tozlaşmasında 2 durum modellenmiştir; Biyotik çapraz tozlaşma yani arı, böcek gibi canlıların polenleri başka çiçekleri taşımasıyla gerçekleşir, biyotik olmayan kendi kendine oluşan tozlaşma ise çiçeğin kendi polenleri ile tozlaşmasıyla gerçekleşir. Bu küresel ve yerel tozlaşma süreçleriyle en iyi sonuca ulaşılmaya çalışılır.

4.3.1. Evrimsel Algoritmalar

Evrimsel algoritmalar, biyolojik evrim ve doğal seleksiyon süreçlerinden ilham alır.

Genetik Algoritma (Genetic Algorithm - GA) Holland ve öğrencileri tarafından Michigan Üniversitesi'nde 60'lar ve 70'lerde geliştirilen genetik algoritma [42], türlerin kromozomal evriminin biyolojik sürecini taklit eden bir problem çözme stratejisidir. GA, çözülmesi gereken belirli bir problem için bir dizi potansiyel çözüm kümesidir, ki buna popülasyon denir, bu potansiyel çözümlerin her birine kromozomlar veya bireyler denir. GA, bu kromozomlar üzerinde doğal seçilim, çaprazlama ve mutasyon evrelerini uygulayarak daha iyi bir gen elde etmeyi hedefler.

Diferansiyel Evrim (Differential Evolution - DE) algoritması, sürekli etki alanlarındaki optimizasyon problemleri için Rainer Storn ve Kenneth Price [43] tarafından geliştirilen bir evrimsel programlama dalıdır. DE'de her değişkenin değeri gerçek bir sayı ile temsil edilir. DE'nin avantajları basit yapısı, kullanım kolaylığı, hız ve sağlamlığıdır. DE, gerçek değerli değişkenlerle problemleri çözmek için en iyi genetik tip algoritmalardan biridir. Diferansiyel Evrim algoritması pratik uygulamalar için hemen erişilebilen iyi bir yöntemdir.

Evrimsel Programlama (Evolutionary Programming - EP) Fogel, Owens, ve Walsh tarafından geliştirildi [44]. EP'de aday çözümler sonlu durum makineleri olarak

temsil edilmiştir ve rastgele mutasyona tâbi tutularak ve en uygun çözümü seçerek gelişen bir tekniktir. Evrim stratejilerinde olduğu gibi, rastlantısal mutasyon tek varyasyon kaynağıdır.

4.4. KÜLTÜREL/SOSYAL TABANLI SEZGİSEL ALGORİTMALAR

Kültürel/sosyal tabalı algoritmalar insan davranışlarında görülen sosyal ve kültürel etkileşimlere dayanarak oluşturulmuş sezgisel algoritmalardır.

4.4.1. Kültürel/Sosyal İletişim Tabanlı

Öğretme Öğrenme Tabanlı Optimizasyon (Teaching–Learning-Based Optimization - TLBO) yöntemi, bir öğretmenin öğrencilerin üzerindeki etkisi üzerinde çalışır [49]. Diğer doğadan esinlenen algoritmalar gibi, TLBO da popülasyona dayalı bir yöntemdir ve küresel çözüme ulaşmak için bir çözüm popülasyonu kullanır. Nüfus, bir grup öğrenci veya öğrencilerin bulunduğu bir sınıf olarak kabul edilir. TLBO süreci iki kısma ayrılmıştır: ilk kısım "Öğretmen Aşaması" ve ikinci kısım "Öğrenci Aşaması" ndan oluşmaktadır. 'Öğretmen Aşaması' öğretmenden ders çıkarmak ve 'Öğrenen Aşaması', öğrenciler arasındaki etkileşim vasıtasıyla öğrenimin gerçekleşmesi anlamına gelir.

Sosyal Duygusal Optimizasyon Algoritmasında (Social Emotional Optimization Algorithm - SEOA) sosyal bilimlerden esinlenmiş bir meta-sezgisel algoritmadır [52]. Her birey bir kişiyi temsil ederken, problem alanındaki tüm noktalar durum toplumu oluşturur. Bu sanal dünyada, tüm bireyler daha yüksek sosyal statüyü aramayı amaçlamaktadır. Bu nedenle, kişisel statüyü arttırmak için işbirliği ve rekabet yoluyla iletişim kurarlar ve kararlarını duygularına göre alırlar. En yüksek puanı alan kişi ise nihai çözüm olarak kazanan olacaktır.

Sosyal Grup Optimizasyon (Social Group Optimization - SGO) nüfus tabanlı bir optimizasyon tekniğidir [55]. İnsanın karmaşık bir problemi çözerken gösterdeği toplumsal davranış kavramından ilham alır. SGO'da, her kişi (bir aday çözüm), farklı seviyede problem çözme kapasitesine sahiptir. SGO için nüfus, bir grup insan olarak

kabul edilir (aday çözümler). Her bir kişi bilgi edinir ve böylece problemi çözmek için bir miktar kapasiteye sahiptir ve bu, "uygunluk" a karşılık gelir.. SGO prosedürü iki bölüme ayrılmıştır. İlk kısım “iyileştirme aşaması” ndan oluşur; ikinci kısım “edinme aşaması” ndan oluşur. “İyileştirme aşamasında”, gruptaki her bir kişinin bilgi seviyesi, gruptaki en iyi insanın etkisi ile güçlendirilir. Gruptaki en iyi kişi, sorunu çözmek için en yüksek bilgi ve kapasiteye sahip olan kişidir. “Edinme aşamasında”, her bir kişi kendi bilgisini, gruptaki başka bir kişi ile karşılıklı etkileşim ile geliştirir.

Kültürel Evrim Algoritması (Cultural Evolution algorithm - CEA) 19. yüzyılda egemen olan sosyo-kültürel geçişin ana evrim mekanizması olarak ortaya çıkarılan çizgisel evrimi taklit eder [56]. CEA, sosyo-kültürel geçiş olgusuna uymayı amaçlamakta, farklı kültürel nüfus evrimi kavramını taklit etmekte ve kültürel türler arasında iletişim, enfeksiyon ve öğrenme gibi davranışlarını içermektedir. Bu sistemde düşünceleri bilgi arama şemasına taşıyan iki mekanizma vardır. Bunlar, popülasyondaki bir sorunu ve nüfusun yenilik modunu temsil eden bir bireyi tanımlamanın yoludur. Kültürel türlerin evrim modları, ortak görüş (veya ana akım değeri olarak adlandırılır), bireysel öğrenme, yenilikçi öğrenme ve kendini geliştirme olarak kategorize edilebilir.

4.4.2. Sosyo-Politik Tabanlı

İdeoloji Algoritması (Ideology Algorithm - IA) bazı inançların, bir toplumdaki bireylerin hedeflerine ulaşmaları için rehber olmaları fikrinden ilham alır [45]. IA, bu düşünceyi, bireylerin politik ideolojilerini izledikleri ve kendi siyasi partilerinin üyeleriyle ve diğer siyasi partilerin liderleri ile üstünlük kurmak için rekabet ettikleri politik bir senaryo aracılığıyla ortaya koymaktadır.

Seçim Algoritması (Election Algorithm - EA) başkanlık seçiminden esinlenen bir optimizasyon ve arama tekniğidir [46]. EA, popülasyon olarak bilinen bir dizi çözümle çalışan yinelemeli popülasyon tabanlı bir algoritmadır. Nüfusun her bir bireyine kişi denir ve ya bir aday ya da seçmen olabilir. Bu kişiler çözüm alanında bir dizi seçim partisini oluştururlar. Reklam kampanyası bu algoritmanın temelini

oluşturur ve üç ana adımı içerir: Olumlu reklam, olumsuz reklam ve koalisyon. Olumlu reklam sırasında, adaylar kendilerini olumlu imaj ve niteliklerini güçlendirerek kendilerini tanıtır. Olumsuz reklamda, adaylar popülaritesini artırmak ve rakiplerini karalamak için birbirleriyle rekabet eder. Bazı durumlarda, benzer fikirlere sahip adaylar, birleşik partinin başarı şansını arttırmak için bir araya gelebilirler. Reklamlar, insanların küresel optimum olan bir çözüm alanı durumuna kavuşmasına neden olur. Bütün bu çabalar seçim gününe (durma koşulu) kadar sürer. Seçim gününde en çok oyu alan aday kazanan olarak ilan edilir ve problem için en iyi çözümün karşılığıdır.

Seçim Kampanyası Optimizasyon (Election Campaign Optimization – ECO) algoritması seçim kampanyası sırasında siyasi adayların tavırlarına dayanarak geliştirilmiştir [47]. Seçim, adayların bir dizi kampanya ile seçmenlerden azami destek almasını gerektiren bir süreçtir ve optimizasyon, mümkün olan çözümlerin en iyisini bulmaktır. Seçmenler daha iyi prestijli (daha iyi işlev değeri) adaya oy vermekten ister ve nihayetinde en güçlü seçim adayı seçmenlerin en yüksek destekleği kazanır.

4.4.3. Kolonileşme Tabanlı

Toplum ve Medeniyet Optimizasyon (Society and Civilization Optimization - SCO) algoritması, toplum bireyleri arasında görülen insanın sosyal davranışlarından ilham alır. Bir toplumdaki bireyler genel davranışlarını geliştirmek için birbirleriyle etkileşir ve bu tür toplumlar arasında işbirlikçi etkileşim uygarlığı temsil eder.

Anarşik Toplum Optimizasyon (Anarchic Society Optimization - ASO) üyelerin kendi durumlarını iyileştirmek için anarşik olarak davrandıkları bir sosyal gruplamadan esinlenilmiştir [50]. ASO'nun temeli, değişken, maceracı, hoşnutsuzluk durumunda, sıklıkla irrasyonel davranan ve keşif evresinde ziyaret ettikleri aşağı pozisyonlara doğru ilerleyen bir grup bireydir. Üyeler arasındaki anarşik davranış düzeyi, üyelerin durumları arasındaki farkın düzeyi arttıkça yoğunlaşmaktadır. Bu anarşik üyeleri kullanarak, ASO çözüm alanını mükemmel bir şekilde araştırır ve yerel optimum tuzaklara düşmekten kaçınır.

Emperyalist Rekabet Algoritması (Imperialist Competitive Algorithm - ICA) emperyalist uluslar arasında görülen zayıf koloni ve imparatorluklara sahip olmak için yarışan sosyopolitik davranışları taklit eden bir algoritmadır [54]. Bu emperyalist rekabet sonunda, daha güçlü ve başarılı emperyalist imparatorlukların güçlendirmesiyle sonuçlanır; zayıf imparatorluklar yavaş yavaş çökerken, daha iyi olan çözümlere doğru ilerleme kaydedilir.

4.4.4. Spor/Rekabet Tabanlı

Lig Şampiyonası Algoritması (League Championship Algorithm - LCA), yapay takımların birkaç hafta boyunca yapay bir ligde oynadığı bir şampiyonluk ortamını taklit etmeye çalışan(iterasyonlar) sürekli küresel optimizasyon için stokastik popülasyon tabanlı bir algoritmadır [53]. Her hafta lig programı verildiğinde, bir grup bireyin oluşturduğu takımlar çiftler halinde oynarlar. Her bir takımın geliştirdiği takım formasyonu / düzeni (çözüm) ile birlikte oyunun sonucu, oynama gücü (kondisyon değeri) göz önünde bulundurulduğunda kazanma veya kaybetme (veya bağlama) açısından belirlenir. Yapay bir eşleşme analizinin modellenmesiyle, her bir takım bir sonraki haftada yarışması için oluşumunda gerekli değişiklikleri (yeni bir çözümün üretilmesi) tasarlar ve şampiyonluk birkaç mevsim (durma koşulu sağlanana kadar) devam eder.

Futbol Ligi Rekabet (Soccer League Competition - SLC) algoritması meta-sezgisel optimizasyon tekniğidir ve optimizasyon problemlerini ayrık veya sürekli uzayda ele almak için başarıyla kullanılmıştır [48]. SLC'nin temel fikri profesyonel futbol liglerinden esinlenerek takımlar ve oyuncular arasındaki yarışmalara dayanmaktadır. Nüfus bireyleri veya oyuncuları iki türdedir: Sabit oyuncular ve birlikte bazı takımlar oluşturan yedekler. Lig tablosunda en üst sıradaki pozisyonlara sahip olmak için takımlar arasındaki yarışma ve kişisel gelişim için her takımdaki oyuncular arasındaki iç yarışmalar nüfustaki bireylerin küresel optimumlara yakınlaşması için kullanılır.

Gezgin Satıcı Problemi (GSP), N'de bir üstel zaman gerektiren bir kombinatoryal optimizasyon problemidir [5] ve klasik matematiksel tekniklerle daha büyük

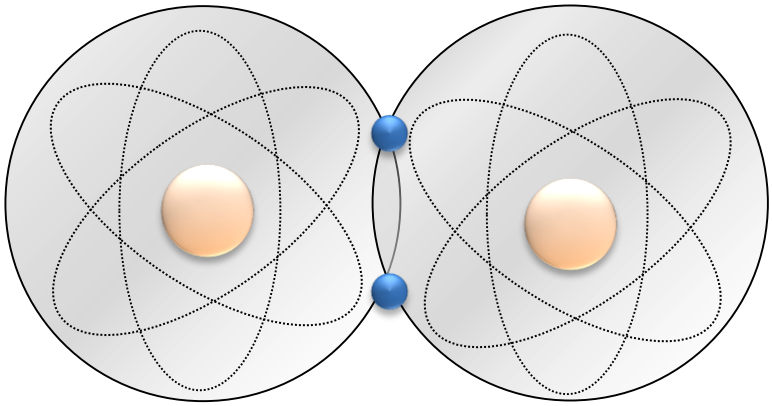
problemleri çözmek imkansızdır, bu nedenle, sezgisel algoritmalar bu problem için daha uygundur. Optimizasyon algoritmalarının çoğu GSP'ye uygulanmış ve verimlilikleri karşılaştırılmıştır.

BÖLÜM 5

YAPAY ATOM ALGORİTMASI (A^3)

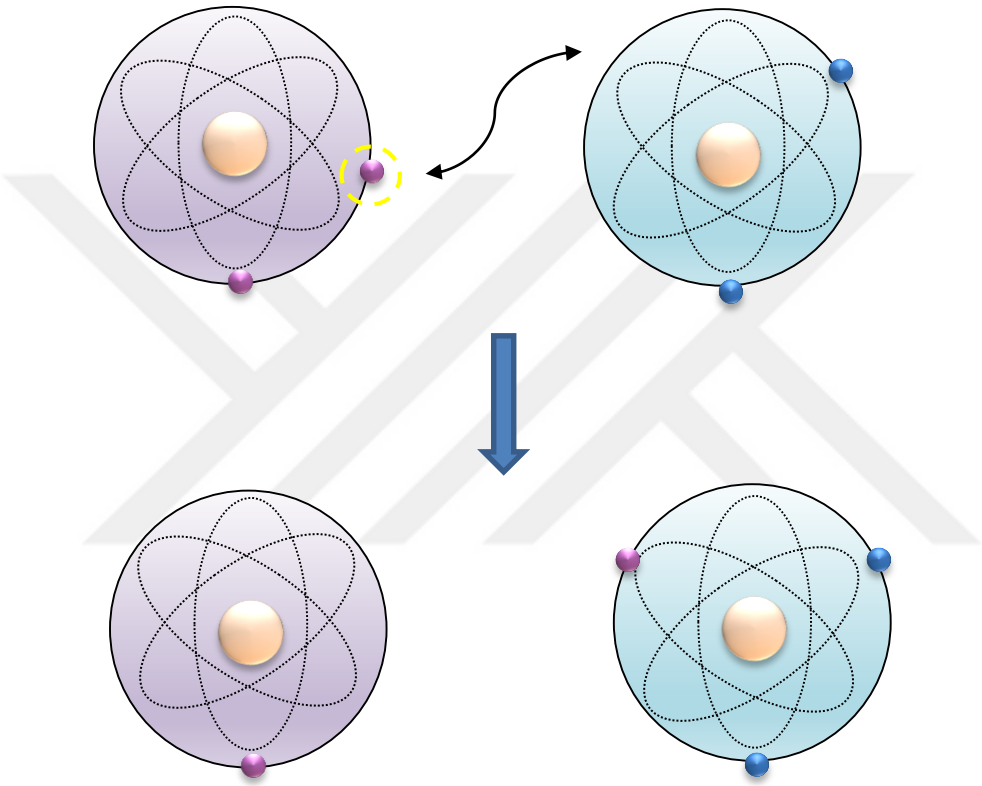
Yapay atom algoritması (A^3) [60] ilk defa Ali Karcı tarafından ortaya atılmış olan, kimyasal bileşiklere dayanan bir meta-sezgisel algoritmadır. Bilimsel ve mühendislik problemlerinin optimizasyonu için Yapay Atom Algoritması kimyasal bileşiklerin oluşum süreci taklit etmektedir. Atom seviyesinde gerçekleşen iyonik bağın ve kovalent bağın kimyasal oluşum süreçleri birbirinden ayrı olarak taklit edilir. Bu iki süreç, algoritmanın temel iki işlemini oluşturmaktadır. Yapay Atom Algoritmasında problem bir atom olarak düşünülür ve buna göre her bir parametre bir elektron olarak kabul edilir.

Kimyada kovalent bağ işlemi iki atom arasında, elektronların bazılarının ortak olarak kullanılması sonucunda oluşan bağdır. Şekil 5.1'de iki atom arasında oluşan kovalent bağ gösterilmektedir. Yapay atom algoritmasında iki atom arasında gerçekleşen kovalent bağ işlemi ise, elektron etki değeri daha iyi olan atomun elektronunun iki atom tarafından ortaklaşa kullanılmasıdır, yani daha iyi etki değerine sahip olan elektron diğer atoma kopyalanır.



Şekil 5.1. Kovalent bağın gösterimi.

İyonik bağ kimyada atomlara arasında elektron alışverişi ile gerçekleşen bağdır. Şekil 5.2'de gösterildiği gibi atomdan bir elektronun koparılması ve diğer atoma eklenmesi ile gerçekleşir. Yapay Atom Algoritmasında iyonik bağ işlemi düşük elektron etki değerine sahip olan elektronların çözüm kümesinden çıkartılarak yerine yeni elektronların çözüm kümesine eklenmesiyle gerçekleşmektedir [60].



Şekil 5.2. İyonik bağın gösterimi.

5.1. TEMEL KAVRAMLAR VE TANIM

Elektron: Her bir parametre değerinin sonuç üzerinde etkisi bulunur, bunlara elektron denir.

Atom: Tüm elektronların oluşturduğu, amaç fonksiyonuna etki eden parametreler kümesine atom denir ve her bir atom olası bir çözümdür. Şekil 5.3'te n adet elektron bulunduran bir atomun temsili şekli verilmiştir [60].

Şekil 5.3. Atomun gösterimi, E bir elektronu temsil eder [60].

Atom Kümesi: İlk etapta atomlar rastgele üretilmektedir ve bu atomların oluşturduğu kümeye atom kümesi denir. Atom sayısı problemin boyutuna uygun olacak şekilde belirlenir.

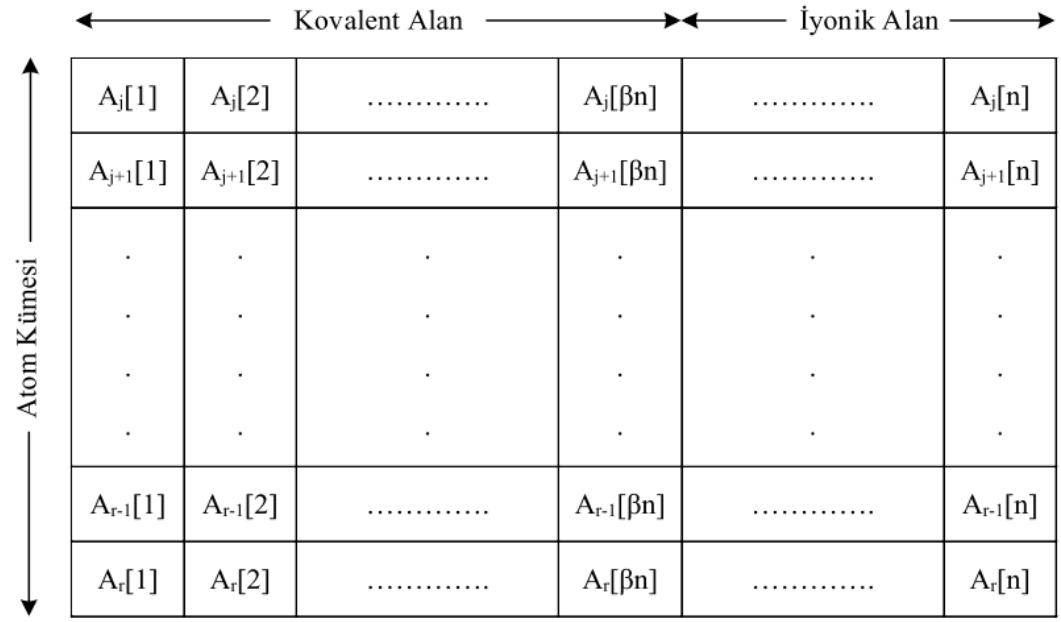
İyonik Oran (α): İyonik bağ operatörünün atom kümesinde nereye uygulanacağını bu katsayı belirler.

Kovalent Oran (β): Kovalent bağ operatörünün atom kümesinde nereye uygulanacağını belirleyen katsayıdır. İyonik oran ile kovalent oranın toplamı 1'e eşittir ($\alpha + \beta = 1$).

İyonik Alan (İA): İyonik bağ operatörünün atom kümesinde uygulandığı alandır. İyonik alandaki parametre sayısı n is, αn tane parametreye iyonik bağ operatörü uygulanmaktadır.

Kovalent Alan (KA): Kovalent bağ operatörünün atom kümesinde uygulandığı alandır. Kovalent alandaki parametre sayısı n is, βn tane parametreye iyonik bağ operatörü uygulanmaktadır. İyonik alandaki parametre sayısı ve kovalent alandaki parametre sayısının toplamı atom kümesindeki atom sayısına eşittir ($\alpha n + \beta n = n$).

Şekil 5.4.'de elektron sayısı n, atom sayısı r olan bir atom kümesinin iyonik alan ve kovalent alan ile beraber gösterimi bulunmaktadır [63].



Şekil 5.4. Atomun, Kovalent Alanın ve İyonik Alanın temsili gösterimi [64].

