



T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

KARINCA KOLONİ OPTİMİZASYONU (KKO) ve PARÇACIK SÜRÜ OPTİMİZASYONU (PSO) ALGORİTMALARI TEMELLİ BİR HİYERARŞİK YAKLAŞIM GELİŞTİRİLMESİ

Hüseyin ELDEM

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Haziran-2014

KONYA

Her Hakkı Saklıdır

ÖZET

YÜKSEK LİSANS TEZİ

KARINCA KOLONİ OPTİMİZASYONU (KKO) ve PARÇACIK SÜRÜ OPTİMİZASYONU (PSO) ALGORİTMALARI TEMELLİ BİR HİYERARŞİK YAKLAŞIM GELİŞTİRİLMESİ

Hüseyin ELDEM

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç.Dr. Erkan ÜLKER

2014, 89 Sayfa

Jüri
Prof. Dr. Ahmet ARSLAN
Doç. Dr. Erkan ÜLKER
Yrd. Doç. Dr. Mehmet HACIBEYOĞLU

Optimizasyon, herhangi bir sistemin olası tüm tasarımları arasından en iyisini bulmaktır. Belirli kısıtları olan bir problemin, sonucunu etkileyen parametre değerlerinin bulunarak, en kârlı sonucun minimum maliyetlerle belirlenmesini hedeflemek, problemin optimize edilmesi anlamını taşır. Optimizasyon problemlerini çözmek için kullanılan teknikler matematiksel ve sezgisel olarak sınıflandırılır. Matematiksel yöntemler, optimum sonucu bulabilseler de özellikle çözüm uzayı büyük olan problemlerin çözümlerinde çok fazla zaman harcamaktadırlar. Sezgisel yöntemler ise çözüm uzayının tümünü ele almadan sezgisel bir şekilde çok kısa sürelerde optimum sonuçlara ya da optimuma çok yakın sonuçlara ulaşabilmektedirler. Sezgisel algoritmalar, tüm tasarımlarda başarılı olmayabilir. Geliştirilen yöntemin her türden problemin çözümünde başarılı olmasını beklemek yerine hangi tür problemlerin çözümlerinde optimum sonuçlar ürettiğini bilerek buna göre sınıflandırılması uygundur.

Optimizasyon alanındaki geliştirilen metasezgisel algoritmalardan bazıları doğada yaşayan bazı canlıların yaşamlarını devam ettirebilmek için sergiledikleri hareketlerden yola çıkarak ortaya çıktığı bilinmektedir. Doğal fenomenlerden esinlenen algoritmalardan Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - KKO) özellikle ayrık optimizasyon problemlerin çözümünde, Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO) ise sürekli optimizasyon problemlerin çözümünde başarılı olmuşlardır.

Optimizasyon problemlerinin çözümünde keşfedilen yöntemlerin buldukları sonuçları iyileştirmek için bu yöntemler geliştirilmiştir. Ayrıca bu çözüm yöntemlerinin birlikte çalışması ile elde edilen çözümlerin iyileştirilmesi izlenen başka bir yoldur. Bu tez çalışmasında, KKO yönteminin ürettiği sonuçların iyileştirilmesi için, bulduğu çözümleri hiyerarşik bir yapıda, ayrıklaştırılmış PSO yönteminin iyileştirmesi üzerine bu iki yöntemin birlikte çalışabileceği ele alınmıştır. Literatürde optimizasyon problemlerinin çözümünde sıkça kullanılan test fonksiyonlarından Gezgin Satıcı Problemi (Traveling Salesman Problem - GSP) nin çözümünde, önerilen yöntem kullanılmıştır. Böylelikle hiyerarşik yöntemde uygun başlangıç çözümlerini üreten KKO algoritmasının bulduğu sonuçların PSO tarafından iyileştirilmesi sağlanmıştır. Ayrıca tez kapsamında, bu iki yöntemin hiyerarşik bir yaklaşımla komşuluk operatörleri yardımıyla birlikte çalışmaları test edilerek performans sonuçları verilmiştir. Tez kapsamında KKO ve PSO algoritmalarının yalın (standart) halleri de kullanılarak ürettiği sonuçlar saklanmış olup,

önerilen hiyerarşik yaklaşımın, standart KKO ve standart PSO algoritmaları sonuçları ile kıyaslandığında daha iyi sonuçlar elde ettiği görülmüştür.