Her atom başlangıçta sorunun rastgele bir çözümü şeklindedir. İşlem birden fazla atomla başlar ve bu çözümler Atom Seti olarak adlandırılır. Karar değişkenlerinin her biri elektron olarak düşünülebilir. Karar değişkenleri dizisi atomu oluşturur ve Genetik Algoritmada bulunan kromozom olarak düşünülebilir, atomlardan oluşan matrisler ise Genetik Algoritmadaki popülasyon olarak düşünülebilir. A^3 algoritması şu şekildedir [65]:

1. Rastgele Atom Seti (A_0) üret
2. Bulunduğu Atom Seti içinde her elektronun kendi atom etkisini hesapla
3. $i \leftarrow 0$
4. Durma koşulu sağlanana kadar aşağıdakileri tekrarla
5. A_i'ye Kovalent Bağ işlemi uygula
6. // $B \leftarrow KovalentBağ(A_i - 1)$
7. A_i'ye İyonik Bağ işlemi uygula
8. // $A_i + 1 \leftarrow İyonikBağ(B)$
9. İA içindeki elektron etkilerini hesapla
10. Her atomun amaç fonksiyonu değerini hesapla
11. $i \leftarrow i + 1$

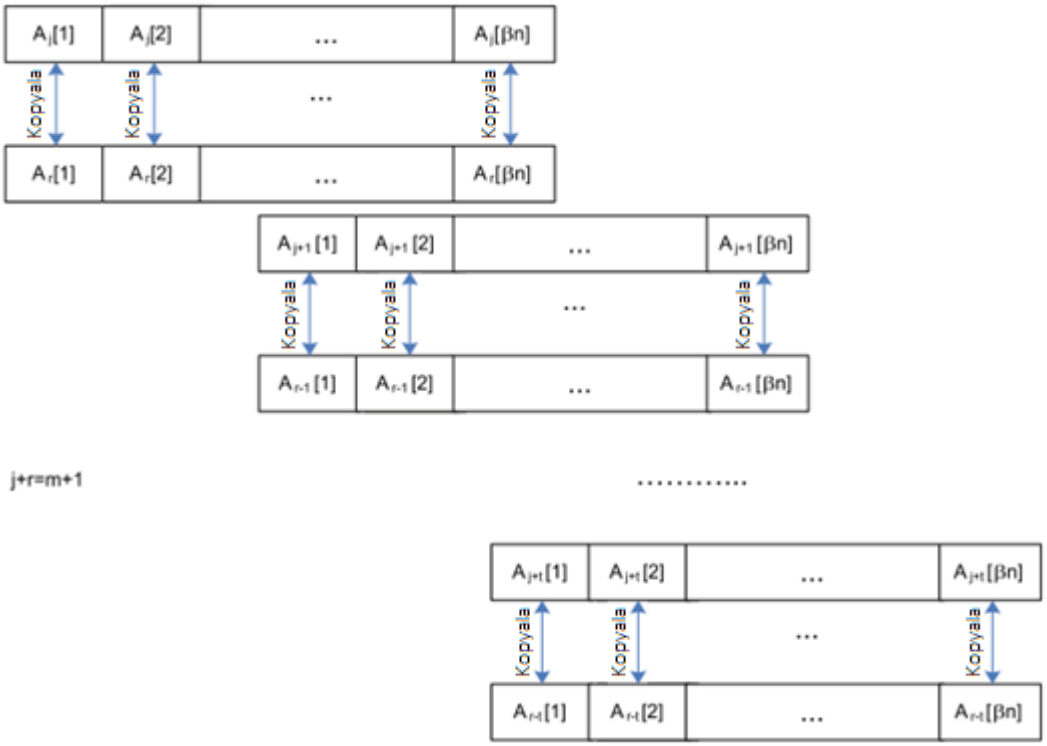
Yöntemde rastgele bir Atom Kümesi oluşturulduktan sonra, her bir atomun hedef fonksiyon değeri ve her elektronun amaç fonksiyonu üzerindeki etkisi hesaplanır. Çözüm üzerinde büyük etkisi olan elektron değerleri atomun ilk endekslerinde, çözüm üzerinde daha az etkiye sahip elektron değerleri atomun son endekslerinde olacak şekilde sıralanır.

5.2. KOVALENT BAĞ

Kovalent Alanda (KA) olan atomun başlangıcına yakın elektron değerleri, kovalent bağ operatörü tarafından etki değerlerinin büyüklüğüne bağlı olarak eşleşen atomlar arasında kopyalanır. Başka bir deyişle, atom setindeki çözüm için pozitif bir etkiye sahip olan parametre değerlerinin sayısını arttırmaya çalışılır. Kovalent bağ operatörünün algoritması [63]:

1. $k \leftarrow 1, 2, \ldots, \beta n$ // $k \leq \beta n$
2. Eğer $E[A_j(k)]$, $E[A_r(k)]$'dan daha iyiyse
3. $E[A_j(k)]$'yı $E[A_r(k)]$'ya kopyala
4. Aksi halde $E[A_r(k)]$, $E[A_j(k)]$'dan daha iyiyse
5. $E[A_r(k)]$'yı $E[A_j(k)]$'ya kopyala

Atom j ile atom r'nin i indeksinde bulunan elektronların elektron etkisi karşılaştırılarak, daha iyi etki değerine sahip olan atomun elektronu diğer atomun elektronuna kopyalanır. Bu operatör, doğadaki kimyasal kovalent bağ oluşumunda iki atomun aynı elektronun paylaşımını taklit eder. Kovalent bağ işleminin gösterimi Şekil 5.5'teki gibidir:



Şekil 5.5. Kovalent bağ operatörünün gösterimi [65].

5.3. İYONİK BAĞ

Kovalent bağ operatöründen sonra, iyonik bağ operatörü, atomun sonunda bulunan İyonik Alandaki değerleri rasgele değiştirerek çözüm üzerindeki olumsuz etkisi olan değerleri kaldırır. Bununla beraber rastgele üretilen elektronlar, algoritmanın işleyiş sırasında yerel optimum değerlerde takılmasının önüne geçer. İyonik bağ operatörünün işleyişi aşağıdaki gibidir [63]:

$k \leftarrow \beta n + 1, \beta n + 2, \dots, n$ // β kovalent orandır, $(1-\beta)n = \alpha n$

$A_r[k] \leftarrow L_k + \eta * (U_k - L_k)$

// L_k k. parametre için alt limittir

// U_k k. parametre için üst limittir

Atom kümesinde, k değeri iyonik alanın başlangıcından sonuna kadar değerler alarak iyonik alandaki tüm elektronlara iyonik bağ operatörü uygulanır. Atomun r indeksindeki k elektronu oluşturulurken kullanılan L_k, k parametresinin alt limitini,

U_k ise k parametresinin üst limitini gösterir ve 0 ile 1 arasında rastgele üretilen η katsayısıyla atomun yeni değeri hesaplanır. Bu operatör bir elektronunu atıp yerine yeni bir elektron alarak, doğadaki kimyasal iyonik bağ oluşumunda bir atomun dışarıdan yeni bir elektronun alımını, verimini taklit eder.

5.4. ELEKTRON ETKİLERİ

Kovalent bağ operatörü uygulanırken her bir elektronun elektron değerlerine göre karşılaştırma yapılır. Elektron etkisi, elektronun amaç fonksiyonu üzerindeki etkisidir ve her kovalent bağ işleminde elektron etkileri de güncellenir. İyonik bağ işleminden sonra da oluşturulan her bir yeni elektronun elektron değerleri de hesaplanır. Elektron etkisi Eşitlik 5.1'deki gibidir:

$$E\left[A_j[i]\right] = f(x_i), \quad \begin{matrix} i = 1, 2, \dots, n \\ j = 1, 2, \dots, m \end{matrix} \tag{5.1}$$

Elektronların amaç fonksiyonu üzerindeki etkileri Eşitlik 5.2'deki gibi hesaplanır:

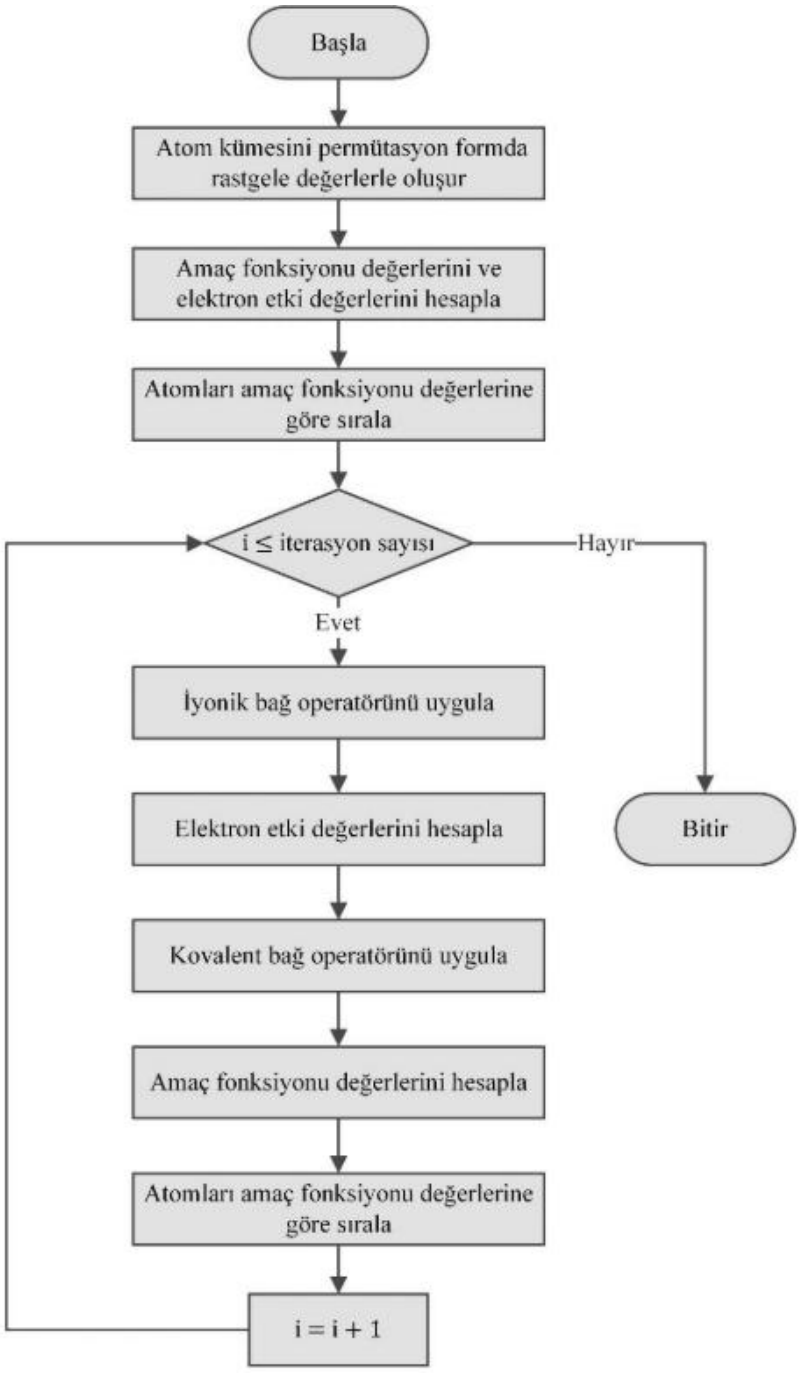
$$f(\vec{x}) = \sum_{i=1}^{n} f(x_i) \tag{5.2}$$

Yeni atom durumunun amaç fonksiyon değeri ve elektron etkileri hesaplanır. Bu süreç, son koşul karşılanana kadar tekrar eder.

5.5. A³'ÜN GEZGİN SATICI PROBLEMİNE UYGULANMASI

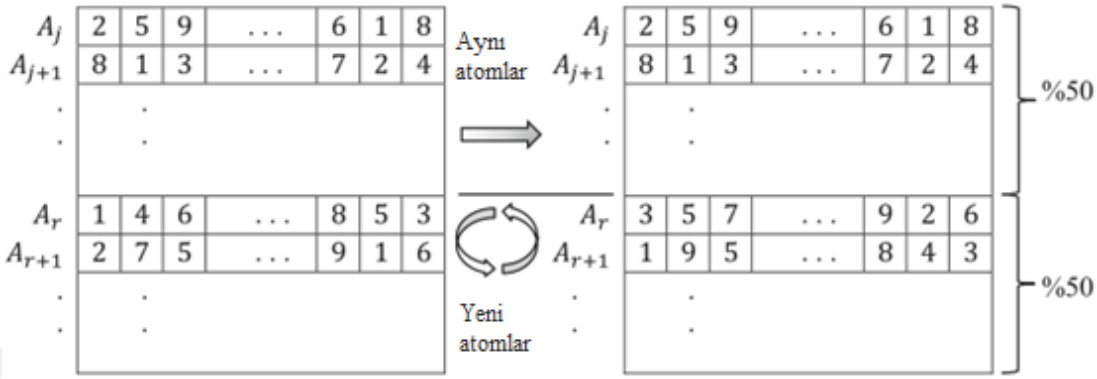
Takviyeli öğrenme [64], grup elevatör kontrolü optimizasyonu [66], veri kümeleme analizi [67] ve optimum günlük beslenme planının hazırlanması [68] gibi farklı problemlere A^3 uygulanmıştır. Yıldırım ve Karcı tarafından küçük ölçekli GSP problemleri A^3 ile tasarlanmıştır [32] ve ayrıca GSP A^3 kullanılarak Türkiye'nin 81 ili üzerine de uygulamıştır [69].

GSP uygulamasını gerçekleştirirken, problemin yapısına göre orijinal A^3 tasarımında yapılan bazı değişiklikler Şekil 5.6'te gösterilmiştir. İlk olarak, daha önce ziyaret edilen bir şehre tekrar ziyaretleri önlemek için, atom setlerinin ilk üretiminde bir permütasyon yöntemi kullanılarak atomlar rastgele üretilir. A^3 operatörleri aynı şekilde uygularken, atom kümesinde tekrarlardan kaçınmak ve atomun permütasyon formunu korumak için bazı teknikler geliştirilmiştir.



Şekil 5.6. GSP için A^3 uygulamasının akış şeması [63].

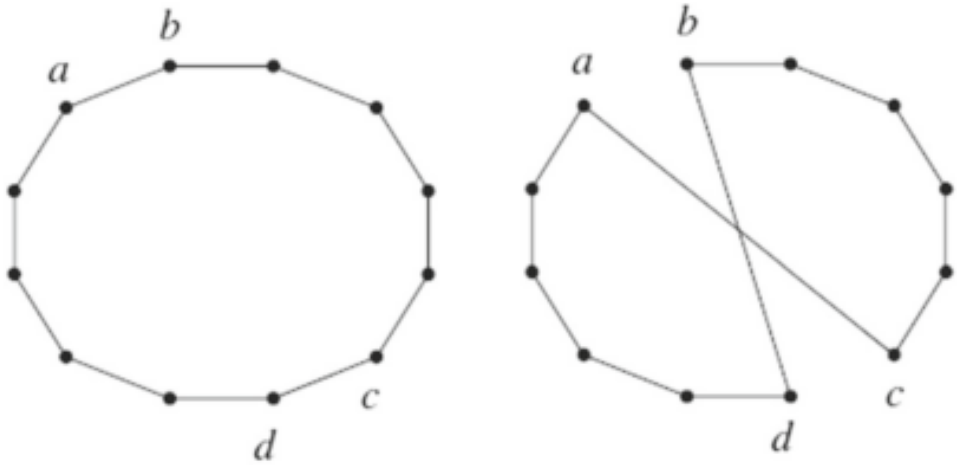
Atomların amaç fonksiyon değerlerini ve bir sonraki şehre olan uzaklığa bağlı elektronların etki değerlerini hesapladıktan sonra, GSP için iyonik bağ operatörü kovalent bağ operatöründen önce uygulanır.



Şekil 5.7. GSP için iyonik bağ işleminin gösterimi [65].

İyonik bağ operatörü uygulandığında, atomların etki değerlerini hesaplamak gerekir çünkü atom kümesine yeni atomlar rastgele üretilir. Bu nedenle kovalent bağ operatöründen önce, elektronların amaç fonksiyonu üzerindeki etki değerleri yeniden hesaplanır.

Daha sonra kovalent bağ operatörü uygulanır. Kovalent bağ operasyonunda, atomlar çiftler halinde eşleştirilir, elektronların amaç fonksiyonu üzerindeki etki değerleri karşılaştırılır. Daha küçük etki değerine sahip elektron, a olarak adlandırılır. Bir sonraki elektrona ise b denir. Daha büyük etki değerine sahip olan elektron c olarak tanımlanır ve bir sonraki endeksi elektron d olarak adlandırılır. Bu tanımlamalar yapıldıktan sonra, daha büyük etki değerlerine sahip elektronlar için 2-opt yöntemine dayanan benzer teknikler geliştirilmiştir.



Şekil 5.8. Eşleştirilmiş bir atomun daha büyük bir değer elektronu için kovalent bağ operasyonu (a) Başlangıç turu (b) Yöntemi uyguladıktan sonraki tur [65].

Kovalent bağ operasyonu uygulandığında 5 farklı duruma göre değerlendirilir. A_j ve A_r n elektrona sahip eşleştirilmiş atom çifti olduğunu varsayalım, $E[A_j(k)]$ A_j atomun k. elektronunun etkisidir ve $E[A_r(k)]$ A_r atomunun k elektronunun etkisidir, öyle ki $E[A_j(k)] < E[A_r(k)]$. Kovalent bağ uygulanma durumları aşağıdaki şekildedir [69]:

$Eğer\ k < l\ ise, A_r[k+1:l] = A_r[l:-1:k+1]$

$Eğer\ l \leq k\ ise;$

$a)\ Eğer\ l = 1\ ve\ k = n-1\ ise, A_r[k+1l] = A_r[lk+1]$

$b)\ Eğer\ l \neq 1\ ve\ k = n-1\ ise, A_r[k+1:\ -1:\ l] = A_r[l:\ k+1]$

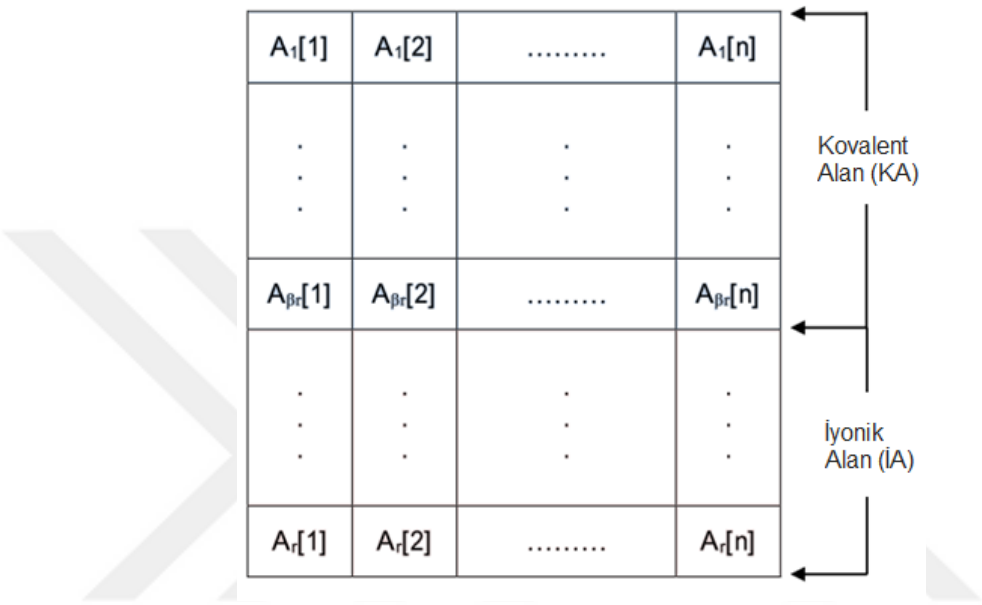
$c)\ Eğer\ l = 1\ ve\ k \neq n-1\ ise, A_r[k:\ -1:l+1] = A_r[l+1:\ k]$

$d)\ Eğer\ l \neq 1\ ve\ k \neq n-1\ ise, A_r[k+1:\ -1:\ l] = A_r[l:\ k+1]$

Kovalent bağ operasyonundan sonra, atom seti için amaç fonksiyon değerleri hesaplanır ve atom seti amaç fonksiyon değerlerine göre sıralanır. İterasyon sayısı kontrol edilir ve iterasyon sayısına ulaşılana kadar A^3 operatörleri uygulanmaya devam eder.

Karşılaştırma problemleri için A^3 uygulandığında, atom grubu, amaç fonksiyon değerlerine göre sıralanır ve her bir atom, elektron etki değerlerine göre sıralanır.

Diğer yandan, GSP için elektronların sırası, şehirlerin ziyaret edilme sırasıdır. Bu nedenle elektronların etki değerlerine göre hizalanması GSP'nin yapısı için uygun değildir çünkü aday çözüm tamamen parçalanır. Bu nedenle, A^3'ün GSP için uyarlanmış formunda, atomların elektronları, elektron etki değerlerine göre sıralanmaz, atomlar atom etkilerine göre küçükten büyüğe doğru sıralanır.



Şekil 5.9. GSP için atom seti, Kovalent Alan (KA) ve İyonik Alan (İA) temsili [69].

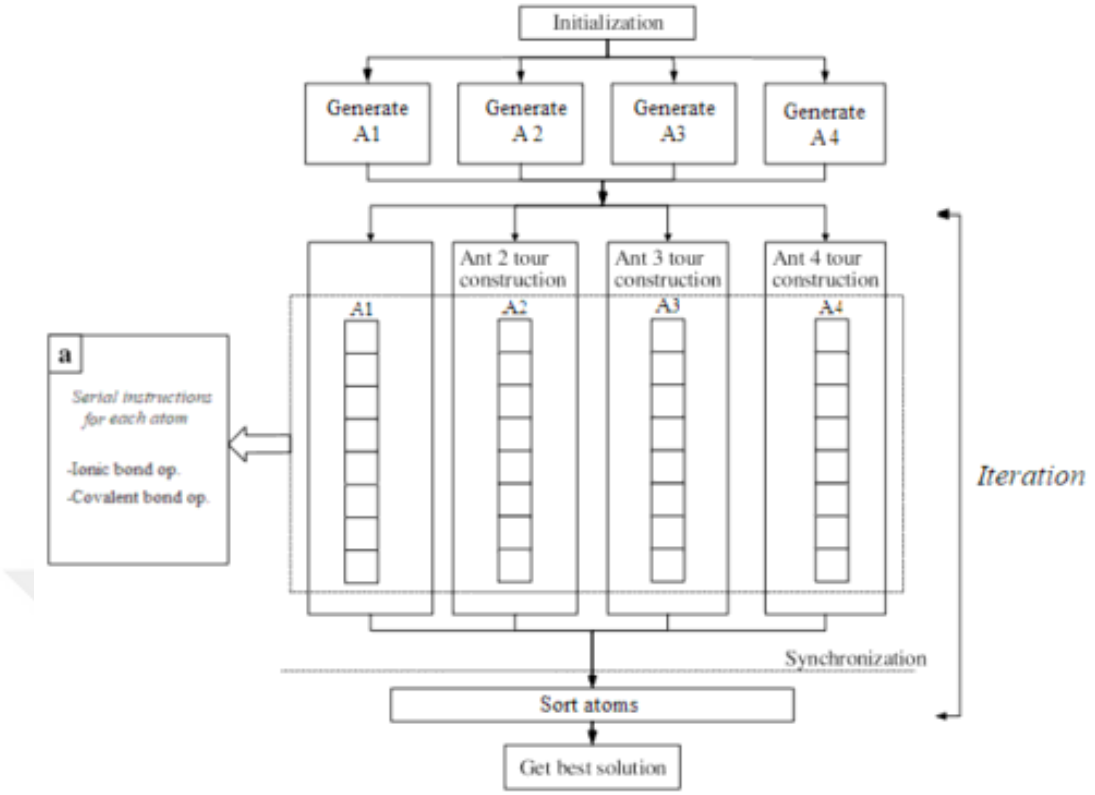
5.6. A^3'ÜN GEZGİN SATICI PROBLEMİNE PARALEL OLARAK UYGULANMASI

Seri algoritmalar paralelleştirilirken öncelikle hangi tür paralel yapı kullanılacağı belirlenmesi gerekmektedir. Bit seviyesinde paralellik, komut seviyesinde paralellik ve görev paralelliği olmak üzere üç farklı tipte paralellik vardır [30]. Bit seviyesinde paralellikte, bilgisayar mimarisinde hızlanma, bilgisayar kelime boyutunun, işlemcide döngü başına işlenebilecek bilgi miktarının iki katına çıkarılmasıyla sağlanmıştır. Sözcük boyutunu artırmak, işlemcinin [30] sözcüğünün uzunluğundan daha büyük değişkenlerde bir işlemi gerçekleştirmek için yürütmesi gereken komutların sayısını azaltır.

Diğer paralelleştirme şekli, programdaki yönergeleri yeniden düzenleyen, bunları gruplar halinde birleştiren ve sonucu değiştirmeden, bu talimatları paralel olarak yürüten komut düzeyinde paralelliktir [30]. Komut seviyesindeki paralellik, boru hattı (pipelining) ve süperskalar hesaplama gibi 2 temel yaklaşıma sahiptir. Boru hattı, birden fazla komutun tek saat çevriminde aynı anda çalışmasını sağlar. Program genellikle aşamalara ayrılır ve her aşama kısmi sonuçları bir sonraki aşamaya ileterek, bir komutun belirli bir bölümünü aynı anda işler Süperskalar bir işlemci tasarımı, tek bir işlemcinin içinde komut seviyesinde paralellik sağlayan, daha fazla işlemin aynı saat çevriminde yapılmasına izin veren bir paralel hesaplama yapısını oluşturur. Bu, CPU'nun, çift fonksiyonlu birimlerde aynı anda birden fazla talimatı (komut gönderme olarak adlandırılır) çalıştırmasıyla, bir saat döngüsü sırasında birden fazla komut yürütmesi anlamına gelir. Bu işlevsel birimler, işlemci çekirdeği içinde aritmetik mantık birimi (ALU), kayan nokta birimi (FPU), bit kaydırıcı veya çarpan gibi bir yürütme kaynağıdır.

Görev paralelliği, birbirinden tamamen farklı iş parçacıklarının birden fazla işlem birimi içeren sistemlerde aynı veya farklı veriler üzerinde çalışmasıdır. Veri paralelliğini, görev paralelliğinden ayıran temel fark da veri paralelliğinde aynı işlerin aynı veya farklı veriler üzerinde çalıştırılmasıdır.

Bu çalışmada GSP için Paralel A^3 (PA^3) uygulamasında veri paralelleştirme yöntemi kullanılmış ve çok çekirdekli CPU üzerinde paralelleştirme yapılmıştır. Atom kümesinde, her bir atom atomu için, etkilerin hesaplanması ve iyonik bağ işlemleri ayrı görevlerdir. Kovalent bağlı operasyon için, her bir çift atom bağımsız olarak işlenir. Her süreç aynı işlevleri yerine getirir ve “Master” Süreci ile iletişim kurar, ancak birbirleriyle iletişim kurmaz. Bu nedenle, farklı veriler üzerinde aynı işleri gerçekleştiren veri paralelliği uygulaması, program için en uygun yöntemdir.



Şekil 5.10. Paralel A^3 (PA^3) yapısının gösterimi.

Paralel Atom Üretme: Atom kümesindeki her atom, farklı veri parçacığını temsil eden belirli şehirlerin bir permütasyonudur. Birbirinden bağımsız olan atom üretme işleminin paralelleştirilmesi yapılabilir.

Paralel İyonik Bağ: Bu işlem atom kümesinin İA'ının yeni atomlarla rasgele değiştirilmesi işlemidir. Atomların ilk oluşturulması ile benzerdir ve paralelleştirme şekli de atom üretimi ile aynıdır.

Kovalent Bağ: Atomun KA'daki atomların ilk yarısı, bölgenin diğer yarısındaki atomlarla ve her bir çift için eşleştirilmiştir; elektronlar, amaç fonksiyonu üzerindeki elektronların etkileri açısından karşılaştırılır. Her bir çift üzerindeki kovalent bağ operasyonu, diğer çiftlerden bağımsızdır ve böylece paralel olarak yapılabilir.

BÖLÜM 6

GENETİK ALGORİTMA

Genetik algoritma, yapay zekâ ve hesaplamada kullanılan sezgisel bir arama yöntemidir. Doğal seleksiyon ve evrimsel biyoloji teorisine dayanarak, arama problemleri için optimize edilmiş çözümler bulmakta kullanılır. Genetik algoritmalar, büyük ve karmaşık veri kümeleri arasında arama yapmak için mükemmeldir. Kısıtlı ve kısıt olmayan optimizasyon problemlerini çözme kabiliyetine sahip olduklarından karmaşık konulara makul çözümler bulabilecekleri düşünülmektedir [70].

Aday bir çözüm popülasyonunu değiştirilerek eldeki probleme en uygun çözümü bulmaya çalışırlar. Popülasyon değerlendirilir ve en iyi çözümler, bir sonraki nesli oluşturmak üzere çoğaltılması ve eşleştirilmesi için seçilir. Birtakım nesiller boyunca, iyi özellikler popülasyona hükmederek, çözümlerin kalitesinin artmasını sağlar. GA'lardaki temel mekanizma, nüfusun kötü özelliklerinin ortadan kaldırılmasıdır. Seçim sürecinden sağ çıkmayan bireylerin kötü özellikleri bir sonraki nesle aktarılmaz. İyi özellikler hayatta kalır ve daha iyi bireyler oluşturmak için rekombinasyon (çiftleşme) ile karıştırılır. GA'larda ayrıca mutasyon mevcuttur, ancak ikincil bir operatör olarak kabul edilir. Onun işlevi, popülasyonda çeşitliliğin kaybolmamasını sağlamaktır, böylece GA yerel optimalara takılmadan çözüm için genel uzayda araştırmaya devam edebilir [6].

6.1. TEMEL KAVRAMLAR VE GENEL ALGORİTMA

Gen: Doğada genetik bilgi taşıyan ve tek başına anlam ifade eden en küçük temel kalıtım birimidir. GA'da ise problemin çözümündeki her bir parametre gen olarak adlandırılmaktadır.

Kromozom: Doğada genetik özelliklerin bir sonraki nesile aktarılması kromozom ile sağlanır. GA'da ise parametrelerin, yani genlerin toplu kümesi kromozomu oluşturmaktadır ve bir sonraki oluşturulacak olan popülasyona özelliklerin aktarılmasını sağlar.

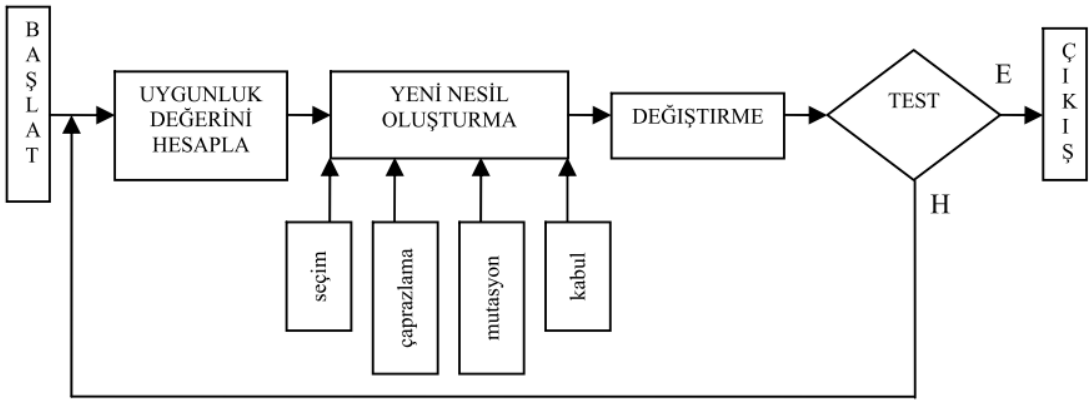
Popülasyon: Kromozom topluluğudur ve problemin için alternatif çözüm kümesini oluşturmaktadır.

GA uygulamasında öncelikle rastgele popülasyon oluşturulur. Daha sonra durdurma kriteri sağlanana kadar bütün bireylerin uygunluk değerleri hesaplanır, seçilim, çaprazlama ve mutasyon işlemleri sırasıyla gerçekleştirilir ve bu işlemlerin sonucunda yeni popülasyon oluşturularak eski popülasyonun yerini alır. GA'nın sözde kodu [6]:

$Population \leftarrow$ initiatePopulation()
Repeat
    calculateFitness($Population$)
    $Offspring \leftarrow$ crossover($Population$)
    $Mutated \leftarrow$ mutate($Population$)
    $Population.$replaceOld($Offspring, Mutated$)
until hasConverged($Population$) || solutionFound($Population$)

Bir bireyin uygunluğu, o birey probleme uygulandığında amaç fonksiyonu tarafından hesaplanır ve çözüm için ne kadar iyi olduğunun bir ölçüsüdür. Uygunluk değeri, çaprazlama ve mutasyon için bireyleri seçmekte kullanılır.

GA'nın akış şemasında algoritma görselleştirilmiştir. Şekil 6.1'de GA'nın akış şeması verilmiştir.



Şekil 6.1. GA'nın genel akış şeması [71].

İyi bir GA kalitesi, bir yerel maksimuma yakınsamadığı sürece, sadece birkaç nesilde oldukça iyi bir tahminin bulunabilmesini sağlar ve nüfusun daha da gelişmesine izin vermek muhtemelen daha da iyi çözümler üretecektir [6].

6.2. BAŞLANGIÇ POPÜLASYONU

Başlangıçta kromozom verilen popülasyon miktarına bağlı olarak rastgele bir şekilde üretilir. Bu durumda $parametre\ sayısı\ x\ popülasyon\ boyutu$ kadarlık bir matris oluşturulur. Matrisin her bir satırı, bir kromozomu temsil eder. Kromozomlar ise, parametreleri temsil eder. Rastgele üretilen popülasyonda bulunan parametreler, amaç fonksiyonundaki yerlerine konulup her bir kromozom için uygunluk değerleri hesaplanarak değerlendirilir.

6.3. DOĞAL SEÇİLİM

Seçim yöntemi, popülasyonun evriminde büyük bir rol oynamaktadır çünkü bu, uygunluğuna bağlı olarak, üzerinde çarprazlanacak veya mutasyona uğrayacak bireyleri seçmek için seçim yönteminin sorumluluğundadır. Seçim, tüm çözüm arama alanını ararken, yüksek uygunluğu olan bireylerin özelliklerinin daha fazla seçilimi olacak şekilde yapılmalıdır. Çaprazlama işleminden önce bir sonraki nesile aktarılacak olan genlerin seçimi farklı yöntemlerle yapılabilmektedir. Bunlardan en çok kullanılan yöntemler sıralama seçim, rulet tekerleği seçim ve turnuva seçimdir.

Rulet tekerleği seçim: Bu seçim için öncelikle kromozomların uygunluk değerleri toplam uygunluk değerlerine bölünerek $0-1$ arasında değişen, bir sonraki nesil için seçilme olasılıkları Eşitlik 6.1'deki gibi hesaplanır. Bu hesaplanan olasılık değerlerine göre seçilim yapılmaktadır [6].

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_i} \tag{6.1}$$

P_i, i geninin seçilme olasılığını, f_i ise uygunluk değerini ifade eder.

Sıralama seçim: Sıralama Seçimi, tek farkla rulet tekerleği seçimi ile benzer şekilde davranır, seçim olasılığı, kişinin uygunluk değeri yerine bireyin sıralamasına dayanır. En düşük uygunluk değeri olan bireyin sıralaması 1, ikinci en düşük değeri olan bireyin sıralaması 2 vs. şeklindedir. En yüksek uygunluğu olan bireylerin sıralaması n'dir, n popülasyondaki birey sayısıdır. Eşitlik 6.2 olasılık hesabını göstermektedir [6].

$$P_i = \frac{r(I_i)}{\sum_{j=1}^{n} j} \tag{6.2}$$

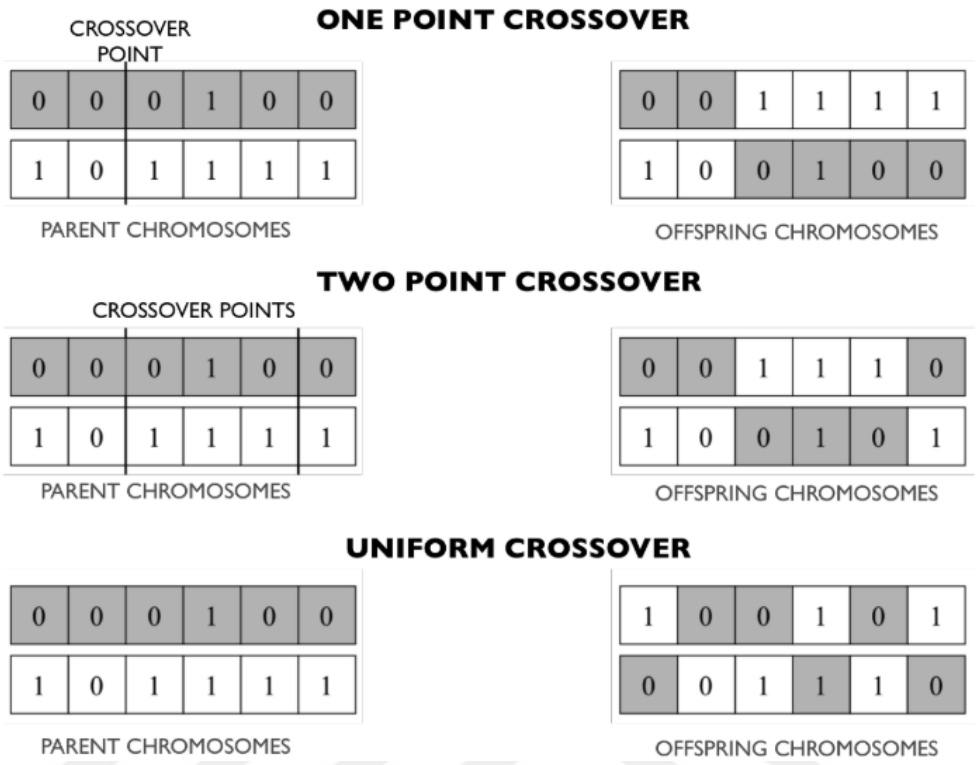
P_i, i geninin seçilme olasılığını, $r(I_i)$ ise sıralamasını ifade eder.

Turnuva Seçimi: Bu yöntemde popülasyonda rastgele bireyler seçilir. Bu bireyleri turnuvaya sokarak aralarından en iyi uygunluk değerine sahip birey seçilir [6]. Çaprazlama işleminde iki turnuva yapılır. Bunlardan her biri çaprazlamaya girecek olan bir ebeveyni seçmek içindir. Turnuva seçiminde kötü olan bireyler turnuvada elenerek bir sonraki nesile aktarılmaz.

6.4. ÇAPRAZLAMA

Çaprazlama işlemi, iki ebeveyn bireyden iki yavru birey üretmekle sorumludur ve biyolojik yeniden üretime benzerdir. Çaprazlama sonucunda meydana gelen nesiller

çaprazlamada kullanılan her iki kromozomun özelliklerini içermektedirler. Şekil 6.2'de bazı çaprazlama yöntemleri gösterilmiştir [6].



Şekil 6.2. Tek nokta çaprazlama, iki nokta çaprazlama ve uniform çaprazlama [72].

6.5. MUTASYON

Mutasyon, bir GA'da büyük bir öneme sahiptir çünkü bu, çözüm üretmek için yeni malzemeler getiren tek işlemdir. Mutasyon olmaksızın, bir popülasyon sadece birkaç kuşakta bir araya gelecek ve muhtemelen yerel bir maksimumda sona erecektir. Mutasyon işleminde verilen olasılığa göre rasgele seçilen genlerin değişimi sağlanır [70].

6.6. GA'NIN GSP'YE UYGULANMASI

GSP'yi çözmek için sayısız yaklaşım vardır. GA, makul bir süre içinde optimal çözümleri bulan bir sezgisel yöntemdir. GA zaman açısından verimli ve GSP için iyi bir yaklaşım olmasına rağmen, bazen yerel optima'ya takılabilir veya şehirlerin sayısı arttıkça zaman alır [7].

GA kombinatoryal optimizasyon problemleri için çok uygundur. Kullanım kolaylığı ve verimli sonuçlarından dolayı, NP-Zor problemlerinin çözümünde çok popülerdirler.

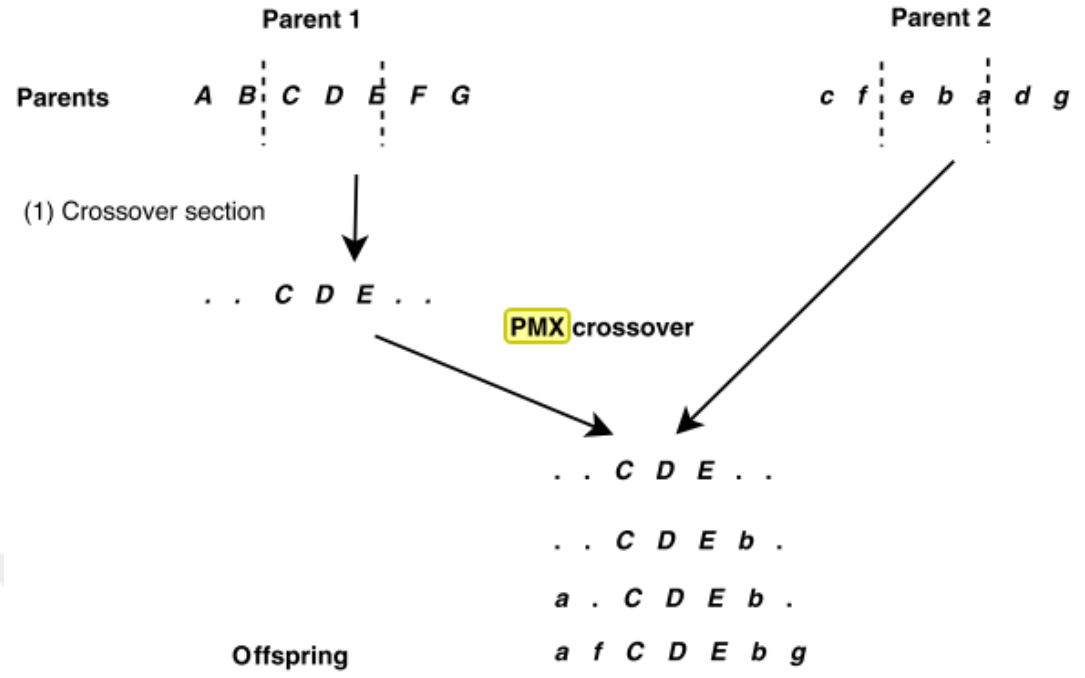
Bir başlangıç popülasyonunun oluşturulması, genetik algoritmanın ilk adımıdır. GSP için, her şehri tam olarak bir kez ziyaret eden ve başladığı şehre geri dönen turların bir listesini oluşturmak anlamına gelir.

Bir tur oluştururken permütasyon kodlaması kullanılmaktadır. Her şehrin, N ile sorun boyutu olan, 0 ile N arasında değişen bir tamsayı tanımlayıcısı vardır. Şehirlerden biri başlangıç şehir olarak kabul edildikten sonra, tüm şehirler ziyaret edilinceye kadar, ziyaret edilmemiş olan şehirler arasında rastgele bir şehir seçilir [7].

Bu çalışmada seçim işleminde, seçim yöntemlerinden rulet tekerleği seçimi kullanılmıştır.

Bir sonraki aşama seçilen iki ebeveyn arasında çaprazlama yapmaktır. GSP için birkaç çapraz yöntem önerilmiştir. Bu yöntemlerden bazıları Kısmen Eşlemeli Çaprazlama (PMX), Çevrim Çaprazlama (CX), Değiştirilmiş Çaprazlama (MC), Sıra Çaprazlama (OX), Sıra Tabanlı Çaprazlama (OBX), Konum Tabanlı Çaprazlama (PBX) [42]. Bu çalışmada, PMX yöntemi kullanılmıştır.

PMX operatörü ilk önce her iki ebeveynde de iki kesme noktası seçer. Bir yavru oluşturmak için, sırayla birinci kromozomdaki iki kesme noktası arasındaki genler, ikinci ana öğede karşılık gelen alt dizenin yerini alır. Daha sonra, çiftleri ortadan kaldırmak ve tüm şehirleri kurtarmak için ters değiştirme, kesme noktalarının dışında uygulanır [42]. Şekil 6.3'te PMX örneği verilmiştir.



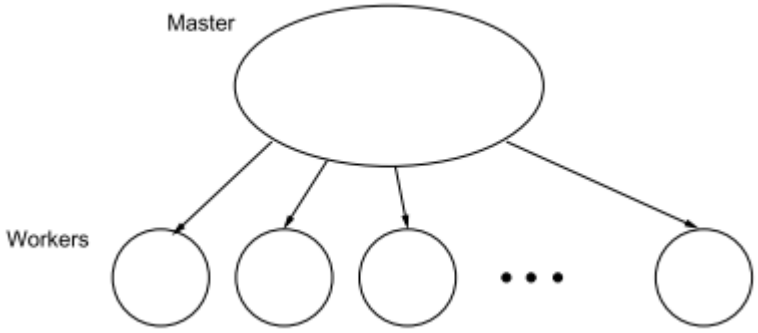
Şekil 6.3. PMX'in uygulanması [31].

6.7. GA'NIN GSP'YE PARALEL UYGULANMASI

Çoğu paralel programın arkasındaki temel fikir, bir görevi parçalara bölmek ve birden fazla işlemci kullanarak aynı anda parçaları çözmek. Bu böl ve ele fethet yaklaşımı birçok farklı yolla GA'lara uygulanabilir ve literatürde başarılı paralel uygulamaların birçok örneği vardır. Bazı paralelleştirme yöntemleri tek bir popülasyonu kullanırken, diğerleri popülasyonu birkaç nispeten izole edilmiş alt popülasyonlara bölmektedir. Bazı yöntemler, büyük ölçüde paralel bilgisayar mimarilerini kullanabilirken, diğerleri daha az ve daha güçlü işlem öğelerine sahip multicomputers için daha uygundur [73].

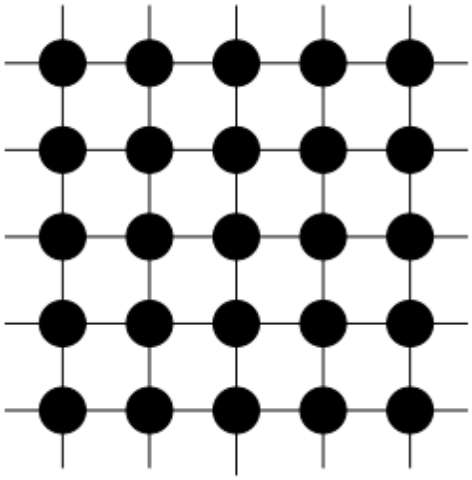
Üç ana Paralel GA (PGA) tipi vardır: Küresel tek popülasyonlu master-slave GA'lar, tek popülasyonlu ince taneli (single-population fine-grained) ve çok popülasyonlu kaba taneli (multiple-population coarse- grained) GA'lar. Bir master-slave GA'da, tek bir popülasyon vardır (tıpkı basit bir GA'da olduğu gibi), ancak uygunluk değerlerinin hesaplanması birkaç işlemci arasında dağıtılır (bkz. Şekil 6.4). Bu tip bir

paralel GA'da, seçim ve çaprazlama tüm nüfusu dikkate aldığından, küresel paralel GA'lar olarak da bilinir.



Şekil 6.4. Bir master-slave paralel GA şeması. Master nüfusu depolar, GA operasyonlarını yürütür ve bireyleri kölelere dağıtır. Köleler sadece bireylerin uygunluğunu değerlendirir [73].