Anahtar Kelimeler: Gezgin Satıcı Problemi, Karınca Kolonisi Optimizasyonu, Komşuluk Operatörleri, Metasezgisel Yöntemler, Optimizasyon, Parçacık Sürü Optimizasyonu, Sürü Zekâsı

ABSTRACT

MS THESIS

DEVELOPMENT OF A HIERARCHICAL APPROACH BASED ON ANT COLONY OPTIMIZATION (ACO) AND PARTICLE SWARM OPTIMIZATION (PSO) ALGORİTHMS

Hüseyin ELDEM

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN COMPUTER ENGINEERING

Advisor: Assoc. Prof. Dr. Erkan ÜLKER

2014, 89 Pages

Jury
Prof. Dr. Ahmet ARSLAN
Assoc. Prof. Dr. Erkan ÜLKER
Asst. Prof. Dr. Mehmet HACIBEYOĞLU

Optimization is finding the best, in any system among all possible design. It means optimizing problem, aiming most profitable results with minimal cost determination, by finding the parameter values that influence the outcome of a problem with certain limitations. Techniques used to solve optimization problems are classified as mathematical and heuristic. Although mathematical methods may find optimal results, they are spending too much time on problems that especially have big solution space. But heuristic methods achieve optimum results or result that very close to optimum in a very short time in an intuitive way without addressing all of the solution space. Heuristic algorithms may not be successful in all designs. Instead of waiting for the developed methodology to be successful for solution of any kind of problems, with knowing producing optimum results in which type of problems, classification is appropriate according to this.

Some of the meta-heuristics algorithms developed in the field of optimization based on movements of some animals living in nature that exhibit to continue their lives. Algorithms inspired from natural phenomena, Ant Colony Optimization (ACO) has been successful particularly in solution of discrete optimization problems, and Particle Swarm Optimization (PSO) in solution of continuous optimization problems.

These methods have been developed to improve the results of methods discovered in the solution of optimization problems. Also it is an another way to improve the solutions obtained from solutions of cooperating methods. In this thesis, for improving the results from the ACO method, obtained solution could work together with improving discrete PSO method in a hierarchical structure is covered. The proposed method has been used in the solution of Traveling Salesman Problem (TSP) that used frequently as a function test at the solution of the optimization problem in the literature. Thus, results founded by ACO algorithm that produces the appropriate starting solution in a hierarchical method was improved by PSO. Also withins the scope of the thesis, these two methods tested by working together with the help of neighborhood operators in a hierarchical approach and performance results are given. Within the scope of

the thesis, proposed hierarchical approach showed better results than the stored result produced by standard versions of ACO and PSO.

Keywords: Ant Colony Optimization, Metaheuristic Methods, Neighborhood Operators, Optimization, Particle Swarm Optimization, Swarm Intelligence, Travelling Salesman Problem

ÖNSÖZ

Bu çalışmamda bana yol gösteren ve akademisyen olma yolunda her türlü bilimsel katkıyı sağlayan çok kıymetli hocam ve danışmanım Doç. Dr. Erkan ÜLKER' e teşekkürü bir borç bilirim. Ayrıca fikirleriyle çalışmaya katkı sağlayan kıymetli hocam Yrd. Doç. Dr. Oğuz FINDIK' a teşekkürlerimi sunarım.

Beni her konuda yüreklendiren, çalışmalar boyunca her türlü hoşgörüyü ve fedakârlığı gösteren sevgili eşim Ayşe ELDEM' e, moral kaynağım oğlum Mehmet Ali' ye ve son olarak aramıza yeni katılan oğlum Furkan Eymen' e teşekkürlerimi sunarım.