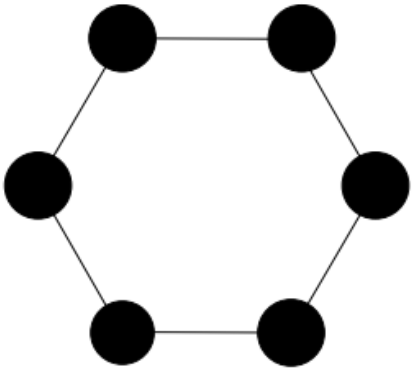
İnce taneli paralel GA'lar, büyük ölçüde paralel bilgisayarlar için uygundur ve mekansal olarak yapılandırılmış bir popülasyondan oluşur. Seçme ve çaprazlama küçük bir alan ile sınırlıdır, ancak komşular tüm bireyler arasında bir miktar etkileşime izin verir (bu GA'ların bir şematiği için Şekil 6.5'e bakınız). İdeal durum, mevcut her işleme elemanı için sadece bir kişiye sahip olmaktır.



Şekil 6.5. Bir ince taneli paralel GA'nın bir şeması. Bu sınıfsal GA'lar, mekansal olarak dağılmış bir popülasyona sahiptir ve büyük ölçüde paralel bilgisayarlarda çok verimli bir şekilde uygulanabilir [73].

Çoklu popülasyon GA'lar daha sofistike olup, bireyleri zaman zaman değiştiren birkaç alt popülasyonda oluşur (Şekil 6.6). Bireylerin bu değişimi, göç olarak

adlandırılır ve daha sonraki çeşitli parametreler tarafından kontrol edilir. Çoklu popülasyon GA'lar çok popülerdir, fakat aynı zamanda anlaşılması en zor olan paralel GA'lar da vardır, çünkü göçün etkileri tam olarak anlaşılamamıştır. Çoklu popülasyon paralel GA'lar GA'nın işleyişinde temel değişiklikleri başlatır ve basit GA'lardan farklı bir davranışa sahiptir.

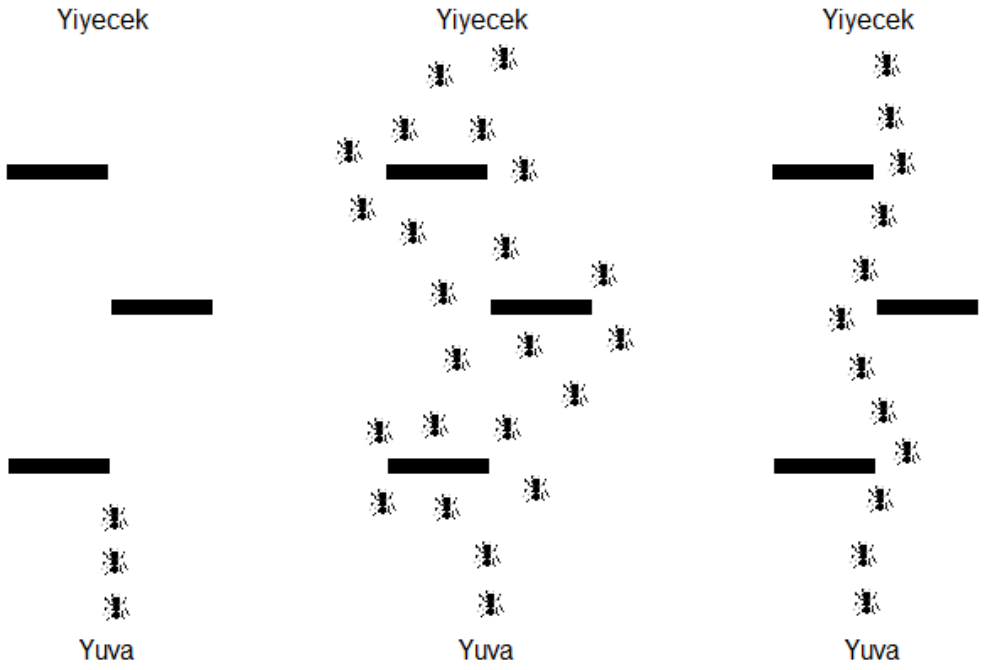


Şekil 6.6. Çoklu popülasyon paralel GA'nın bir şeması. Her süreç basit bir GA'dır ve popülasyonlar arasında (seyrek) bir iletişim vardır [73].

BÖLÜM 7

KARINCA KOLONİSİ ALGORİTMASI

Karınca kolonilerinin doğadaki davranışlarını baz alan ilk algoritma Marco Dorigo tarafından 1996 yılında ortaya atılmıştır [74]. ACO (Ant Colony Algorithm), karıncaların yön duygusundan ve yiyecek kaynaklarını bulmak mantığından ilhamla geliştirilen bir meta-sezgisel yöntemdir. Gerçek karıncaların yuvaları ile yiyecek topladıkları noktalar arasındaki mesafeyi en aza indirgeyen rota, salgıladıkları feromon kimyasının saptanmasına dayanmaktadır [8]. Gerçek karıncalar, yuvalarını yiyecek aramak için terk ettiklerinde, daha önceki karıncalar tarafından salgılanan feromondan izleyecekleri rotayı belirlerler [25]. Feromon, karıncaların bacaklarından salgılanan, belirli bir süre yolda kalan ve zamanla buharlaşan bir kimyasaldır.



Şekil 7.1. Gerçek karıncaların adaptasyon davranışı a) Karıncalar yiyecek aramaya başlar b) Karıncalar, yollarda feromon olmadığı zaman, eşit olasılıkla yiyecek kaynağını arar, c) Bir süre sonra karıncaların çoğu en kısa yolu seçer.

Başlangıçta karıncalar rastgele yiyecek ararlar. Daha kısa sürede geri dönmek için daha kısa yol kullanan karıncalar yolda daha fazla feromon bırakırlar. Gıda maddesi araştırması ilerledikçe, kısa mesafelerdeki feromon miktarı konsantre olacak ve feromonun buharlaşması, zamanın kısalmasına paralel olarak azalacaktır. Aynı şekilde, yollardaki daha az başlangıçtaki feromonlara bağlı olarak karıncaların uzun mesafe yolunun tercih oranı azalır ve bir süre sonra yol üzerindeki feromon tamamen buharlaşır ve karıncalar bunları kullanmaz [8]. Bu süreç, Şekil 2'de gösterilen ACO Algoritması olarak modellenmiştir.

```
procedure ACO
    initialise pheromone trails;
    while (termination condition not satisfied) do
        construct candidate conformations;
        perform local search;
        update pheromone values;
    end
end
```

Şekil 7.2. Kombinasyonel optimizasyon problemleri için Ant Koloni Optimizasyon Algoritmasının temel yapısı.

ACO algoritmaları, optimizasyon problemleri için çok kullanışlıdır, çünkü algoritmanın her adımında sayısız çözümü incelerler.

7.1. PARAMETRELER

Karınca sayısı: Kolonide ne kadar karınca olacağını belirleyen parametredir.
Yineleme Sayısı: Aramanın kaç defa tekrarlanacağını belirleyen parametredir.

Feromon Artış Oranı (α): Düğümler arasında bulunan feromonun artış seviyesini belirleyen parametredir.

Sezgi Arttırma Oranı (β): Düğümler arasında bulunan mesafenin önemini belirleyen parametredir.

Feromon Buharlaşma Hızı (ρ): Her bir tekrar sonunda düğümlerin feromonlarının buharlaşma hızını belirleyen parametredir.

7.2. GEÇİŞ KURALI

ACO algoritmasında karınca turu esnasında bir sonraki gideceği şehri verilen olasılığa göre 2 şekilde seçebilir. İlk alternatif, karıncanın gideceği yolu yollardaki feromon miktarını göz önüne alarak eşitlik 7.1'deki gibi yaptığı hesaplamada çıkan en yüksek değere sahip yolu seçmektir [75]. Genellikle bu şekilde tercih yapma olasaılığı (q_0) %90 olarak belirlenir.

$$j = \max_{u \epsilon J_k(i)} \left\{ [\tau(i,u)]^{\alpha} * [\eta(i,u)]^{\beta} \right\} \qquad \text{eğer } q \leq q_0 \tag{7.1}$$

Burada $\tau(i,u)$, (i,u) yolundaki feromon miktarıdır. $\eta(i,u) = 1/d\ (i,u), i$ noktasından u noktasına olan mesafenin tersidir. $J_k(i)$, i noktasındaki k karıncasının henüz gitmediği şehirleri temsil göstermektedir. $q0\ (0 < q0 < 1)$çözüm uzayını araştırmanın göreli önemliliğini gösteren parametredir.

$q \leq q0$ olduğunda diğer geçiş kuralı uygulanır. Bu kurala göre, gidilecek bir sonraki nokta Eşitlik 7.2'ye göre hesaplanan olasılık değerlerine bağlı olarak rastsal bir şekilde seçilmektedir [75]. Böylelikle feromon miktarı daha yoğun olan yolların seçilme olasılığı daha fazla olacaktır.

$$p_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)]^{\alpha} * [\eta(i,j)]^{\beta}}{\sum_{u \in j_k(i)} [\tau(i,j)]^{\alpha} * [\eta(i,j)]^{\beta}} & \text{eğer } j \in J_k(i) \\ 0 & \text{Diğer durumlarda} \end{cases} \tag{7.2}$$

7.3. FEROMEN GÜNCELLEMESİ

Tüm karıncalar turlarını tamamladıktan sonra yollardaki feromon miktarlarında güncelleme yapılmaktadır [75]. Öncelikle bütün yollardaki feromonlar, belirlenen buharlaşma oranına göre buharlaştırılmaktadır. Daha sonra ise karıncaların geçiş yaptıkları yollardaki feromon miktarları, o yolu kullanan karıncanın toplam yol

uzunluğu ile ters orantılı olacak şekilde arttırılmaktadır. Böylelikle daha kısa yol gitmiş olan karıncaların kullandıkları yollardaki feromon miktarlarındaki artış daha fazla olacaktır. Feromon güncellemesi iki yolla yapılabilmektedir. Bunlar lokal feromon güncellemesi ve global feromon güncellemesidir [75].

7.3.1. Lokal Feromen Güncellemesi

Karıncaların hepsi turlarını tamamladıktan sonra, feromon miktarları verilen oranda buharlaştırılır, her bir karıncanın turu esnasında geçiş yaptığı yollarda belli bir miktar feromon artışı yapılır. $\tau_{ij}(t)$, t iterasyonuna kadar biriken feromon miktarı, $\Delta\tau_{ij}^{k}(t+1)$, t iterasyonundaki feromon düzeyi, ρ $(0 \leq \rho \leq 1)$ feromon buharlaşma parametresi olmak üzere lokal feromon düzeyi Eşitlik 7.4'teki gibi hesaplanır [75].

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t+1) \tag{7.3}$$

$$\Delta\tau_{ij}^{k}(t+1) = \begin{cases} 1/L^{k}(t+1) & \text{k karıncası } (i.j) \\ & \text{yolunu kullanmışsa} \\ 0 & \text{diğer durumlarda} \end{cases} \tag{7.4}$$

$L_k(t+1)$ bir karıncanın toplam gittiği tur uzunluğudur. Lokal feromon güncellemesi, turları her iterasyonda değiştirerek geçiş yapılan yolları daha tercih edilen hale getirir. Karıncalar, her iterasyonda değişen feromon miktarına göre yaptıkları turları da değiştirmektedir. Bu şekilde sürekli olarak daha kısa olan turları bulmaktır.

7.3.2. Global Feromen Güncellemesi

ACO algoritmasında Global Feromon güncellemesi, geçerli olan adımdaki karıncalar arasından en iyi sonuca sahip karıncanın izlemiş olduğu yolun feromon düzeyinin arttırılmasıyla oluşur ve iterasyonlardaki en iyi sonuçların belli bir oran ile sonraki iterasyonalara aktarılmasını sağlar. Global Feromon güncellemesi de Lokal Feromon güncellemesine benzemektedir. Eşitlik 7.6'ya göre yapılır [75].

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}^{k}(t+1) \tag{7.5}$$

$$\Delta\tau_{ij}(t+1) = \begin{cases} 1/L_{best}(t+1) & (i.j) \text{ en iyi tura aitse} \\ 0 & \text{diğer durumlarda} \end{cases} \tag{7.6}$$

$L_{best}(t+1)$ geçerli iterasyondaki en iyi turun uzunluğunu belirtmektedir.

7.4. PARALEL ACO UYGULAMASI

Paralell ACO (PACO) algoritması teknikleri ile algoritma genel çözüm uzayında optimum çözümü aramayı hızlandırır. ACO'nun paralelleştirilmesi için farklı yaklaşımlar vardır. Bu yaklaşımlar 3 farklı başlıkta kategorize edilebilir [76].

1) Karınca kolonilerinin paralelleşmesi [77]
2) Karıncaların Paralelleşmesi [78]
3) Çözüm elemanlarının değerlendirilmesinin paralelleştirilmesi

Bu yazıda GSP'ye paralel karınca yaklaşımı uygulanmıştır. Her bir karınca, çözümü oluşturmak için ayrı bir işlemciye atanır. Girdiyi almak, karıncaları rastgele yerleştirmek, feromonu güncellemek ve karıncalar arasında iletişimi kurmak ana işlemcinin sorumluluğundadır [76]. Bu kullanılan model master-slave modelidir, ana işlemci master, işlerin gerçekleştiği işlemciler ise slave olarak adlandırılır ve slave tarafından paralel olarak işlemler gerçekleştirilir.

Paralel ACO'da öncelikle her karınca rastgele şehirlere atanır. Daha sonra karıncalar paralel olarak, geçiş kurallarına göre yol oluştururlar. Yol oluşturma işleminin yani bir döngünün sonunda her karınca gittiği yollarda, lokal feromon güncellemesi yaptıktan sonra master, global feromon güncellemesini yapar.

BÖLÜM 8

YAPAY ARI KOLONİSİ ALGORİTMASI

Yapay arı kolonisi (ABC), popüler sürü zeka optimizasyon tekniklerinden biridir. Karaboga [61] tarafından 2005 yılında bal arısı kolonilerinin yiyecek arama ve dans davranışlarından ilham almıştır. ABC'de, suni arıların toplayıcı davranışları, çözüm alanını verimli bir şekilde aramaktır ve arıların dans davranışları, diğer arılarla çözüm hakkındaki bilgileri paylaşmak için yapılır. Bu davranışlar, işçi arılar, gözcü arılar ve kâşif arılar olarak adlandırılan üç tür arı tarafından gerçekleştirilir. İşçi arılar nektar kaynaklarından yararlanırlar (uygulanabilir çözüm) ve gözcü arılar ile paylaşmak için besin kaynağı pozisyonu hakkındaki bilgileri kovana taşır. Gözcü arılar, işçi arılar tarafından paylaşılan besin kaynağı pozisyonu hakkındaki bilgileri dikkate alarak yeni gıda kaynaklarını araştırır. Üçüncü tip arı, kâşif arıdır. Kâşif arı sadece çözüm alanını arar ve yeni bir besin kaynağı bulduğunda, işçi arıya dönüşür. Eğer işçi bir arı belli bir süre içinde kendini geliştiremediyse, geri kâşif arıya dönüşür. Gözcü arıya rastgele bir çözüm görevinden sonra, tekrar çalışan bir arıya döner. Kısacası, işçi ve gözcü arılar küresel optimum seviyeye ulaşmaya çalışırken, kaşif arı, popülasyonun optimizasyon problemi için durgunluğunu önler.

8.1. PARAMETRELER VE ALGORİTMANIN TANIMI

Problem: Matematiksel formül ile ifade edilen ve çözümünü aradığımız problemdir.

Uygunluk fonksiyonu: Çözülmek istenen problemin çözüm adaylarının uygunluğunu bulmamıza yardımcı olan formüldür ve probleme göre tasarlanmaktadır.

Popülasyon Boyutu: Toplam arı sayısıdır.

Paremetre Boyutu: Bilinmeyen değişkenlerin uygunluk fonksiyonundaki sayısını gösterir.

Paremetre Aralığı: Problemin boyutuna göre değişiklik gösteren parametrelerin alabileceği maksimum ve minimum değerlerdir. Parametre değerleri algoritmanın daha iyi sonuçlar elde edebilmesi için dikkatli bir şekilde seçilmelidir.

Maksimum İterasyon: Maksimum döngü sayısını belirtir.

Besin Kaynağı Sayısı: Popülasyon boyutunun yarısı kadardır. Toplam çözüm kümesini ifade eder.

Deneme Limiti: Çözüm kümesi üzerinde geliştirme formülü uygularken kullanılan deneme limitidir.

Algoritmanın sözde kodu [61]:

Kaşif arıları ilk besin kaynaklarına gönder
TEKRAR ET
İşçi arıları besin kaynaklarına gönderin ve nektar miktarlarını belirleyin
Tercih ettikleri kaynakların olasılık değerini hesaplayınız.
Gözcü arılar
Gözcü arıları besin kaynaklarına gönder ve nektar miktarlarını belirle
Arılar tarafından tüketilen kaynakların kullanım sürecini durdurun
Yeni yiyecek kaynakları bulmak için kaşifleri rastgele arama alanına gönder
Şimdiye kadar bulunan en iyi besin kaynağını hafızada tut
TEKRARA GİT (durdurma şartı yerine getirilene kadar)

8.2. BESİN KAYNAKLARININ OLUŞTURULMASI

Her çözüm kümesi için başlangıçta parametre boyutu kadar minimum ve maksimum parametrelerine göre çözüm kümesi Eşitlik 8.1'deki gibi rastgele oluşturulur. Uygunluk değerleri verilen uygunluk fonksiyonuna göre hesaplanır.

$$X_{i,j} = X_{min,j} + rand(0,1) * (X_{max,j} - X_{min,j}) \quad (8.1)$$

Farklı çözüm kaynakları sayısı $i = (1, 2, \ldots, N)$ ile, çözümün boyutu $j = (1, 2, \ldots, D)$ ile ifade edilmiştir. $X_{max,j}$ ve $X_{min,j}$, $X_{i,j}$için üst ve alt limitleri ifade eder.

8.3. İŞÇİ ARI AŞAMASI

Her bir çözüm kümesine sırayla Eşitlik 8.2'deki gibi geliştirilme formülü uygulanır. Geliştirmeden sonra eğer çözüm kümesinin uygunluk değeri artmışsa, eski çözüm kümesinin yerine yenisi getirilir. Uygunluk değerinde artış olmadığı zaman çözüm kümesinin bulunduğu indeks değerine ait olan deneme değeri bir attırılır.

$$V_{i,j} = X_{i,j} + \phi_{i,j} * (X_{i,j} - X_{k,j}) \quad (8.2)$$

Verilen $k = (1, 2, \ldots, N)$ ile, $j = (1, 2, \ldots, D)$ değerleri rastgele seçilmiş değerlerdir. $\phi_{i,j}$ değeri $[-1, 1]$ aralığında rastgele seçilmiş bir değerdir.

8.4. GÖZCÜ ARI AŞAMASI

İşçi arıların uygunluk değerine göre çözüm kümeleri için seçim şansı oluşturulur. Rulet Seçim Yöntemi, Turnava Seçim Yöntemi gibi seçim yöntemleri kullanılabilir. Bu çalışmada rulet seçim yöntemi kullanılmıştır. Gözcü arı sayısı kadar çözüm kümesi uygunluk değerlerinin olasılıklarına (Eşitlik 8.3) göre rastgele çözüm seçilir ve geliştirilme formülü işçi arı aşamasındaki gibi uygulanır. Tek farkı geliştirme formülünün tüm besin kaynakları üzerinde değil de, besin kaynaklarının olasılıklarına göre rastgele bir seçilen kaynaklar üzerinde yapılmasıdır. Yani aynı çözüm kümesi bu seçim yöntemine göre birden fazla seçilerek geliştirilebilir. Eğer çözümün uygunluk değeri, eski çözüm kümesine göre artmışsa, yeni çözüm kümesi ile eskisi değiştirilir. Uygunluk değerinde artış olmadığı zaman işçi arı yöntemindeki gibi çözüm kümesinin bulunduğu indeks değerine ait olan deneme değeri bir attırılır.

$$p_i = \frac{fit_i}{\sum_{j=1}^{n} fit_j} \tag{8.3}$$

Eşitlik 8.3'te fit_i i çözümünün uygunluk değeridir ve Eşitlik 8.4'e göre hesaplanır.

$$fit_i = \begin{cases} \frac{1}{1+f_i} & eğer\ f_i \geq 0 \\ 1 + abs(f_i) & eğer\ f_i < 0 \end{cases} \tag{8.4}$$

8.5. KAŞİF ARI AŞAMASI

Deneme değeri kaynağın tükenip tükenmediği gösterir ve eğer bu değer deneme limitinin üstündeyse kaynak tükenmiş demektir. Bu durumda kaşif arılar tükenen kaynaklar yeirne rastgele yeni kaynaklar üretirler.

8.6. ABC'NİN GSP'YE UYGULANMASI

GSP için ABC algoritması farklı şekillerde uygulanmıştır. Sharma ve ark. [79] arıların dinamik bölünmesiyle, Pathak ve ark. [80] geliştirilmiş lokal arama ile, Kıran ve ark. [81] komşuluk operatörleri kullanarak, Kocer ve ark. [82] yerel aramalardan faydalanarak, Choong ve ark. [83] seçim fonksiyonunu değiştirerek, El-Abd ve ark. [84] ABC ve PSO algoritmaların hibrit olarak GSP çözümü için uygulamışlardır.

Bu çalışmada tersine döndürme işlemi dizisi baz alan Khan ve ark. [85] uyguladığı yöntem kullanılmıştır.

İlk etapta çözüm kümesi n parametreye sahip problemin rastgele permütasyonlarını oluşturarak elde edilir ve Eşitlik 8.5'te olduğu gibi uygunluk fonksiyonları hesaplanır.

$$fit_i = 1/(M_c - f(X_i) + 1 \tag{8.5}$$

fit_i i çözümünün uygunluk değeri, $f(X_i)$ i çözümünün toplam tur uzunluğu ve M_c çözüm setindeki maksimum toplam uzunluktur.

Permütasyonların algoritma sırasında bozulmaması için seçilen iki nokta arasında döndürme işlemi uygulanır. Döndürme işlemi bir çözümü diğerine benzetmek için kullanılır ve işçi arı aşamasında çözümleri geliştirmek için uygulanacak olan fonksiyonlar (8.8 –8.15 arasındaki eşitlikler) bu işlemlerle yapılır.

$$Y_i = X_j \Diamond \left(\mathrm{r} \odot (X_i \ominus X_k)\right), \quad i = 1, 2, \dots, f_s, \qquad i \neq k \tag{8.8}$$

$$Y_i = X_i \Diamond \left(\mathrm{r} \odot \left(X_j \ominus X_k\right)\right), \quad i = 1, 2, \dots, f_s, \qquad j \neq k \tag{8.9}$$

$$Y_i = X_{best} \Diamond \left(\mathrm{r} \odot (X_i \ominus X_k)\right), \quad i = 1, 2, \dots, f_s, \qquad i \neq j \neq k \tag{8.10}$$

$$Y_i = X_i \Diamond \left(\mathrm{r} \odot (X_i \ominus X_{best})\right), \quad i = 1, 2, \dots, f_s \tag{8.11}$$

$$Y_i = X_{best} \Diamond \left(\mathrm{r} \odot (X_{best} \ominus X_k)\right), \quad i = 1, 2, \dots, f_s, \qquad i \neq k \tag{8.12}$$

$$Y_i = X_i \Diamond \left(\mathrm{r} \odot (X_{best} \ominus X_{worst})\right), \quad i = 1, 2, \dots, f_s, \qquad i \neq k \tag{8.13}$$

$$Y_i = X_i \Diamond \left(\mathrm{r} \odot (X_{best} \ominus X_k) \oplus \mathrm{r}_1 \odot (X_k \ominus X_i)\right) \tag{8.14}$$

$$Y_i = X_j \Diamond \left(\mathrm{r} \odot (X_{best} \ominus X_i)\right), \quad i = 1, 2, \dots, f_s, \qquad i \neq j \tag{8.15}$$

$A \Theta B$ işlemi B'nin A'ya benzetilmesi için gerekli olan ters döndürme işlemlerinin tamamıdır. $B \Diamond SS$ ise B üzerinde ters döndürme işlem dizilerinin yapımını, $\oplus$ operatörü iki ters çevirme operasyonu dizisinin arka arkaya uygulanmasını ve $\odot$ ise verilen r olasılığına göre işlemin gerçekleşmesini ifade eder. X_i i çözümünü, Y_i yeni elde edilen çözümü, X_{best} en iyi çözümü, i, j ve k işlem yapılan çözümün indeksini göstermektedir.

İşçi arılar geliştirme fonksiyonlarını geliştirme fonksiyonunun başarılı geliştirmeden sonra 1 arttırılan deneme değerine göre elde edilen olasılıkla seçerler. Eğer işçi arı çözümü geliştiremezse deneme sayacını bir arttırır, eğer geliştirirse yeni çözümü eski çözümün yerini alır.

Gözcü arılar asıl algoritmadaki gibi işçi arıların uygunluk değerine göre rastgele seçtiği çözümlerde yeniden işçi arıların yaptığı gibi iyileştirme işlemi yaparlar.

Son olarak kâşif arılar asıl algoritmadan farklı olarak tükenmiş kaynaklar üzerinde 3-opt işlemiyle geliştirme yaparlar. 3-opt yönteminde çözüm 3 parçaya ayrılır ve bu parçaların kombinasyonları tek tek hesaplanır. Eğer 3-opt uygulanan çözümlerde bir gelişme yoksa, eski çözümlerin yerine yeni çözümler rastgele olarak üretilir.

8.7. ABC'NİN GSP'YE PARALEL UYGULANMASI

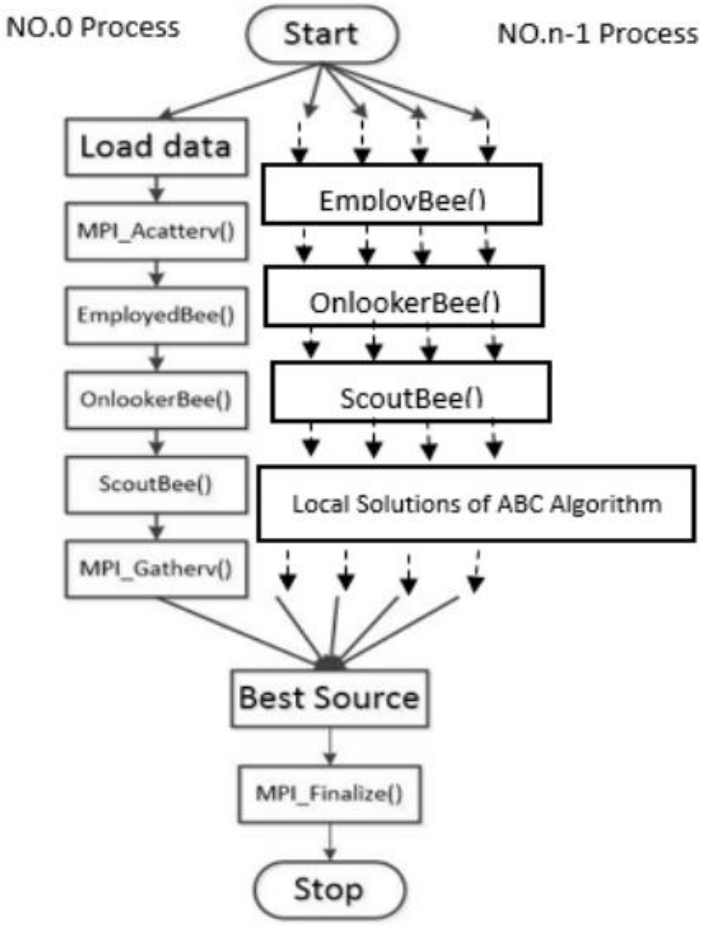
ABC algoritmasında çok fazla iterasyon bulunmaktadır ve bu iterasyonlarda arılar bal toplarken birbirinden bağımsız olarak işlem yapmaktadır. Buna göre Li [86] ve ark. uyguladığı şekilde arıların yaptığı işler paralel olarak gerçekleştirilebilir. Aşağıda ABC algoritmasının sözde kodu verilmiştir:

Kaynakların ilk değer ataması
Maksimum tekrar sayısı kadar
    İşçi Arı Aşaması
    Gözcü Arı Aşaması
    Kaşif Arı Aşaması
Tüm Bilgilerin Toplanması
Kaynakların Bilgilerinin Güncellenmesi
Bitiş
Tüm çözümler arasından en iyi çözümün seçilmesi

Paralel ABC (PABC) algoritmasında işlemler şu şekilde gerçekleşir:

1. No.0 süreci ana süreçtir. Yol bilgilerinin başlatılmasından ve nektar kaynağının bilgilerinin diğer işlemlere dağıtılmasından sorumludur. Sonunda sonuçları toplar.
2. Her yolun araştırılmasından ve güncellenmesinden diğer süreçler sorumludur. Her işlemde paralel olarak birden fazla yol güncellemesi ve sonuçların toplanması anlamına gelir.

Şekil 8.1'de paralel ABC'nin akış şeması verilmiştir.



Şekil 8.1. ABC'nin paralel uygulanması [86].

BÖLÜM 9

UYGULAMA

Paralel uygulamalar için donanım özellikleri, algoritmanın hızı için önemlidir. Bu projede CPU paralelleştirme yöntemi kullanıldığından işlemci performansı uygulamanın hızını etkiler. Bu projede kullanılan işlemci, Çizelge 1'de belirtilmiş olan Intel Core i7-6700K'dır.

Çizelge 9.1. İşlemci özellikleri.

Ürün	Core i7-6700K
Marka	Intel
# CPU Çekirdekleri	4
Frekans (GHz)	4 GHz
Cache (MB)	8
Thread sayısı/Çekirdek	8
Turbo Frekan (GHz)	4.2 GHz
Instructions Width (bits)	64
# Hafıza Kanalları	2
Hafıza Bant Genişliği (GB/Sn)	34.1GB/s

GSP ayrık bir problemdir ve NP-Zor bir problem olduğu için çözümü zordur. Bu çalışmada GSP problemi meta-sezgisel optimizasyon algortimaları olan A^3, ABC, ACO ve GA ile seri ve paralel olarak uygulanıp farklı parametre değerleri için karşılaştırılması yapılmıştır.

A^3, ABC, ACO ve GA algoritmaları farklı boyuttaki GSP değerleri için seri ve paralel olarak uygulanmıştır. Seri olarak ve paralel olarak uygulanan algoritmalarda, boyutları 20, 50, 100, 250 ve 500 olan GSP kullanılmıştır. Kullanılan gezgin satıcı problemleri her boyut için rastgele olarak üretilmiştir ve her yöntem için aynı şehirler kullanılmıştır.

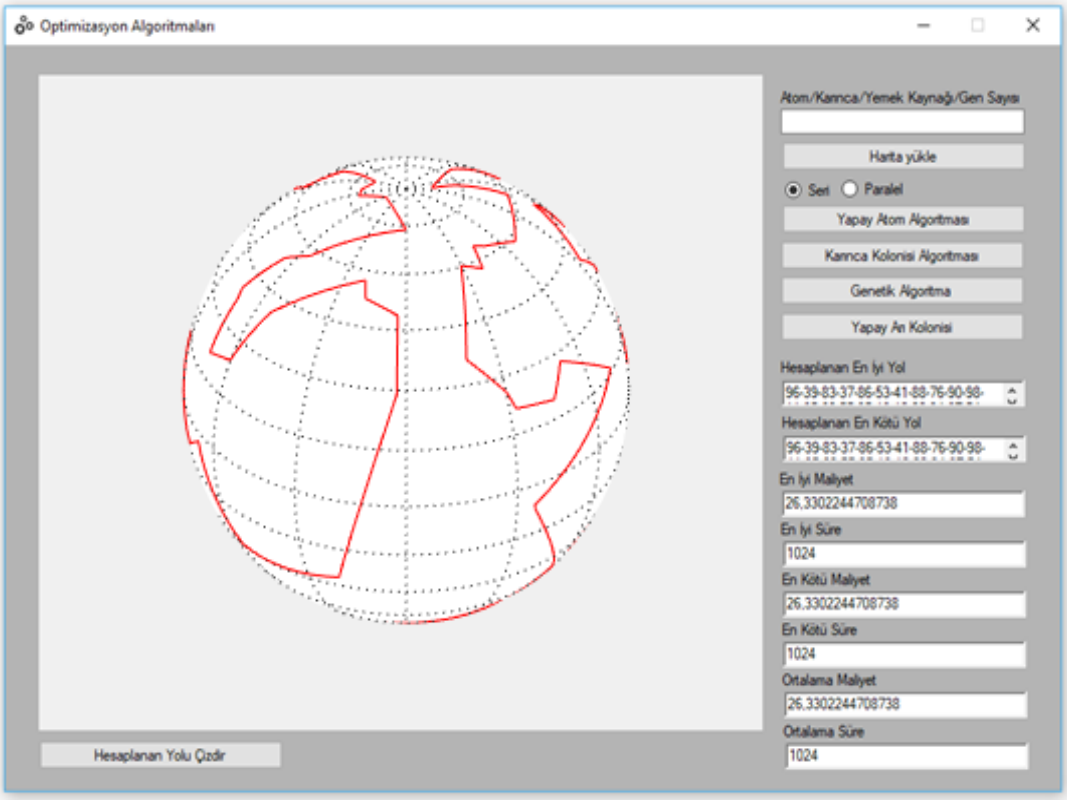
Çizelge 9.2. Değişik boyutlardaki GSP için uygulanan parametreler.

Yöntem	Atom/Karınca/Yemek/Gen	Şehir Sayısı	Fonk. Değ. Sayısı
A^3 ABC ACO GA	200	20	2000000
		50	5000000
		100	10000000
		250	25000000
		500	50000000

Çizelge 9.2, denenmiş problemleri, bu problemler için performans değerlerini nasıl etkilediğini ve sonuçları nasıl karşılaştırdıklarını görmek için algoritma parametrelerinin hangi değerlerinin kullanıldığını gösterir.

Tüm yöntemlerde adil bir karşılaştırılma yapılabilmesi için her yöntemin fonkiyon değerlendirme sayısı GSP şehir sayısına orantılı olarak ayarlanmıştırtır. Her fonksiyon şehir sayısının 10^5 katı fonkiyon değerlendirme sayısına ulaşana kadar döngüye sokulmuştur. A^3, ABC [87] ve GA'da [88] fonksiyon değerlendirme sayısı uygunluk değerinin hesaplanma sayısı olarak alınırken, ACO'da [89] karıncanın bir sonraki gideceği şehri hesaplama sayısı olarak alınmıştır .

GSP çözümünde kullanılan yöntemler üzerinde testlerin yapılabilmesi için C# programlama dili kullanılarak bir arayüz tasarlanmıştır. C#, Microsoft tarafından geliştirilmiş, çok esnek ve güçlü bir programlama dilidir. C#, C ++ ve Java programlama dilleri tarafından sağlanan özelliklere sahiptir [90]. Java ve C ++ 'nın kökenleri C adlı bir dile dayanır. Şekil 9.1'de uygulanan programın ekran görüntüsü bulunmaktadır.



Şekil 9.1. Uygulanan programın ekran görüntüsü. 100 şehirli GSP için çalıştırılmıştır.

Tasarlanan arayüzde atom/karınca/kromozom/yemek kaynağı sayısı belirlenebilir, eğer belirlenmezse varsayılan değerler kullanılır. Bu varsayılan değerler Çizelge 9.2'deki gibidir. Kullanılacak olan harita, harita yükle butonu ile önceden rastgele üretilmiş olan haritalardan seçerek yüklenir. Daha sonra seri yada paralel olarak uygulanabilecek olan yöntemlerden birisi seçilir ve sonuçlar hesaplanır.

Test verilerinin sonuçları, 20 defa çalıştırılan yöntemlerin sonuçlarından elde edilir. Hesaplanan en iyi yol, hesaplanan en kötü yol, en iyi maliyet, en kötü maliyet, ortalama maliyet, en iyi çalışma süresi, en kötü çalışma süresi ve ortalama çalışma süresi testler sonunda elde edilen verilerdir.

Hesaplanan yolu çizdir butonuna tıklandığı zaman hesaplanan en iyi yol küre üzerine çizdirilir ve ekranda gösterilir.

Kullanılan arayüzde çizim için MATLAB fonksiyonu kullanılmıştır. Kullanılan fonksiyonda enlem ve boylam değerleri $-180{,}180$ ve $-90{,}90$ arasındadır, bu yüzden 0 ile 1 arasında olan u, v değerleri öncelikle MATLAB'e uygun olarak hesaplanmış ve daha sonra $plotm(lat, lon, linetype)$ fonksiyonu kullanılarak hesaplanan enlem boylam değerleri ile noktalar arasına küre üzerinde yollar çizdirilmiştir.

İki nokta arasındaki mesafenin doğruluğunu MATLAB fonksiyonu kullanarak kontrol etmek mümkündür. Bunun için yolları çizdirmedeki gibi 0 ile 1 arasında olan u, v enlem ve boylam değerleri öncelikle MATLAB fonksiyonuna uygun olarak $-180{,}180$ ve $-90{,}90$ arasında olacak şekilde düzenlenmiş ve daha sonra $distance(lat1, lon1, lat2, lon2, ellipsoid)$ MATLAB fonksiyonu ile birim küre üzerindeki iki nokta arasındaki mesafe hesaplanmıştır. Elde edilen değer Bölüm 2.5'teki şekilde hesapladığımız değer ile karşılaştırıldığında sonucun doğruluğu görülmüştür. Örnek MATLAB kodu aşağıdaki şekildedir:

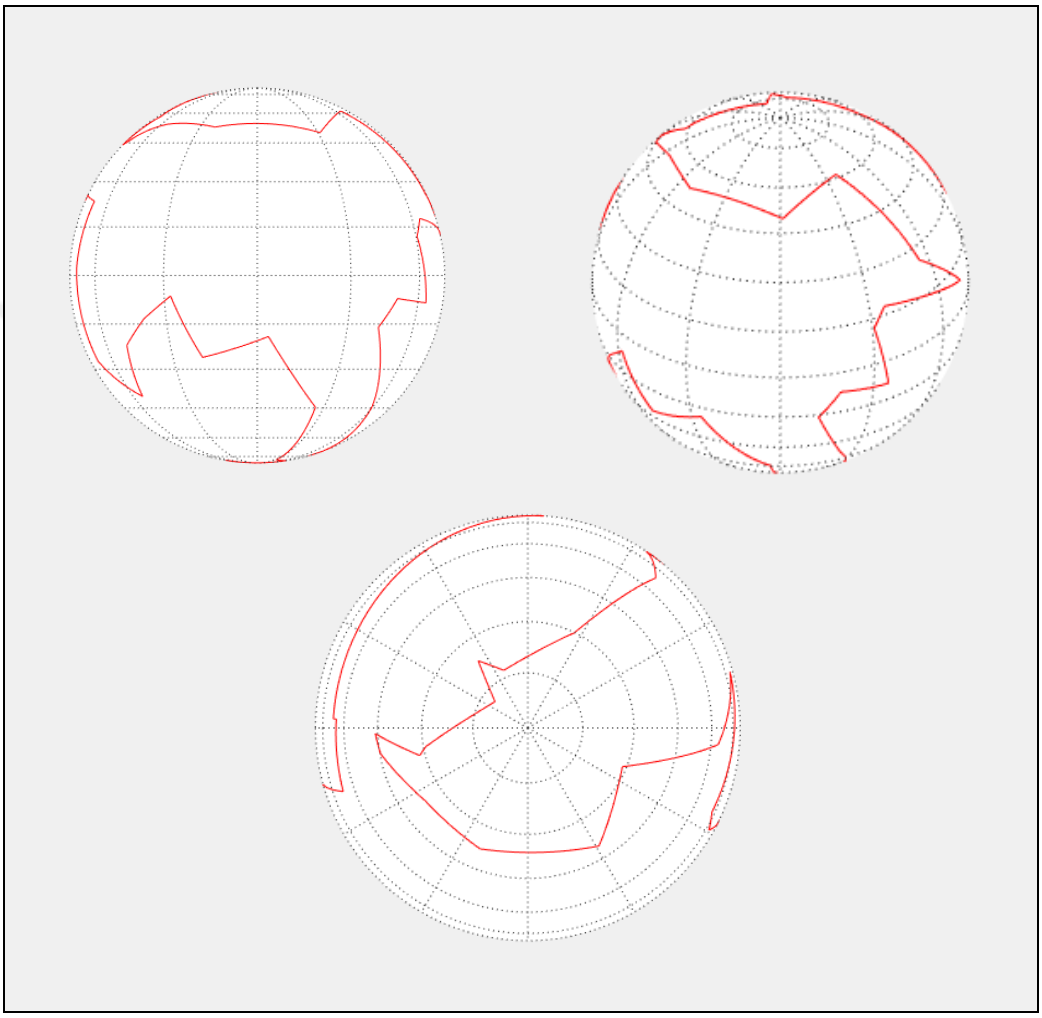
```
>> u1=0.1;
>> v1=0.1;
>> u2=0.3;
>> v2=0.5;
>>  acos(sin(pi*v1)*cos(2*pi*u1)*sin(pi*v2)*cos(2*pi*u2)+
    sin(pi*v1)*sin(2*pi*u1)*sin(pi*v2)*sin(2*pi*u2)+cos(p
    i*v1)*cos(pi*v2))

ans =
    1.4752


>> distance(-144,-72,-72,0,referenceSphere)

ans =
    1.4752
```

Kullanlan arayüzde küre farklı açılar için çizdirilebilmektedir. Şekil 9.2'de farklı açılar için çizdirilmiş olan 20 şehirli GSP örneği yer almaktadır.



Şekil 9.2. Küre üzerine çizdirilmiş olan 50 şehirli GSP'nin farklı açılardan görüntüsü.

Birim küre üzerinde rastgele elde edilmiş olan haritaların testlerinin sonuçlarında birimi cm olarak düşünürsek maliyet cm olarak, süre ise saniye olarak hesaplanmıştır.

Çizelge 9.3'te 20 şehir içeren GSP tüm seri yöntemler için test edilmiştir. En iyi yol A^3, $PA^{3,}$ ABC, PABC, ACO ve PACO yöntemleri ile elde edilirken, PABC en kısa sürede sonuca ulaşan yöntem olmuştur.

Çizelge 9.3. 20 şehirli GSP için seri olarak uygulanan yöntemlerde elde edilen sonuçlar.

Yöntem	En İyi Süre	En Kötü Süre	Ortalama Süre	En İyi Maliyet	En Kötü Maliyet	Ortalama Maliyet	STD (Maliyet)
A^3	1,454	1,475	1,462	10,800	10,800	10,801	0
ABC	0,531	0,728	0,569	10,800	10,800	10,801	0
GA	3,138	3,202	3,179	11,477	14,028	12,973	0,657
ACO	2,105	2,173	2,133	10,800	10,800	10,801	0
PA^3	0,796	0,809	0,802	10,800	11,179	11,008	0,095
PABC	0,264	0,323	0,284	10,800	10,800	10,801	0,000
PGA	4,125	16,968	8,414	12,613	14,869	13,523	0,554
PACO	2,296	2,517	2,403	10,800	10,800	10,801	0,000

Çizelge 9.4'te 50 şehir içeren GSP tüm seri yöntemler için test edilmiştir. En iyi yol ABC yöntemi ile elde edilirken, PABC en kısa sürede sonuca ulaşan yöntem olmuştur.

Çizelge 9.4. 50 şehirli GSP için seri olarak uygulanan yöntemlerde elde edilen sonuçlar.

Yöntem	En İyi Süre	En Kötü Süre	Ortalama Süre	En İyi Maliyet	En Kötü Maliyet	Ortalama Maliyet	STD (Maliyet)
A^3	7,378	12,441	7,914	16,927	17,527	17,259	0,201
ABC	2,769	4,108	3,154	16,519	17,035	16,819	0,185
GA	13,288	16,691	13,737	29,808	33,994	31,842	0,596
ACO	17,694	30,248	24,618	17,269	25,497	21,517	2,205
PA^3	2,750	2,821	2,785	17,321	18,834	18,110	0,433
PABC	1,272	1,293	1,280	16,836	18,207	16,792	0,150
PGA	13,404	13,478	13,447	30,644	32,286	31,461	0,512
PACO	8,315	14,680	12,947	17,850	19,625	18,652	0,593

Çizelge 9.5'te 100 şehir içeren GSP tüm seri ve paralel yöntemler için test edilmiştir. En iyi yol PABC yöntemi ile elde edilirken, PABC en kısa sürede sonuca ulaşan yöntem olmuştur.

Çizelge 9.5. 100 şehirli GSP için seri ve paralel olarak uygulanan yöntemlerde elde edilen sonuçlar.

Yöntem	En İyi Süre	En Kötü Süre	Ortalama Süre	En İyi Maliyet	En Kötü Maliyet	Ortalama Maliyet	STD(Maliyet)
A^3	26,943	27,023	26,996	26,997	27,681	27,346	0,182
ABC	11,338	17,674	13,320	25,344	26,388	25,912	0,308
GA	46,843	50,253	48,820	59,925	68,285	64,039	2,614
ACO	108,364	152,941	120,174	219,254	234,415	228,037	4,910
PA^3	8,753	9,157	9,001	25,729	28,711	26,907	0,828
PABC	5,233	5,263	5,248	25,434	27,021	25,775	0,288
PGA	34,414	34,498	34,456	60,440	68,969	64,004	2,850
PACO	52,546	53,358	52,905	225,680	231,848	228,927	1,580

Çizelge 9.6'da 250 şehir içeren GSP tüm seri ve paralel yöntemler için test edilmiştir. En iyi yol ABC yöntemi ile elde edilirken, PGA en kısa sürede sonuca ulaşan yöntem olmuştur.

Çizelge 9.6. 250 şehirli GSP için seri ve paralel olarak uygulanan yöntemlerde elde edilen sonuçlar.

Yöntem	En İyi Süre	En Kötü Süre	Ortalama Süre	En İyi Maliyet	En Kötü Maliyet	Ortalama Maliyet	STD(Maliyet)
A^3	166,849	167,099	166,984	50,218	50,650	50,426	0,123
ABC	90,281	176,248	136,539	39,246	40,045	39,729	0,162
GA	318,563	344,227	334,707	147,477	161,031	152,455	3,659
ACO	1246,312	1438,432	12974,844	611,278	632,103	622,828	5,479
PA^3	49,840	50,638	50,204	50,030	54,230	52,525	1,234
PABC	45,107	45,801	45,531	40,008	40,561	40,236	0,209
PGA	139,651	142,809	141,322	146,854	155,325	150,484	2,237
PACO	637,089	659,269	642,267	613,504	617,492	615,076	1,225

Çizelge 9.7'de 500 şehir içeren GSP tüm seri ve paralel yöntemler için test edilmiştir. En iyi yol PABC yöntemi ile elde edilirken, PA^3 en kısa sürede sonuca ulaşan yöntem olmuştur.