Hüseyin ELDEM
KONYA-2014

İÇİNDEKİLER

ÖZET ..... **iv**

ABSTRACT ..... **vi**

ÖNSÖZ ..... **viii**

İÇİNDEKİLER ..... **ix**

1. GİRİŞ **1**

2. KAYNAK ARAŞTIRMASI **5**

2.1. Karınca Kolonisi Algoritması (KKO) ile yapılan çalışmalar 5
2.2. Parçacık Sürü Optimizasyonu (PSO) ile yapılan çalışmalar 6
2.3. Hibrit ve Hiyerarşik yöntemlerle yapılan çalışmalar 7

3. TEMEL KAVRAMLAR **10**

3.1. Optimizasyon 10
3.2. Sezgisel Optimizasyon 15
3.3. Metasezgisel Algoritmalar 16
3.4. Gezgin Satıcı Problemi (GSP) 18
3.5. Sürü Zekâsı 20
3.6. Karınca Optimizasyon Yöntemleri 21
3.6.1. Tanım 21
3.6.2. Karınca Sistemi (Ant System) 24
3.6.3. Elit Karınca Sistemi (Elitist Ant System) 28
3.6.4. Max-Min Karınca Sistemi (Max-Min Ant System) 29
3.6.5. Rütbeye Dayalı Karınca Sistemi (Rank-Based Ant System) 30
3.6.6. Karınca-Yoğunluk, Karınca-Çevrim ve Karınca - Nicelik (Ant Density, Ant Cycle and Ant Quantity 30
3.6.7. Ant Q 33
3.6.8. Karınca Koloni Sistemi (Ant Colony System) 33
3.7. Parçacık Sürü Optimizasyonu 35
3.7.1. Tanım 35
3.7.2. PSO Algoritması 37

4. KOMŞULUK OPERATÖRLERİ İLE KKO-PSO TABANLI HİYERARŞİK YÖNTEM **43**

4.1. Komşuluk Operatörleri 43
4.2. PSO Algoritmasının Ayrıklaştırılması 46
4.3. KKO & PSO Hiyerarşik Yaklaşım 47
4.4. Deneysel Çalışmalar 51

5. SONUÇLAR VE ÖNERİLER **64**

5.1 Sonuçlar 64

5.2 Öneriler 66

KAYNAKLAR 68

EKLER 73

ÖZGEÇMİŞ 88

SİMGELER VE KISALTMALAR

Simgeler

$\Re$	: Tasarım alanı - arama uzayı
f(x)	: Amaç fonksiyonu
Φ_j ve ψ_k	: Kısıtlayıcılar
γ	: Q-Learning Öğrenme Katsayısı
$\tau_{ij}(t)$	: Feromon izi yoğunluğu
ρ	: Buharlaşma Katsayısı
d_{ij}	: i ve j noktaları arası uzaklık
η_{ij}	: Görünürlük ($1/d_{ij}$)
α	: Feromon Katsayısı
β	: Görünürlük Katsayısı
q	: Rastgele Seçim Parametresi

Kısaltmalar

GA	: Genetik Algoritma (Genetic Algorithms)
PSO	: Parçacık Sürü Optimizasyonu (Particle Swarm Optimization)
KKO	: Karınca Kolonisi Optimizasyonu (Ant Colony Optimization)
GSP	: Gezgin Satıcı Problemi (Travelling Salesman Problem)
AGSP	: Asimetrik Gezgin Satıcı Problemi
SGSP	: Simetrik Gezgin Satıcı Problemi
KS	: Karınca Sistemi (Ant System)
ACS	: Karınca Koloni Sistemi (Ant Colony System)
ASrank	: Rütbeye Dayalı Karınca Sistemi (Rank Based Ant System)
MMAS	: Maksimum-Minimum Karınca Sistemi (MAX-MIN Ant System)
SA	: Benzetimli Tavlama (Simulated Annealing)
DEA	: Diferansiyel Evrim Algoritması (Differential Evoluation Algorithm)
NP-Zor Hard)	: Deterministik olmayan Polinomsal Zor (Non-deterministic Polinominal
ABC	: Yapay Arı Kolonisi Algoritması (Artificial Bee Colony Algorithm)
gbest	: Genel En İyi (Global Best)
pbest	: Yerel En İyi (Local Best)
TSPLIB	: Gezgin Satıcı Problemleri Kütüphanesi