Çizelge 9.7. 500 şehirli GSP için seri ve paralel olarak uygulanan yöntemlerde elde edilen sonuçlar.

Yöntem	En İyi Süre	En Kötü Süre	Ortalama Süre	En İyi Maliyet	En Kötü Maliyet	Ortalama Maliyet	STD (Maliyet)
A^3	732,216	738,188	734,081	91,721	92,561	92,082	0,133
ABC	545,671	970,255	695,859	59,032	60,825	59,689	0,804
GA	1355,440	1756,898	1598,934	295,779	309,880	302,730	2,917
ACO	9034,580	926,701	9115,186	1255,555	1285,819	1267,195	5,382
PA^3	212,059	226,267	220,925	90,320	92,372	91,351	0,471
PABC	274,578	286,070	280,709	58,337	61,088	59,508	0,743
PGA	520,674	525,501	522,327	291,577	310,361	302,843	4,078
PACO	4506,492	4610,954	4536,934	1267,769	1269,417	1268,892	0,413

Literatürde elde edilen sonuçlarla karşılaştırıldığında, birim küre üzerinde elde edilen rastgele haritalar şehir sayısı yakın olarak karşılaştırıldığında aşağıdaki tablo elde edilmektedir.

Çizelge 9.8. Literatürde 3D GSP için uygulanan yöntemlerden elde edilen en iyi sonuçlar

GSP Boyutu	100	150	200	250	300	350	400
Maliyet [13][16]	37,194	70,574	115,116	165,567	226,118	291,179	354,375
Maliyet [18]	22,294	27,989	34,173	38,744	42,32	43,946	49,538
Maliyet [19]	25,334	30,974	36,379	40,756	45,201	45,092	51,673

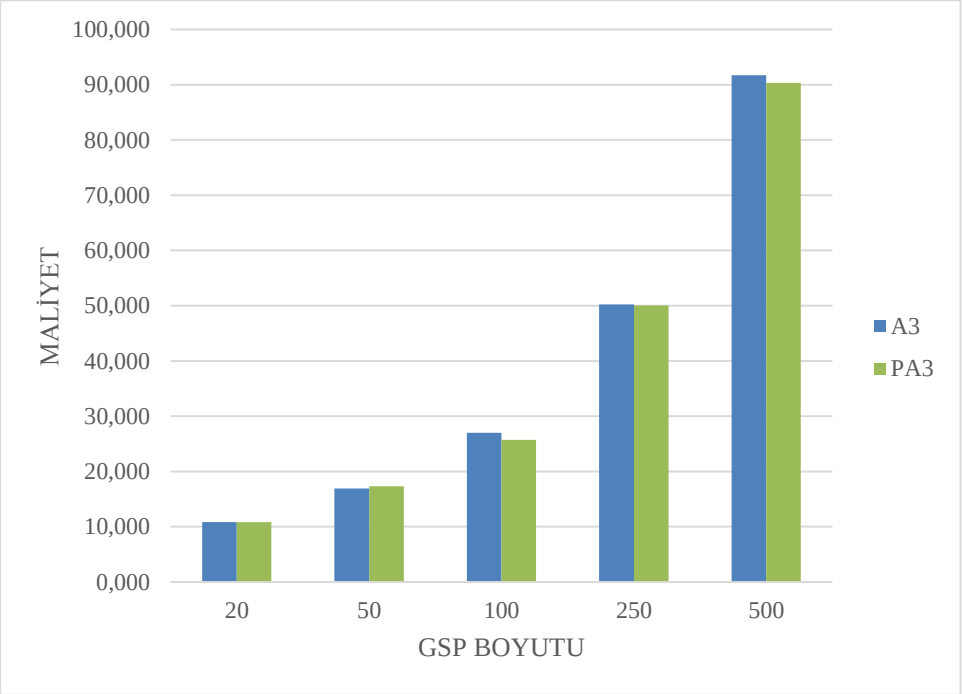
Çizelge 9.9. Literatürde 3D GSP için uygulanan yöntemlerden elde edilen en iyi sonuçlar

GSP Boyutu	100	144	196	256	324	400
Maliyet [15]	22,503	26,627	34,09	42,022	50,84	60,929

Çizelge 2.8 ve 2. 9'da görüldüğü gibi literatürdeki bazı çalışmalarda [13,16,88,89,15] test verileri birim küre üzerinde olsa da haritalar farklı olduğu için ve şehir sayıları tam olarak aynı olmadığı için sadece yaklaşık olarak karşılaştırma yapılabilmektedir. Genel olarak literatürde bulunan çalışmalarla yaklaşık olarak aynı veya daha iyi sonuçlar elde edilmiştir.

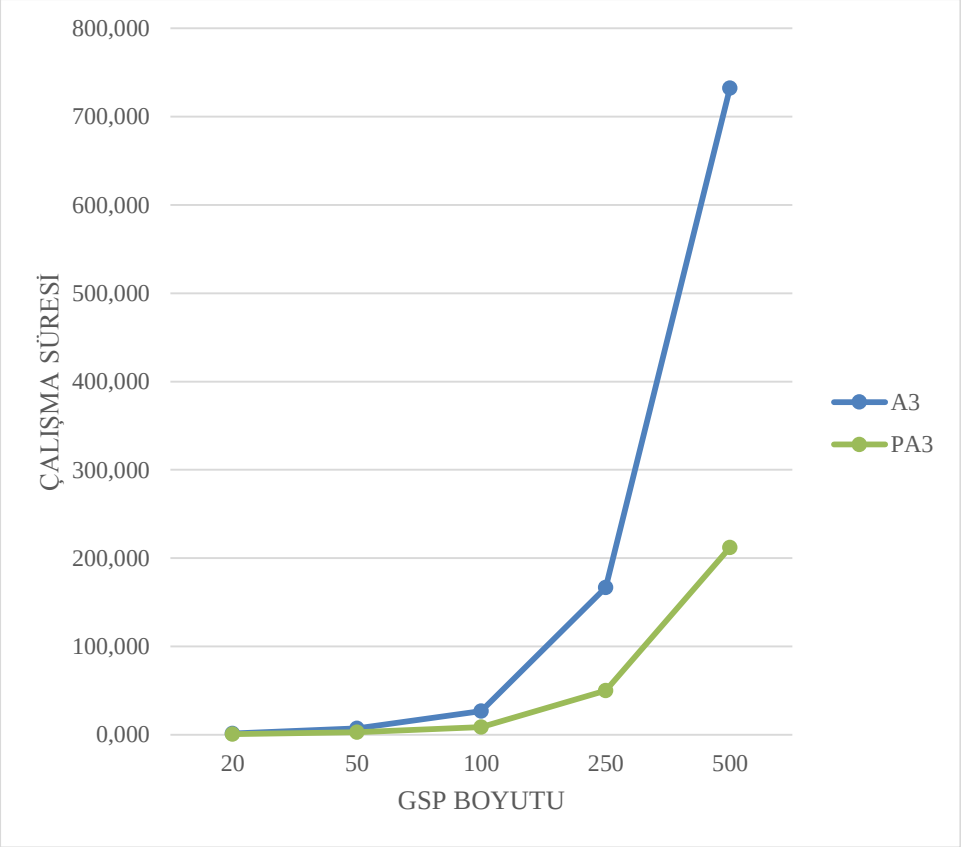
Birim küre üzerinde olmayan diğer çalışmalarla da karşılaştırıldığında [17,26] da elde edilen yollara bakıldığında özellikle şehir sayısı arttıkça bulunan yollar bu çalışmada daha açık ve daha az karmaşıktır.

Şekil 9.3'te A^3 yönteminin PA^3 yöntemi ile maliyet yönünden karşılaştırmasını içeren grafik gösterilmektedir. İki yöntemin ortak olarak çalıştırdıkları GSP'lerin boyutları 20, 50, 100, 250 ve 500 şehirlidir. Bu GSP'ler için çalıştırılan iki yöntemin sonuçları arasında çok bir fark bulunmamaktadır. Yöntemler hesaplama açısından yaklaşık olarak aynıdır.



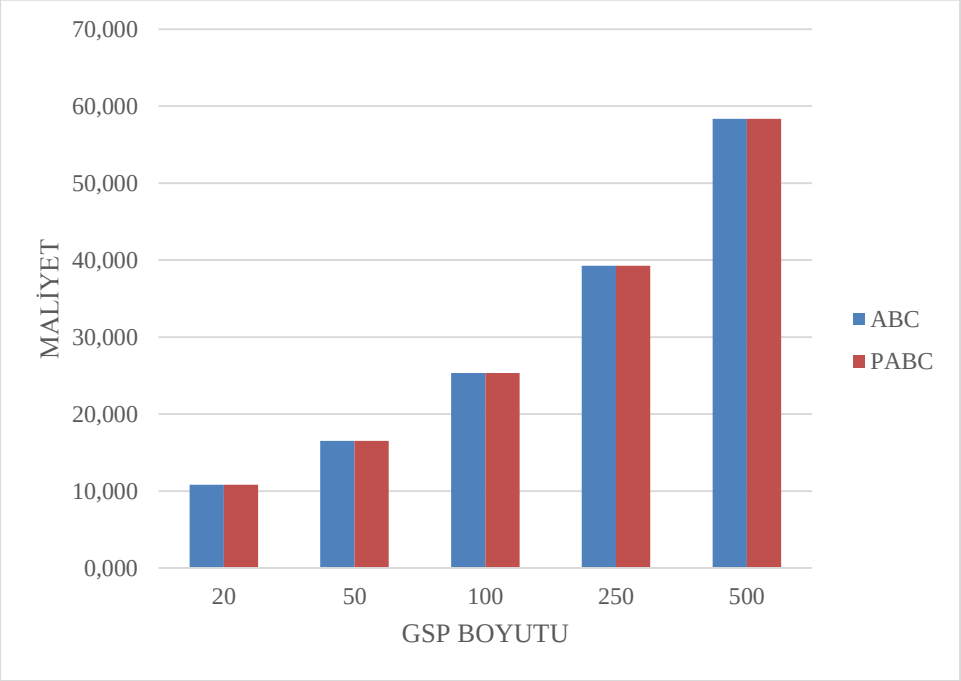
Şekil 9.3. A^3 yönteminin seri ve paralel uygulamasının maliyetlerinin değişik boyutlardaki GSP'ler için karşılaştırması.

Şekil 9.4'te A^3 yönteminin PA^3 yöntemi ile çalışma süresi yönünden karşılaştırmasını içeren grafik gösterilmektedir. İki yöntemin ortak olarak çalıştırdıkları GSP'lerin boyutları 20, 50, 100, 250 ve 500 şehirlidir. Bu GSP'ler için çalıştırılan iki yöntemin sonuçlarına göre PA^3 yöntemi A^3 yöntemine göre 4 işlemciye sahip bir bilgisayarda 3-3,5 katı daha hızlıdır. Bu sonuç A^3'ün paralel olarak uygulandığında veriminin yüksek olduğunu göstermektedir.



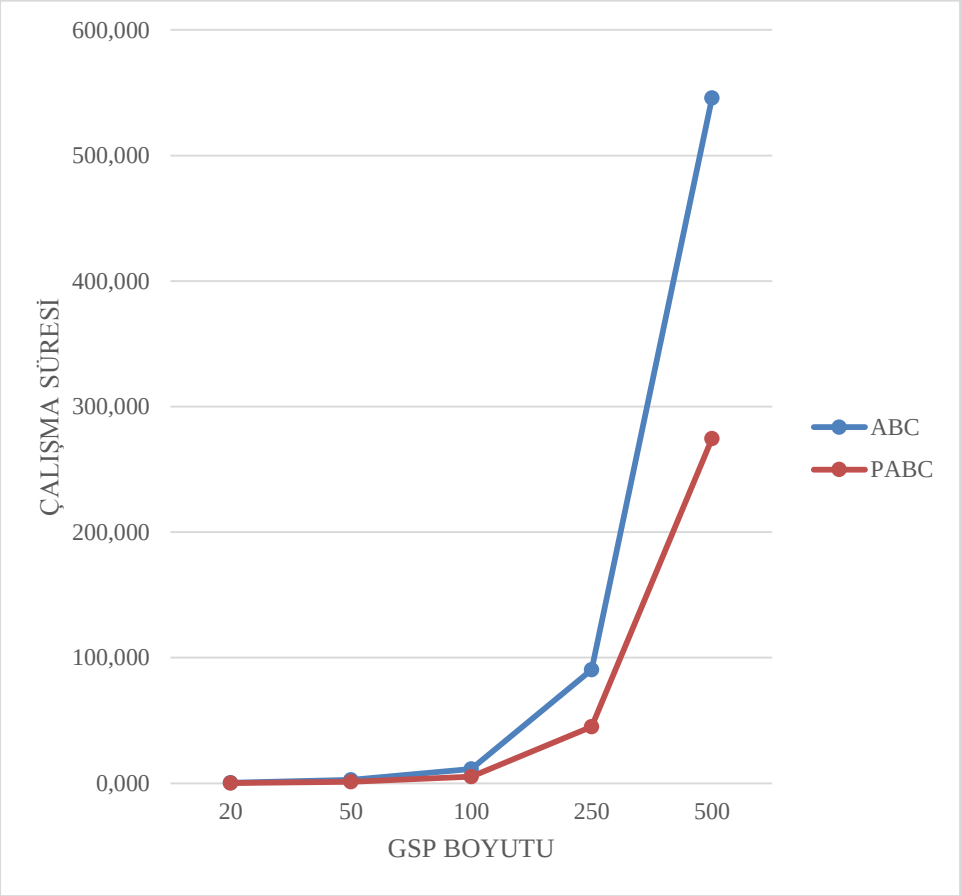
Şekil 9.4. A^3 yönteminin seri ve paralel uygulamasının çalışma sürelerinin değişik boyutlardaki GSP'ler için karşılaştırması.

Şekil 9.5'te ABC yönteminin PABC yöntemi ile maliyet yönünden karşılaştırmasını içeren grafik gösterilmektedir. İki yöntemin ortak olarak çalıştırdıkları GSP'lerin boyutları 20, 50, 100, 250 ve 500 şehirlidir. Bu GSP'ler için çalıştırılan iki yöntemin sonuçları arasında çok bir fark bulunmamaktadır. Yöntemler hesaplama açısından yaklaşık olarak aynıdır.



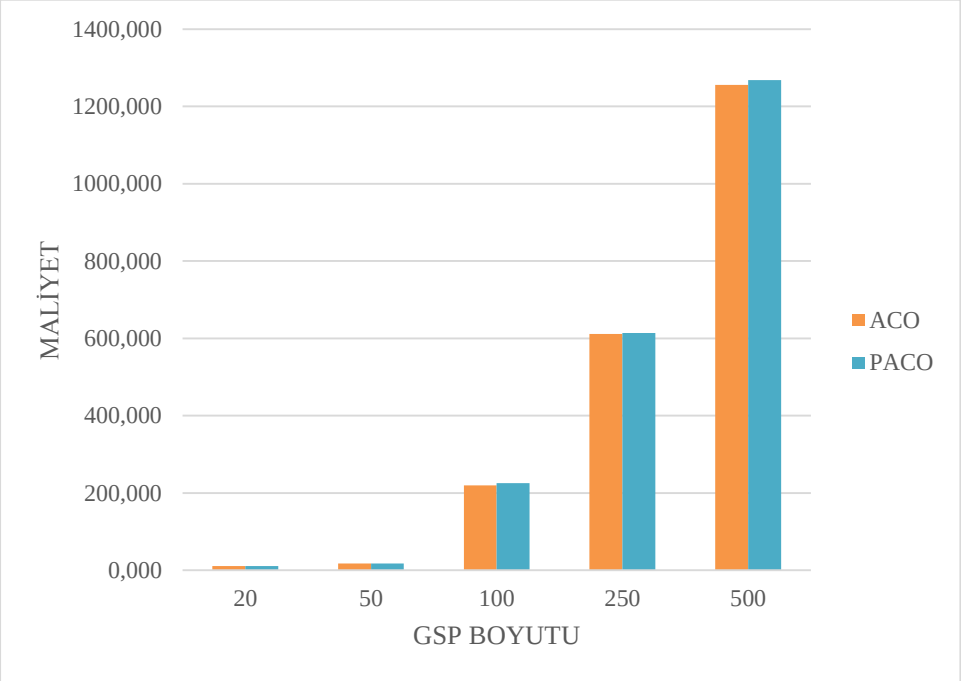
Şekil 9.5. ABC yönteminin seri ve paralel uygulamasının maliyetlerinin değişik boyutlardaki GSP'ler için karşılaştırması.

Şekil 9.6'da ABC yönteminin PABC yöntemi ile çalışma süresi yönünden karşılaştırmasını içeren grafik gösterilmektedir. İki yöntemin ortak olarak çalıştırdıkları GSP'lerin boyutları 20, 50, 100, 250 ve 500 şehirlidir. PABC'nin çalışma süresi, ABC'nin çalışma süresine göre yaklaşık olarak 2 kat daha hızlıdır.



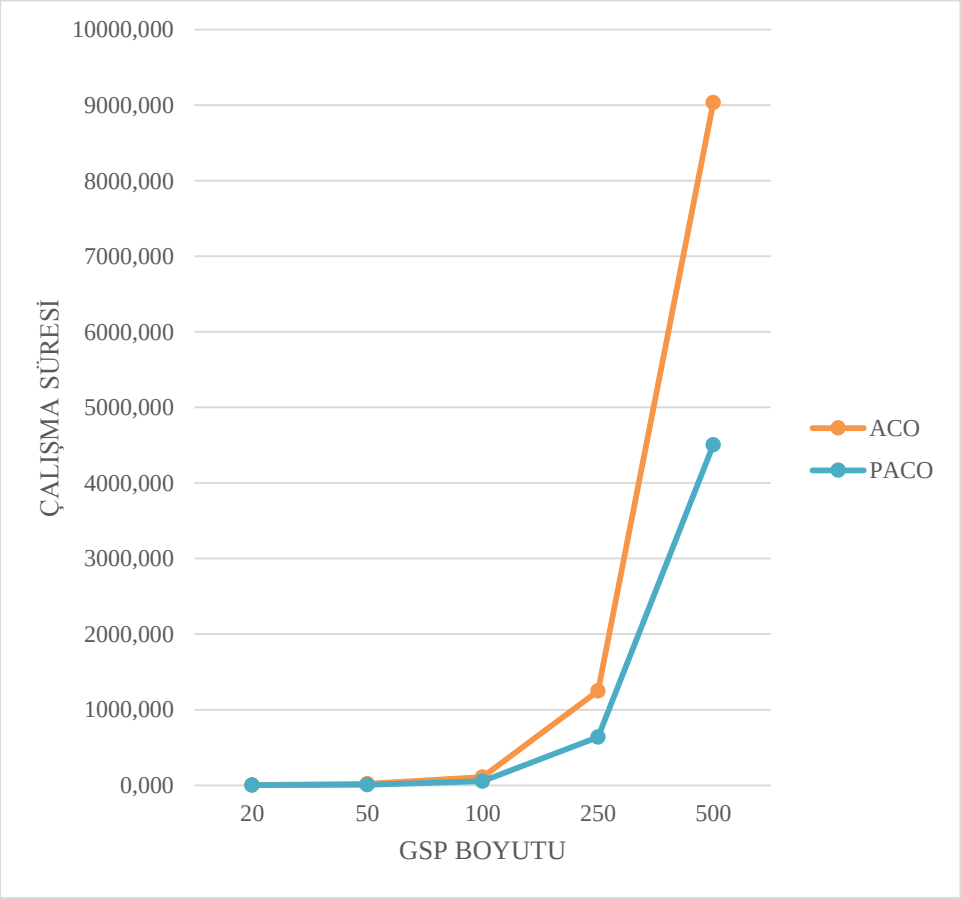
Şekil 9.6. ABC yönteminin seri ve paralel uygulamasının çalışma sürelerinin değişik boyutlardaki GSP'ler için karşılaştırması.

Şekil 9.7'de ACO yönteminin PACO yöntemi ile maliyet yönünden karşılaştırmasını içeren grafik gösterilmektedir. İki yöntemin ortak olarak çalıştırdıkları GSP'lerin boyutları 20, 50, 100, 250 ve 500 şehirlidir. Bu GSP'ler için çalıştırılan iki yöntemin sonuçları arasında çok bir fark bulunmamaktadır. Yöntemler hesaplama açısından yaklaşık olarak aynıdır.



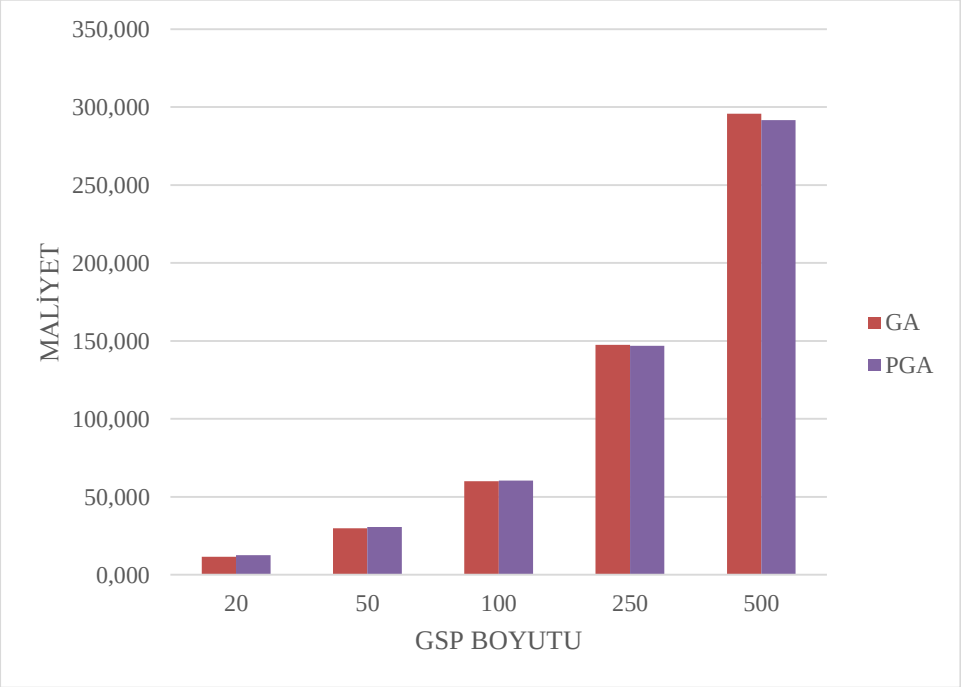
Şekil 9.7. ACO yönteminin seri ve paralel uygulamasının maliyetlerinin değişik boyutlardaki GSP'ler için karşılaştırması.

Şekil 9.8'de ACO yönteminin PACO yöntemi ile çalışma süresi yönünden karşılaştırmasını içeren grafik gösterilmektedir. İki yöntemin ortak olarak çalıştırdıkları GSP'lerin boyutları 20, 50, 100, 250 ve 500 şehirlidir. Bu GSP'ler için çalıştırılan iki yöntemin sonuçlarına göre PACO yöntemi ACO yöntemine göre çalışma süresi açısından şehir sayısı düşükken 20 ve 50 iken seri yöntem paralel yönteme göre daha hızlı çalışmıştır. GSP boyutu arttıkça çalışma süresi yaklaşık 2,5 kata kadar hızlanmaktadır. Düşük boyutlu GSP için PACO süre olarak dezavantajlıdır, bunun nedeni paralelleştirmede işlerin bölüştürülüp yeniden sonuçlarının değerlendirilmesi işlemin kendisinden çok daha fazla süre almasıdır. 4 işlemcili bir bilgisayarda sadece 2,5 kat hızlanmasının nedeni karıncaların geçtiği yollardaki lokal ve global feromon güncellemelerinin paralel olarak gerçekleştirilememesinden kaynaklanmaktadır.