1. GİRİŞ

İnsanoğlu, varoluşundan bu güne kadar gerekli incelemeleri ve gözlemleri yaparak mevcut problemlere ait çözümlerin iyileştirilmesi anlamında sürekli yeni yöntemler bulma çabasını devam ettirmektedir. Bilinen problemlerin var olan çözüm yöntemlerine yenilerinin eklenmesinde en önemli faktörler şüphesiz ki o çözümün iyileştiğine işaret eden, maliyetlerin ve sürenin azalmasıdır. Bir başka deyişle, çözüm kalitesinin artırılması ve uygun hesaplama zamanının bulunması hedeflenmelidir. Kısacası bir problemi etkileyen kısıtların, problemin türüne göre maksimize edilmesi ya da minimize edilmesi daha verimli çözümlerin elde edilmesine katkı sağlamaktadır.

Problemlerin çözümünü matematiksel hesaplamalar yaparak kesin olarak bulabilen klasik yöntemlerin kabiliyetleri, uzun hesaplama sürelerinin oluşması ve maliyetlerin artmasından dolayı azalmaktadır. Problemi etkileyen unsurlar arttıkça zorluk seviyesi artan ve *Non-deterministic Polynomial-time Hard* (*NP-Hard*) problem sınıfında değerlendirilen çözüm uzayı çok büyük olan problemlerin çözümünde *sezgisel optimizasyon* tekniklerinden faydalanılmaktadır. Sezgisel optimizasyon teknikleri sayesinde NP-Hard problemlerin çözümünde, çözüm uzayının tamamı taranmadan, sezgisel bir şekilde çok daha kısa hesaplama sürelerinde optimum ya da optimuma en yakın sonuçlar elde edilebilmektedir.

Sezgisel optimizasyon tekniklerinden bir kısmı *sürü zekası* (*Swarm Intelligence*) algoritmaları olarak isimlendirilmektedir. Sürü zekâsı algoritmalarının temelinde doğada yaşayan bazı canlıların yaşamlarını devam ettirebilmek için sergiledikleri davranışlar bulunmaktadır. Özellikle koloni halinde yaşayan ve birbirleriyle görev paylaşımı içinde bulunan canlıların savunma, yiyecek arayışı, yuva inşa etme ya da mevcut yuvaya erişme gibi problem çözme kabiliyetlerinden esinlenerek, bilinen dünya problemlerine ait yeni çözüm yaklaşımları elde edilebilmiştir. Koloni halinde yaşayan canlılardan karıncaların, gözlemler sonucu, yiyecek toplama ve yuvalarına geri dönüşte önlerindeki engellere rağmen en kısa yolu bulma becerileri ve kolonideki diğer canlılarında zamanla bir etkileşimle kolektif bir şekilde hareket edebilmeleri, kendilerine has bu problemi optimize edebildiklerini göstermektedir. Doğada yaşayan canlıların birliktelikleri ile kendi problemlerinin çözümünü bulmalarından ortaya çıkan sürü zekâsı algoritmalarına, Diferansiyel Gelişim Algoritması (Differential Evoluation Algorithm, DEA), Genetik Algoritma (Genetic Algorithm, GA), Benzetilmiş Tavlama (Simulated Annealing, SA), ***Parçacık Sürü Optimizasyonu*** (*Particle Swarm*

Optimization, PSO) ve ***Karınca Koloni Optimizasyonu*** (*Ant Colony Optimization, KKO*) yöntemleri başta olmak üzere çok sayıda *doğa esinli* sezgisel optimizasyon teknikleri örnek verilebilir.