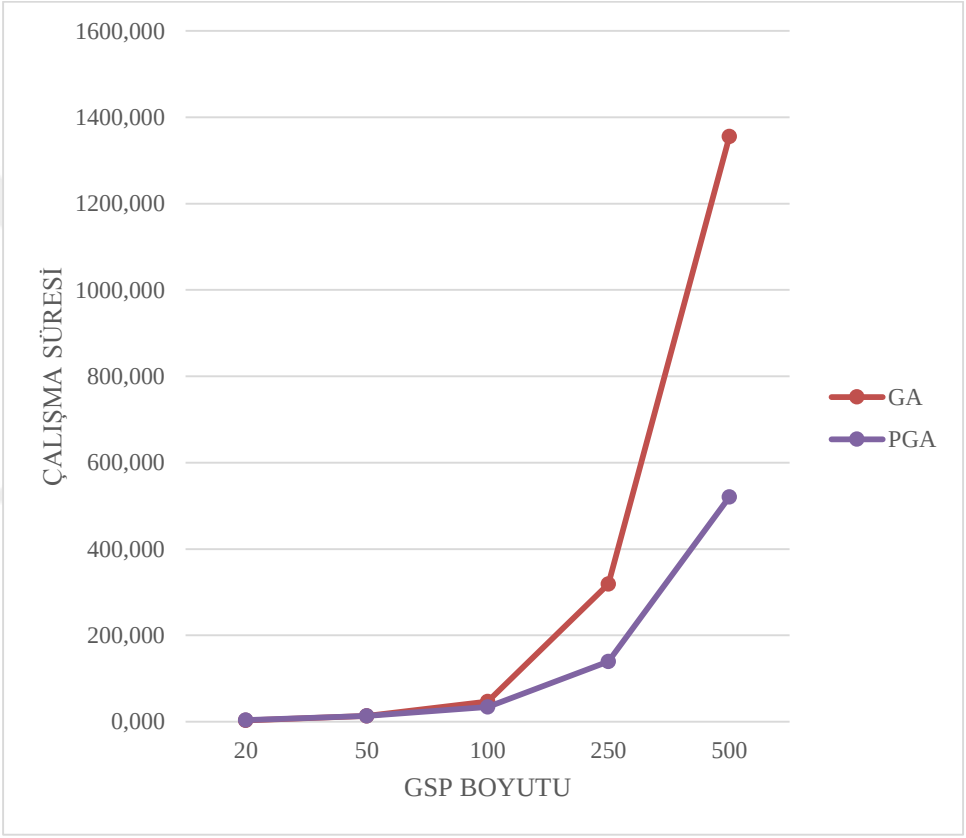
Şekil 9.8. ACO yönteminin seri ve paralel uygulamasının çalışma sürelerinin değişik boyutlardaki GSP'ler için karşılaştırması.

Şekil 9.9'da GA yönteminin PGA yöntemi ile maliyet yönünden karşılaştırmasını içeren grafik gösterilmektedir. İki yöntemin ortak olarak çalıştırdıkları GSP'lerin boyutları 20, 50, 100, 250 ve 500 şehirlidir. Bu GSP'ler için çalıştırılan iki yöntemin sonuçları arasında çok bir fark bulunmamaktadır. Yöntemler hesaplama açısından yaklaşık olarak aynıdır.



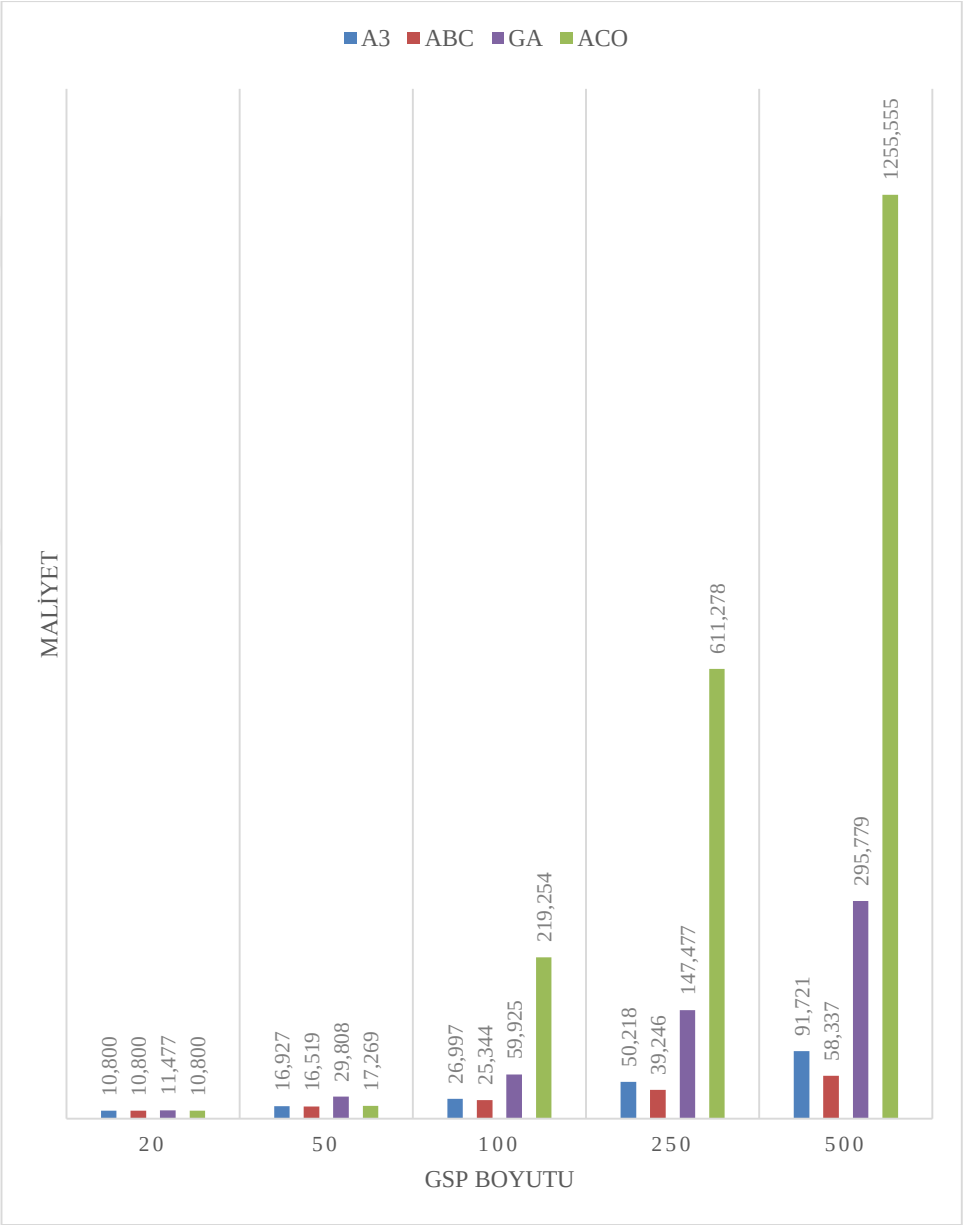
Şekil 9.9. GA yönteminin seri ve paralel uygulamasının maliyetlerinin değişik boyutlardaki GSP'ler için karşılaştırması.

Şekil 9.8'de GA yönteminin PGA yöntemi ile çalışma süresi yönünden karşılaştırmasını içeren grafik gösterilmektedir. İki yöntemin ortak olarak çalıştırdıkları GSP'lerin boyutları 20, 50, 100, 250 ve 500 şehirlidir. Bu GSP'ler için çalıştırılan iki yöntemin sonuçlarına göre PGA yöntemi GA yöntemine göre çalışma süresi açısından yaklaşık 2 kat daha hızlıdır. Paralel olarak uygulanan yöntemin verim olarak düşük olması GA'nın ara işlemlerde çaprazlama, mutasyon ve seçimde harcadığına yakın süre harcamasıdır.



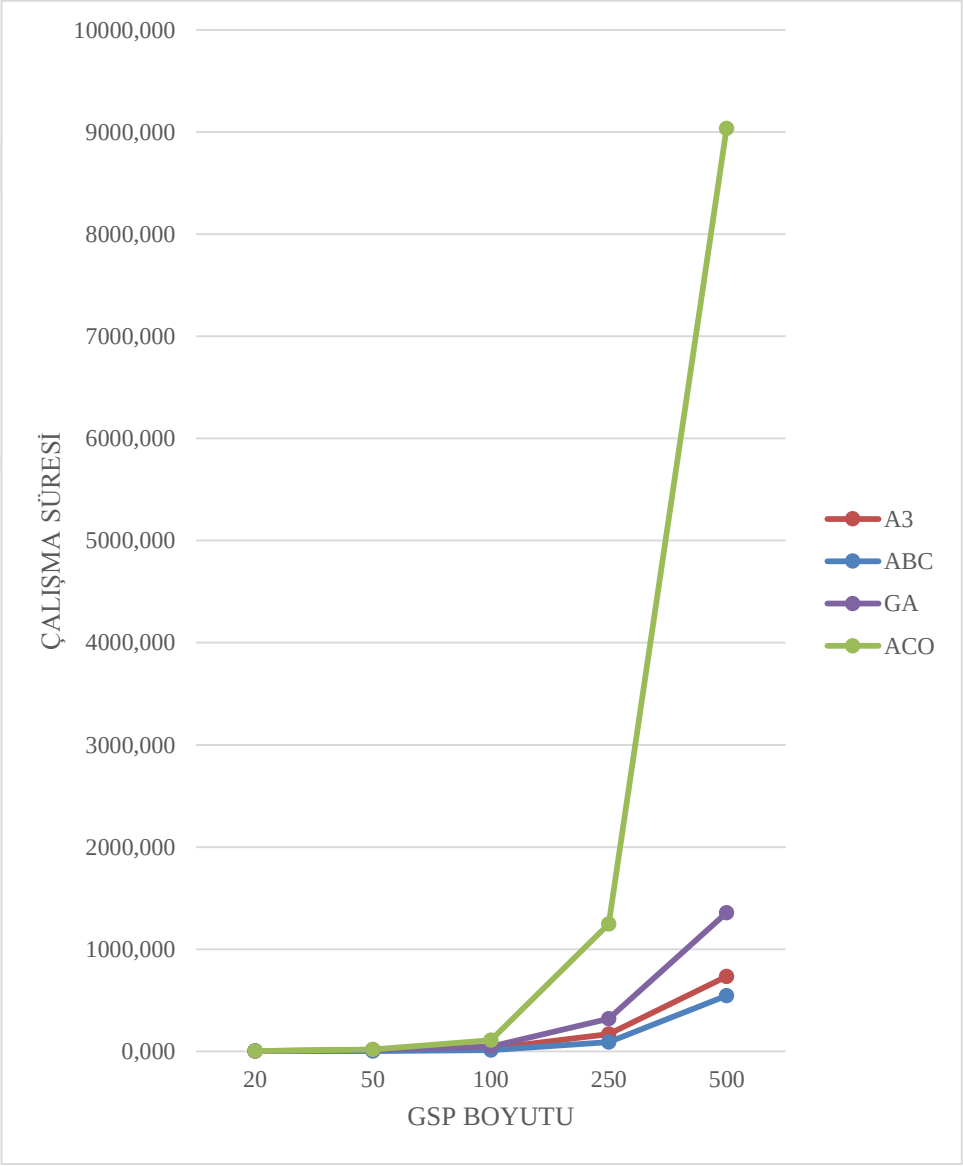
Şekil 9.10. GA yönteminin seri ve paralel uygulamasının çalışma sürelerinin değişik boyutlardaki GSP'ler için karşılaştırması.

Şekil 9.11'de seri yöntemlerin maliyet sonuçları karşılaştırılmıştır. Şehir sayısı arttıkça ACO ve GA yöntemlerinden elde edilen sonuçlar ABC ve A^3 yöntemlerinden elde edilen sonuçlara göre çok daha fazla artmıştır. Bu da büyük ölçekli GSP'ler için A^3 ve ABC'nin daha iyi olduğunu göstermektedir. En iyi sonuçlar ABC yöntemi tarafından elde edilmiştir ve A^3 yöntemi de ABC'ye yakın sonuçlar elde etmiştir.



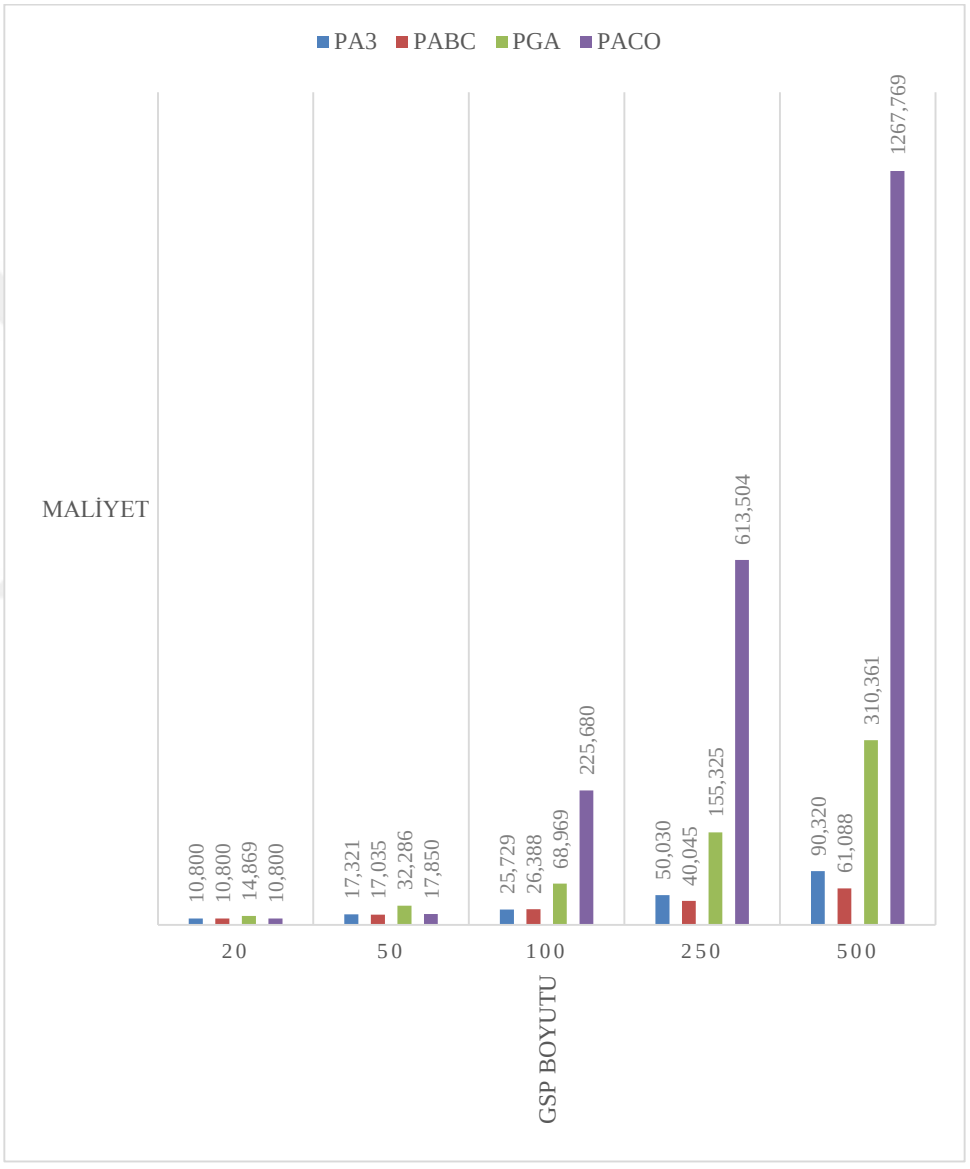
Şekil 9.11. Seri yöntemlerin uygulamalarının maliyetlerinin değişik boyutlardaki GSP'ler için karşılaştırması.

Şekil 9.12'de seri yöntemlerin çalışma süresi sonuçları karşılaştırılmıştır. Şehir sayısı ile çalışma süresi A^3, ABC ve GA yöntemleri için doğru orantılı olarak artarken ACO için üstsel olarak artmıştır. Bu, ACO'da karıncaların yol arama sürecinde, artan şehir sayısına bağlı olarak her bir karıncanın bir sonraki gideceği şehiri seçerken yaptığı karşılaştırma sayısının artmasından kaynaklanmaktadır. Çalışma süresi açısından bakıldığında ABC ve A^3 diğerlerine göre daha iydir. En iyi çalışma süresi ABC'ye, en kötü çalışma süresi ise ACO'ya aittir.



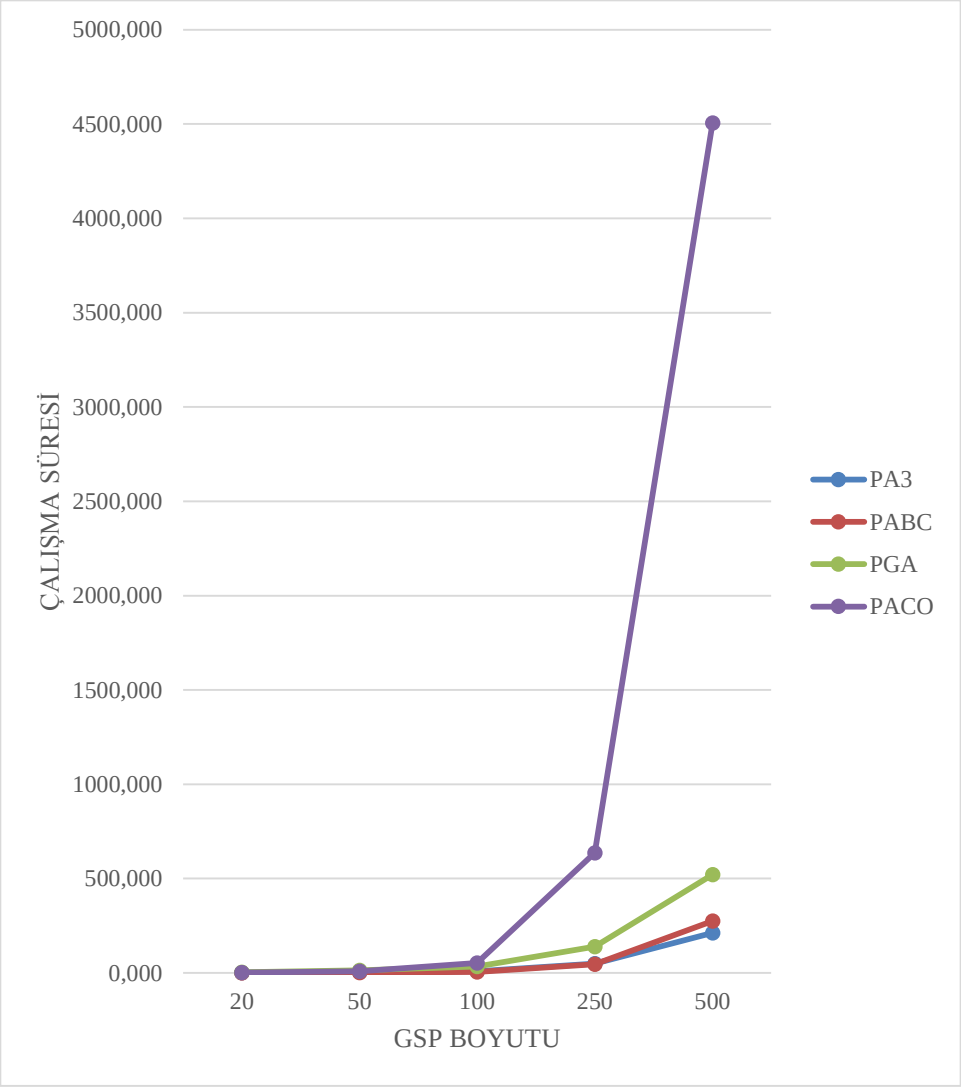
Şekil 9.12. Seri yöntemlerin uygulamalarının çalışma sürelerinin değişik boyutlardaki GSP'ler için karşılaştırması.

Şekil 9.13'te paralel yöntemlerin maliyet sonuçları karşılaştırılmıştır. Şehir sayısı arttıkça PACO ve PGA yöntemlerinden elde edilen sonuçlar PABC ve PA^3 yöntemlerinden elde edilen sonuçlara göre çok daha fazla artmıştır. Bu da büyük ölçekli GSP'ler için PA^3 ve PABC'nin daha iyi olduğunu göstermektedir. En iyi sonuçlar PABC yöntemi tarafından elde edilmiştir.



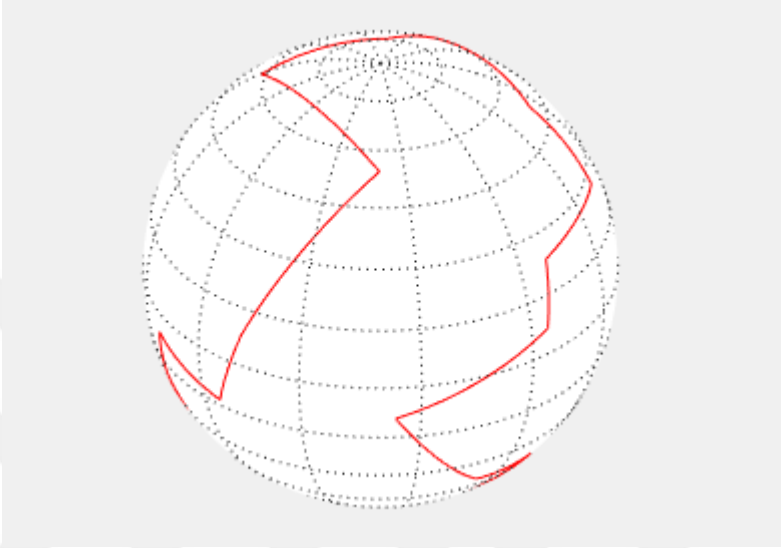
Şekil 9.13. Paralel yöntemlerin uygulamalarının maliyetlerinin değişik boyutlardaki GSP'ler için karşılaştırması.

Şekil 9.14'te paralel yöntemlerin çalışma süresi sonuçları karşılaştırılmıştır. Seri yöntemlerdeki gib şehir sayısı ile çalışma süresi A^3, ABC ve GA yöntemleri için doğru orantılı olarak artarken ACO için üstsel olarak artmıştır. Çalışma süresi açısından bakıldığında ABC ve A^3 diğerlerine göre daha iydir. En iyi çalışma süresi GA'ya, en kötü çalışma süresi ise ACO'ya aittir.