KKO yöntemi, ilk olarak 1992 yılında Marco Dorigo tarafından doktora tezi olarak önerilmiştir (Dorigo, 1992). KKO, graflar üzerindeki olası yolları araştırarak hesaplama problemlerinin çözümü için kullanılan olasılıksal *metasezgisel* bir yöntemdir. Gerçek karıncaların salgıladıkları *feromon* kimyasalı sayesinde yuva ve yiyecek arasındaki mesafeyi bir süre sonra en aza indirmeleri kendilerince bu problemlerini optimize ettiklerini göstermektedir. Gerçek karıncaların bir takım özelliklerinden faydalanarak ve geçici hafıza gibi bir takım özellikler eklenerek elde edilen yapay karıncalarla gerçek mesafe matrislerinin üzerinde test edilen KKO yöntemi, birçok optimizasyon problemlerinin çözülmesinde başarılı olmuştur (Boussaïd ve ark., 2013).

Optimizasyon problemlerinin çözümünde kullanılan metasezgisel yöntemlerden birisi olan *PSO,* 1995'te Dr. Eberhart ve Dr. Kennedy tarafından geliştirilmiş popülasyon tabanlı sürü zekası kategorisinde değerlendirilen sezgisel bir optimizasyon tekniğidir (Kennedy ve Eberhart, 1995). Kuş sürülerinin gıda ararlarken sergiledikleri birliktelikten esinlenilerek ortaya çıkan bu yöntem, çok modelli lineer olmayan karmaşık optimizasyon problemlerinin çözümünde sıklıkla kullanılmıştır (Boussaïd ve ark., 2013). *PSO'* da rastgele çözümlerden oluşan başlangıç popülasyonundaki bireyler, *parçacık* olarak adlandırılır. Her bir parçacığın durum uzayında hareket ettiği varsayılır ve her parça potansiyel çözümü taşır. Her parça en iyi durumu hatırlayabilir ve parçacıklar kendi arasında bilgi alışverişinde bulunabilirler (Kennedy ve Eberhart, 1995).

Gezgin satıcı problemi (*Travelling Salesman Problem - TSP*), *ayrık optimizasyon* problemlerinin çözümünde kullanılan test fonksiyonlarından birisidir (Dokümanın sonraki kısımlarında Gezgin satıcı problemi, GSP şeklinde kısaltma olarak kullanılacaktır). GSP, bilgisayar bilimlerinde, verilerin modellenmesinde kullanılan Hamilton döngüsü olarak ta ifade edilmektedir. Graf teorisinde kullanılan yol biçimi belirlemesi olarak ifade edilen Hamilton döngüsü, her düğüme bir kez uğrayarak en kısa kapalı yolun bulunmasıdır. GSP, gezgin satıcının minimum maliyetle şehirleri seyahat ederek başlangıç noktasına geri dönmesi olarak tanımlanmaktadır. GSP, tanım gereği kolay bir problem gibi gözükmekte olup literatürde NP-Hard problemler sınıfında değerlendirilmektedir.

Gezgin satıcı problemlerinde KKO ve PSO yalın (saf, standart) halleriyle denenmiş ancak her zaman optimum sonuca ulaşılması mümkün olmamıştır. Bu iki algoritmanın yalın halleriyle optimum sonuca ulaşmalarında birbirlerinin eksikliklerini tamamlayarak optimum sonuca ulaşabilmek için daha verimli bir çalışma sergileyecekleri öngörülebilir. Literatürde de, genellikle metasezgisel yöntemlerin yapısal olarak benzer yöntemlerle yardımlaşmasıyla optimizasyon problemlerinin sonuçlarını iyileştirilebildiği görülmektedir (Marinakis ve ark., 2005; Tsai ve ark., 2004; Yun ve ark., 2013; Gündüz ve ark. 2014).