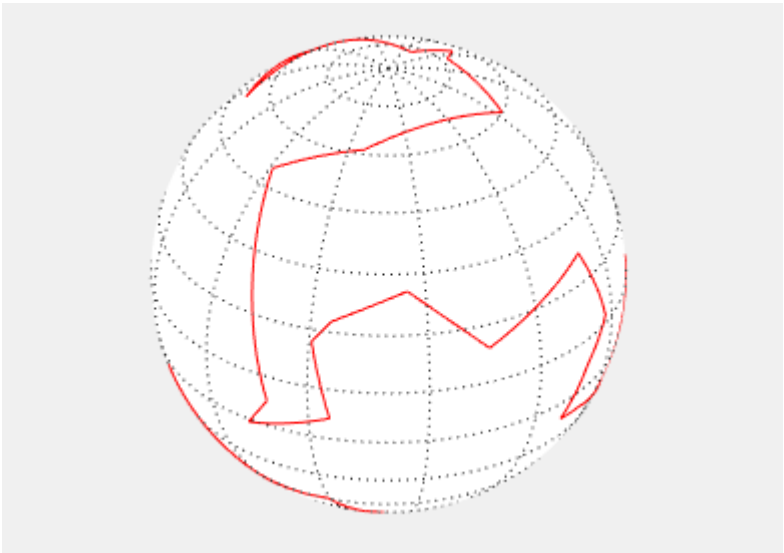


Şekil 9.14. Paralel yöntemlerin uygulamalarının çalışma sürelerinin değişik boyutlardaki GSP'ler için karşılaştırması.

Şekil 9.15'te 20 şehirli GSP, Şekil 9.16'da 50 şehirli GSP, Şekil 9.17'de 100 şehirli GSP, Şekil 9.18'de 250 şehirli GSP, Şekil 9.19'da 500 şehirli GSP üzerine uygulanan farklı yöntemlerden elde edilen en iyi sonuçların küre üzerine çizimi gösterilmiştir.

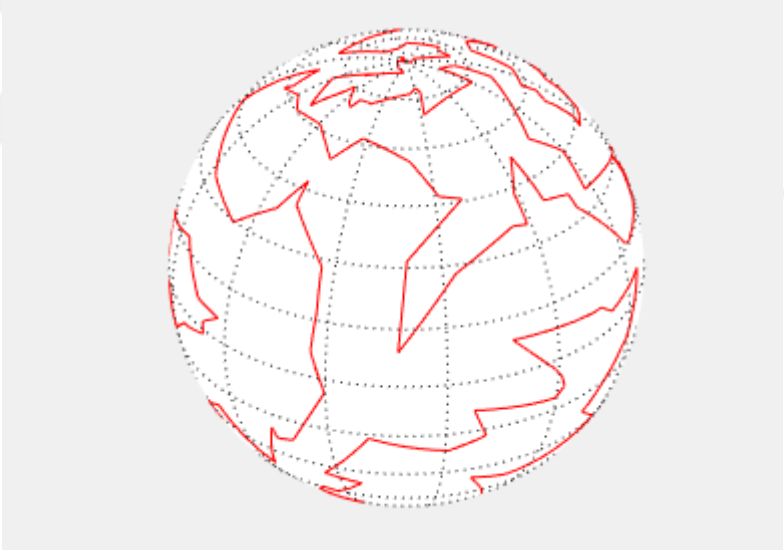


Şekil 9.15. 20 şehirli GSP için elde edilen en iyi sonucun küre üzerine çizdirilmesi.



Şekil 9.16. 50 şehirli GSP için elde edilen en iyi sonucun küre üzerine çizdirilmesi.

Şekil 9.17. 100 şehirli GSP için elde edilen en iyi sonucun küre üzerine çizdirilmesi.



Şekil 9.18. 250 şehirli GSP için elde edilen en iyi sonucun küre üzerine çizdirilmesi.

Şekil 9.19. 500 şehirli GSP için elde edilen en iyi sonucun küre üzerine çizdirilmesi.

BÖLÜM 10

SONUÇ

Meta-sezgisel algoritmalar bilim ve mühendislikteki çeşitli problerin çözümünde oldukça verimli sonuçlar verebilmektedir. Literatürde 100'den fazla sezgisel ve meta-sezgisel algoritmalar bulunmaktadır ve bulunmaya devam etmektedir. Genellikle doğadan esinlenen bu algoritmalar başlangıçta rastgele olası çözümlerden başlayarak en iyi çözüme ulaşmak için her adımda çözümde iyileştirmeye gitmektir.

Bu tez çalışmasında ilk olarak meta-sezgisel algortimalar incelenerek belirli gruplara ayrılmış ve kısaca açıklanmıştır. Daha sonra atomların kimyasal bileşik oluşturma sürecinden esinlenen A^3 optimizasyon algoritması, diğer doğadan esinlenerek modellenmiş ABC, ACO ve GA algortimaları küre üzerinde çeşitli boyutlardaki GSP'ler ile seri ve paralel olarak uygulanmış ve sonuçları karşılaştırılmıştır.

Bu yöntemler 4 çekirdekli CPU üzerinde C# programlama dili ile uygulanmış, boyutları 20, 50, 100, 250 ve 500 olan GSP için 20 defa çalıştırılarak test edilmiştir. Bu GSP haritaları rastgele olarak üretilmiş ve her yöntem için aynı haritalar kullanılmıştır.

Elde edilen sonuçlara göre en iyi maliyet daha küçük boyutlu GSP'ler için ABC tarafından hesaplanmıştır. Çalışma süresi olarak bakıldığında en hızlı çalışan yöntem ABC'dir.

Literatürde bulunan diğer algoritmalarla birebir karşılaştırma yapılamasa da genel itibariyle elde edilen sonuçlar literatürde elde edilen sonuçlarla benzer ya da çok daha iyidir.

Bu çalışmanın asıl amacı olan paralelleştirme için, orjinal algoritmadan 3,5 kat daha hızlı çalışması, A^3'ün diğer yöntemlere göre paralel olarak daha etkili olduğunu göstermektedir. Kovalent ve iyonik operatörlerin paralel olarak çalıştırılması ile uygulanan PA^3'ün, 4 işlemcili bir bilgisayarda 3,5 kat hızlanması bu algoritmanın %90'a kadar paralelleştirildiğini göstermektedir. A^3'ün paralelleştirilmesindeki verimlilik, ABC, ACO ve GA'ya göre daha yüksek olduğu görülmektedir. Çok daha büyük boyutlu algoritmalarda A^3 paralelleştirme olarak daha verimli olduğu için daha hızlı çalışacaktır.

Gelecekteki çalışmalarda, A^3 için daha iyi bir çalışma süresi performansı elde etmek için GPU üzerinde NVIDIA C programlama diline bir eklenti olarak sunulan bir mimari ve teknoloji olan CUDA (Compute Unified Device Architecture) ile uygulanabilir.

KAYNAKLAR

1. Weise, T., "Global Optimization Algorithms–Theory and Application", ***URL: Http://Www. It-Weise. De, Abrufdatum***, 1: 820 (2009).

2. Astolfi, A., "Optimization: An Introduction", New York, (2006).

3. Canayaz, M., "Cırcır Böceği Algoritması: Yeni Bir Meta-Sezgisel Yaklaşım Ve Uygulamaları", (2015).

4. Michalewicz, Z. and Fogel, D. B., "How to Solve It: Modern Heuristics", Second, Re. Ed., ***Springer***, (2004).

5. Grefenstette, J., Gopal, R., Rosmaita, B., and Van Gucht, D., .

6. Hassani, A. and Treijs, J., "An Overview of Standard and Parallel Genetic Algorithms", ***IDT Workshop On Interesting Results In Computer Science And Engineering, Mälardalen University***, 1–7 (2009).

7. Rasit Er, H. and Erdogan, N., "Parallel Genetic Algorithm to Solve Traveling Salesman Problem on MapReduce Framework using Hadoop Cluster", ***Jscse***, 3 (3): 380–386 (2013).

8. Kuzu, S., Önay, O., Şen, U., Tunçer, M., Yıldırım, B. F., and Keskintürk, T., "Gezgin Satıcı Problemlerinin Metasezgiseller ile Çözümü", ***Istanbul University Journal Of The School Of Business***, 43 (1): 1–27 (2014).

9. Papadimitriou, C. H. and Steiglitz, K., "Combinatorial optimization:Algorithms and complexity", ***IEEE Transactions On Acoustics, Speech, And Signal Processing***, 32 (6): 1258–1259 (1984).

10. Haxhimusa, Y., Carpenter, E., Catrambone, J., and Foldes, D., "2D and 3D Traveling Salesman Problem Abstract : Keywords :", 3 (2): (2011).

11. Grasses, "Machine Learning - K-Nearest Neighbor", (2017).

12. Jones, F., "Euclidean space", .

13. Aybars Ugur, S. K., "Genetic Algorithm based solution for TSP on a sphere", ***Earth***, 14 (3): 219–228 (2009).

14. Paper, C., "Route Optimization for Unmanned Aerial Systems İnsansız Hava Sistemleri için Güzergâh Optimizasyonu Route Optimization for Unmanned

Aerial Systems", (July 2017): (2016).

15. Eldem, H. and Ülker, E., "The application of ant colony optimization in the solution of 3D traveling salesman problem on a sphere", ***Engineering Science And Technology, An International Journal***, 20 (4): 1242–1248 (2017).

16. Eldem, H. and Ülker, E., "Küre Üzerinde 3 Boyutlu Gezgin Satıcı Problemi Çözümünde Parçacık Sürü Optimizasyonu Uygulaması", .

17. Chen, X., Zhou, Y., Tang, Z., and Luo, Q., "A hybrid algorithm combining glowworm swarm optimization and complete 2-opt algorithm for spherical travelling salesman problems", ***Applied Soft Computing Journal***, 58 (April): 104–114 (2017).

18. Zhou, Y., Wang, R., Zhao, C., Luo, Q., and Metwally, M. A., "Discrete greedy flower pollination algorithm for spherical traveling salesman problem", ***Neural Computing And Applications***, (August): 1–16 (2017).

19. Ouyang, X., Zhou, Y., Luo, Q., and Chen, H., "A novel discrete cuckoo search algorithm for spherical traveling salesman problem", ***Applied Mathematics And Information Sciences***, 7 (2): 777–784 (2013).

20. Stewart, A., "Spherical Geometry", .

21. Hearn, D. and Baker, M. P., "Computer Graphics - C Version", 619-620 (1996).

22. Çakmak, H., "A digital method to calculate the true areas of sunspot groups", ***Experimental Astronomy***, 37 (3): 539–553 (2014).

23. "Co-Ordinate Transformation Algorithms for the Hand-over of Targets between POEMS Interrogators", **https://www.eurocontrol.int/sites/default/files/content/documents/nm/surveillance/surveillance-mode-s-co-ordinate-transformation-algorithms-hand-over-of-targets-poems-annex-a.pdf** .

24. Internet: Weisstein and W., E., "Spherical Coordinates", **http://mathworld.wolfram.com/SphericalCoordinates.html** .

25. Matai, R., Singh, S., and Lal, M., "Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches", ***Traveling Salesman Problem, Theory And Applications***, (2010).

26. Özalp, N., "3 Boyutlu Arazi Üzerinde Çoklu Otonom İnsansız Hava Aracı Rota Planlaması", ***Tez Hava Harp Okulu***, 89 (2013).

27. Kiessling, A., "An Introduction to Parallel Programming with OpenMP", ***The University Of Edinburgh***, (April): 1–32 (2009).

28. Shekhar, S. and Xiong, H., "Parallel Computing", ***Encyclopedia Of GIS***, 843–843 (2008).

29. Woodard, L., "Introduction to Parallel Programming", (2013).

30. Internet: Barney, B. (Lawrence L. N. L., "Introduction to Parallel Computing", **https://computing.llnl.gov/tutorials/parallel_comp/#HybridMemory** .

31. Talbi, E.-G., "METAHEURISTICS FROM DESIGN TO IMPLEMENTATION", ***John Wiley & Sons, Inc., Hoboken, New Jersey***, (1965).

32. Dorigo, M. and Blum, C., "Ant colony optimization theory: A survey", ***Theoretical Computer Science***, 344 (2–3): 243–278 (2005).

33. Saha, S. K., Ghoshal, S. P., Kar, R., and Mandal, D., "Cat Swarm Optimization algorithm for optimal linear phase FIR filter design", ***ISA Transactions***, 52 (6): 781–794 (2013).

34. Karaboga, D., Gorkemli, B., Ozturk, C., and Karaboga, N., "A comprehensive survey: Artificial bee colony (ABC) algorithm and applications", ***Artificial Intelligence Review***, 42 (1): 21–57 (2014).

35. Gandomi, A. H., Yang, X. S., and Alavi, A. H., "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems", ***Engineering With Computers***, 29 (1): 17–35 (2013).

36. Yang, X.-S., "A New Metaheuristic Bat-Inspired Algorithm", 65–74 (2010).

37. Kennedy, J. and Eberhart, R., "Particle swarm optimization", ***Neural Networks, 1995. Proceedings., IEEE International Conference On***, 4: 1942–1948 vol.4 (1995).

38. Kohli, M. and Arora, S., "Chaotic grey wolf optimization algorithm for constrained optimization problems", ***Journal Of Computational Design And Engineering***, (2017).

39. Dasgupta, D., "Guest editorial: Special issue on artificial immune systems", ***IEEE Transactions On Evolutionary Computation***, 6 (3): 225–226 (2002).

40. Passino, K. M., "Biomimicry of bacterial foraging for distributed optimization and control", ***Control Systems, IEEE***, 22 (3): 52–67 (2002).

41. Nabil, E., "A Modified Flower Pollination Algorithm for Global Optimization", ***Expert Systems With Applications***, 57: 192–203 (2016).

42. Potvin, J.-Y., "Genetic Algorithms", ***Annals Of Operations Research***, 63: 339–370 (1996).

43. Storn, R. and Price, K., "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces", ***Journal Of Global Optimization***, 11 (4): 341–359 (1997).

44. Mitchell, M. and Taylor, C. E., "EVOLUTIONARY COMPUTATION: An

Overview", ***Annu. Rev. Ecol. Syst.***, (30): 593–616 (1999).

45. Huan T.T., Kulkarni A.J., Kanesan J., "Ideology algorithm: a socio-inspired optimization methodology", ***Neural Comput. Appl. (2016)***, 1–32 (2016).

46. Emami, H. and Derakhshan, F., "Election algorithm: A new socio-politically inspired strategy", ***AI Communications***, 28 (3): 591–603 (2015).

47. Lv, W., He, C., Li, D., Cheng, S., Luo, S., and Zhang, X., "Election campaign optimization algorithm", ***Procedia Computer Science***, 1 (1): 1377–1386 (2010).

48. Moosavian, N., "Soccer league competition algorithm for solving knapsack problems", ***Swarm And Evolutionary Computation***, 20: 14–22 (2015).

49. Rao, R. V., Savsani, V. J., and Vakharia, D. P., "Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems", ***CAD Computer Aided Design***, 43 (3): 303–315 (2011).

50. Ahmadi-Javid, A., "Anarchic Society Optimization: A human-inspired method", ***2011 IEEE Congress Of Evolutionary Computation, CEC 2011***, 2586–2592 (2011).

51. Ray, T. and Liew, K. M., "Society and civilization: an optimization algorithm based on the simulation of social behavior", ***IEEE Transactions On Evolutionary Computation***, 7 (4): 386–396 (2003).

52. Cui, Z., Xu, Y., and Zeng, J., "Social Emotional Optimization Algorithm with Random Emotional Selection Strategy", ***Theory And New Applications Of Swarm Intelligence***, (2012).

53. Husseinzadeh Kashan, A., "League Championship Algorithm (LCA): An algorithm for global optimization inspired by sport championships", ***Applied Soft Computing Journal***, 16: 171–200 (2014).

54. Ardalan, Z., Karimi, S., Poursabzi, O., and Naderi, B., "A novel imperialist competitive algorithm for generalized traveling salesman problems", ***Applied Soft Computing Journal***, 26: 546–555 (2015).

55. Satapathy, S. and Naik, A., "Social group optimization (SGO): a new population evolutionary optimization technique", ***Complex & Intelligent Systems***, 2 (3): 173–203 (2016).

56. Kuo, H. C. and Lin, C. H., "Cultural Evolution Algorithm for Global Optimizations and its Applications", ***Journal Of Applied Research And Technology***, 11 (4): 510–522 (2013).

57. Henderson, D., Jacobson, S., and Johnson, A., "THE THEORY AND PRACTICE OF SIMULATED ANNEALING", Handbook of Metaheuristics, (2003).

58. Ouyang, H. bin, Gao, L. qun, Li, S., Kong, X. yong, Wang, Q., and Zou, D. xuan, "Improved Harmony Search Algorithm: LHS", ***Applied Soft Computing Journal***, 53: 133–167 (2017).

59. Erol, O. K. and Eksin, I., "A new optimization method: Big Bang-Big Crunch", ***Advances In Engineering Software***, 37 (2): 106–111 (2006).

60. Karcı, A., "A new meta--heuristic algorithm based on chemical process: Atom algorithm", ***Proceedings Of 1st International Eurasian Conference On Mathematical Sciences And Applications, PriĢtine, Kosova***, (2012).

61. KARABOGA, D., "AN IDEA BASED ON HONEY BEE SWARM FOR NUMERICAL OPTIMIZATION", Kayseri/Türkiye, (2005).

62. Geem, Z. W., Kim, J. H., and Loganathan, G. V., "A New Heuristic Optimization Algorithm: Harmony Search", ***Simulation***, 75 (2): 60–68 (2001).

63. Erdogan Yildirim, A. and Karci, A., "YAPAY ATOM ALGORİTMASI VE AYRIK PROBLEMLERE UYGULANMASI", (2018).

64. KARADOĞAN, A. and KARCI, A., "Takviyeli Öğrenme İçin Yapay Atom Algoritması (A3) Kullanımı", (2014).

65. Yildirim, A. E. and Karci, A., "Applications of artificial atom algorithm to small-scale traveling salesman problems", ***Soft Computing***, (2017).

66. Erdogan Yildirim, A. and Karci, A., "Group elevator control optimization using artificial atom algorithm", ***IDAP 2017 - International Artificial Intelligence And Data Processing Symposium***, (September 2017): (2017).

67. Demir, M., "Veri Kümeleme de Yapay Atom Algoritması ve Cırcır Böceği Algoritmasının Karşılaştırılmalı Analizi", 2016 (November): (2016).

68. Cifci, E., Acma, B., and Selcuk, B., "Yapay atom algoritmasi kullanarak bireye Özgü Öğün programli beslenme Çizelgesi hazirlanmasi", ***2017 International Artificial Intelligence And Data Processing Symposium (IDAP)***, (November): 1–4 (2017).

69. Yildirim, A. E. and Karci, A., "Application of Traveling Salesman Problem for 81 Provinces in Turkey using Artificial Atom Algorithm", (April): (2018).

70. Michalewicz, Z., .

71. Bolat, B., Erol, K. O., and İrmak, C. E., "Genetic Algorithms in Engineering Applications an The Function of Operators", ***Journal Of Engineering And Natural Sciences***, 4 (0212): 264–271 (2004).

72. Mehboob, U., Qadir, J., Ali, S., and Vasilakos, A., "Genetic algorithms in wireless networking: techniques, applications, and issues", ***Soft Computing***, 20 (6): 2467–2501 (2016).

73. Cantú-Paz, E., "A Survey of Parallel Genetic Algorithms", ***Calcul. Paralleles Reseaux Syst. Repart.***, 10 (2): 141–171 (1998).

74. Dorigo, M. and Stützle, T., "Ant Colony Optimization", Encyclopedia of Machine Learning, 1-24 (2004).

75. Dorigo, M., Birattari, M., and Stutzle, T., "Ant colony optimization", ***IEEE Computational Intelligence Magazine***, 1 (4): 28–39 (2006).

76. Liu, H., Li, P., and Wen, Y., "Parallel Ant colony optimization algorithm", ***And Automation, 2006. WCICA 2006. The …***, 1 (2): 77–82 (2006).

77. Manfrin, M., Birattari, M., Stützle, T., and Dorigo, M., "Parallel Ant Colony Optimization for the Traveling Salesman Problem", ***Interface***, 4150 (9): 224–234 (2006).

78. Tsutsui, S. and Fujimoto, N., "Parallel ant colony optimization algorithm on a multi-core processor", ***Swarm Intelligence***, 488–495 (2010).

79. Sharma, P. and Gupta, M., "TSP problem using modified ABC based on dynamically division of bees", ***Proceedings - 1st International Conference On Computing, Communication, Control And Automation, ICCUBEA 2015***, 427–431 (2015).

80. Pathak, N., Mishra, M., and Kushwah, S. P. S., "Improved local search based modified ABC algorithm for TSP problem", ***Proceedings Of 2017 4th International Conference On Electronics And Communication Systems, ICECS 2017***, 17: 173–178 (2017).

81. Kıran, M. S., İşcan, H., and Gündüz, M., "The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem", ***Neural Computing And Applications***, 23 (1): 9–21 (2013).

82. Kocer, H. E. and Akca, M. R., "An improved artificial bee colony algorithm with local search for traveling salesman problem", ***Cybernetics And Systems***, 45 (8): 635–649 (2014).

83. Choong, S. S., "An Artificial Bee Colony Algorithm with a Modified Choice Function for the Traveling Salesman Problem", (2017).

84. El-Abd, M., "A hybrid ABC-SPSO algorithm for continuous function optimization", ***IEEE SSCI 2011 - Symposium Series On Computational Intelligence - SIS 2011: 2011 IEEE Symposium On Swarm Intelligence***, 96–101 (2011).

85. Khan, I. and Maiti, M. K., "A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem", ***Swarm And Evolutionary Computation***, (May): 1–11 (2018).

86. Li, D., Feng, Y., Zhong, J., Zhou, J., Yin, L., and Zhou, J., "Parallel Optimization Based on Artificial Bee Colony Algorithm", 955–959 (2017).

87. Sağ, T. and Çunkaş, M., "A new ABC-based multiobjective optimization algorithm with an improvement approach (IBMO: Improved bee colony algorithm for multiobjective optimization)", ***Turkish Journal Of Electrical Engineering And Computer Sciences***, 24 (4): 2349–2373 (2016).

88. Kita, H. and Sano, Y., "Genetic Algorithms for Optimization of Noisy Fitness Functions and Adaptation to Changing Environments Optimization of Noisy Fitness Functions", ***Evaluation***, 1–10 .

89. Aidov, A. and Dulikravich, G. S., "Modified continuous ant colony algorithm", ***2nd International Congress Of Serbian Society Of Mechanics (IConSSM)***, (June): 1–5 (2009).

90. Miles, R., "C# Programming Yellow Book", Department of COmputer Science, 214 (2016).

ÖZGEÇMİŞ

Ayşe Nur ALTINTAŞ TANKÜL 1989 yılında Gaziantep'te goğdu; ilk öğrenimine Sakarya'da başladı ve 2 Tekirdağ'da tamamladı. Orta öğrenimi Çankırı Süleyman Demirel Fen Lisesi'nde tamamladı. 2007 yılında İhasan Doğramacı Bilkent Üniversitesi'nde Bilgisayar Mühendisliği Bölümü'nde lisans eğitimine başladı ve eğitimini 2013 yılı Ocak ayında tamamladı. 2014 yılında yüksek lisans eğitimine Karabük Üniversitesi Bilgisayar Mühendisliği Bölüm'nde başladı ve 2018 yılında tamamladı. 2013 yılında Karabük Üniversitesi Bilgisayar Mühendisliği Bölümü Yazılım Anabilim Dalı'nda Araştırma Görevlisi olarak göreve başladı ve halen aynı yerde çalışmaya devam etmektedir.

ADRES BİLGİLERİ

Adres : Karabük Üniversitesi
Mühendislik Fakültesi Bilgisyar Mühendisliği
Balıklarkayası Mevkii / KARABÜK
Tel : (530) 244 2483
E-posta : altintasaysenur@gmail.com