KKO, karıncaların yuva – yiyecek arasındaki en kısa yolu bulmadaki kolektif becerileri ve kolonideki bağımsız her bir karıncanın davranışlarının birliktelik hareketine dönüşmesini ele alan ***ayrık*** optimizasyon yöntemlerinden birisidir. **PSO** ise, kuşların yön tayininde birbirlerinden faydalanmaları ve balık sürülerinin sosyal davranışlarından esinlenilerek geliştirilen popülasyon temelli sürü zekası kategorisinde değerlendirilen ***sürekli*** optimizasyon yöntemlerindendir. Bu tez çalışmasında, KKO ile PSO yöntemlerinin *komşuluk operatörleri* yardımıyla birlikte çalışmasını sağlayan ***hiyerarşik*** bir yöntem geliştirilmiştir. Gezgin satıcı problemlerinin ayrık problem olmasından dolayı hiyerarşik yöntemde KKO nun yapısına uygun şekilde PSO yöntemi ayrıklaştırılmıştır. Böylece, KKO nun yerel minimumlara takılarak optimum sonucu bulmasının zorlaşması engeli; PSO nun KKO tarafından üretilen çözümleri, çözüm uzayının farklı bölgelerine aktarması ve komşuluk operatörleri yardımıyla çeşitlendirmesi ile aşılmıştır. GSP çözümlerinde sıklıkla kullanılan KKO yönteminin performansı; komşuluk operatörleri ve ayrıklaştırılmış PSO algoritması yardımıyla artırılarak, geliştirilen hiyerarşik yeni bir yöntem bu tez çalışmasında önerilmiştir.

Geliştirilen KKO-PSO tabanlı hiyerarşik yöntemin performansı, literatürde sıklıkla kullanılan benchmark problemlerinden birisi olan GSP problemine uygulanarak ele alınmıştır. Ayrıca çalışmada komşuluk operatörlerinin KKO-PSO hiyerarşik yönteme katkısı karşılaştırılmıştır. Performans testlerinde, yedi farklı komşuluk operatörü, belirli popülasyon yüzdelerinde GSP lere uygulanmıştır.

Tezin organizasyonu şu şekildedir:

KKO-PSO tabanlı hiyerarşik yeni bir yaklaşımın önerildiği bu çalışmanın *ikinci bölümünde* KKO, PSO algoritmalarının optimizasyon problemlerinin çözümünde kullanıldığı literatürdeki çalışmalara ait bir kaynak araştırması sunulmuştur.

Tez çalışmasında kullanılan temel kavramların açıklandığı *üçüncü bölümde* optimizasyon, optimizasyon algoritmaları, sezgisel optimizasyon, sürü zekası, metasezgisel algoritmalar, GSP, KKO ve PSO hakkında detay bilgiler verilmiştir.

Deneysel çalışmaların yer aldığı *dördüncü bölümde* komşuluk operatörleri, PSO yönteminin ayrıklaştırılması ve önerilen KKO-PSO hiyerarşik yöntem açıklanmış ve GSP problemleri üzerindeki başarısını sergileyen deneysel testlerin sonuçları sunulmuştur.

Son olarak, sonuçlar ve önerilerin ele alındığı *beşinci bölümde* ise yapılan tez çalışması hakkında genel değerlendirmeler yapılarak yöntemin başarısı hakkında yorumlar verilmiştir. Ayrıca gelecekte optimizasyon alanında yapılacak yeni çalışmalar için öneriler de sunulmuştur.

2. KAYNAK ARAŞTIRMASI

Gerçek dünyadaki problemlerin çözümleri için önerilmiş ve geliştirilmiş birçok metasezgisel algoritma mevcuttur. Önerilen yöntemlerin performanslarının artırılması için bu yöntemlerin zamanla geliştirilmiş versiyonları literatüre kazandırılmıştır. Literatürde izlenen bir başka yol ise, mevcut metasezgisel yöntemlerin bir arada kullanılarak hibrit ve hiyerarşik yöntemlerin geliştirilmesidir. Kaynak araştırmasında öncelikle, yalın KKO ve PSO yöntemleri kullanılarak yapılmış çalışmalara yer verilmiştir. Sonrasında metasezgisel yöntemleri hibrit ve hiyerarşik yapıda kullanan çalışmalar hakkında bilgiler sunulmuştur.

2.1. Karınca Kolonisi Algoritması (KKO) ile yapılan çalışmalar

Karınca Algoritması, ilk kez Marco Dorigo (1992) tarafından "Optimization, Learning and Natural Algorithms" isimli doktora tezi ile en kısa yolun bulunmasında kullanılan temel prensipleri ile ele alınmıştır. Buradaki algoritmaya Karınca Sistemi (Ant System - AS) adını vermiş, algoritmadaki parametrelerin uygunluk değerlerini saptamış, değişik boyutlardaki birçok GSP problemlerinde deneyerek 75 şehirden az, küçük ölçekli GSP problemlerinde başarılı sonuçlar elde etmiştir (Dorigo, 1992).

Algoritma üzerinde ilk iyileştirme yine Marco Dorigo tarafından elitist yaklaşım olarak bilinen, her bir iterasyonda belirlenen en kısa tur uzunluğuna çekiciliği artırmak için daha fazla feromon ekleyerek daha iyi sonuçlar elde edilmesini sağlayan yöntem geliştirilmiştir (Dorigo, 1992).

Dorigo ve Gambardella (1996-1997) tarafından sınırlı düğüme sahip problemlerin makul bir hesaplama süresi içinde iyi sonuçların elde edilebildiği ve yüksek optimizasyon başarısının sağlandığı Karınca Koloni Sistemi (Ant Colony System – ACS) geliştirilmiştir.

Bullnheimer ve ark., tarafından (1999) elitist yaklaşımın gelişmişi olarak rütbe yada dereceye bağlı olarak artan miktarda feromon bırakıldığı, Rütbeye Dayalı Karınca Sistemi (Rank Based Ant System - AS_{rank}) geliştirilmiştir.

Thomas Stützle ve Holger H. Hoos (1996, 2000) mevcut algoritmanın feromon miktarının belirli sınırlar arasında (üst sınır, alt sınır aralığında) kalmasını sağlayacak

şekilde algoritmada değişiklik yapmışlardır. Bu algoritmaya sınırlamalardan dolayı Maksimum-Minimum Karınca Sistemi (MAX-MIN Ant System - MMAS) adını vermişlerdir.

Gambardella ve Dorigo tarafından (1995) Q-Learning algoritmasındaki öğrenme katsayısının (γ) katkısıyla feromon güncellemesinin hesaplandığı Ant-Q geliştirilmiştir. GSP çözümlerinde Karınca Koloni Sistemi ile aynı performanslarda sonuçlar üretmektedir.

KKO yöntemi ve türevleri GSP dışında başka benchmark problemlerinin çözümünde de kullanılmıştır (Dorigo ve Stützle, 2004) . Tez çalışmasında GSP' ye odaklanıldığı için, GSP çözümünde KKO yöntemini kullanan çalışmalara ait örnekler aşağıda verilmiştir.

Gambardella ve Dorigo (1996), karınca kolonisi yöntemiyle simetrik ve asimetrik GSP problemlerinin çözümünü ele almışlardır.

Jun-man ve Yi (2012), geleneksel KKO daki karıncaların yönlendirme ile ilgili işlem maliyetlerini azaltmak için varyasyon stratejisini kullanarak geliştirdikleri KKO ile genelleştirilmiş GSP problemlerinin bazı örneklerinde başarılı sonuçlar elde etmişlerdir.

Tuba ve Jovanovic (2013), GSP çözümlerinde kullanılmak üzere, KKO nun durgunluk davranışını önlemek için, yerel minimumlara takılmasını önleyen yeni bir KKO yöntemi geliştirmişlerdir.

Ilie ve ark. (2013), KKO için yapılandırılabilir dağıtık bir mimari önermiştir. Çoklu ajan sistemi ortamında, uyumsuz ajanlar arasındaki mesaj yönetimini azaltarak çözümü zor GSP örneklerinde deneysel sonuçlar sunmuştur.

2.2. Parçacık Sürü Optimizasyonu (PSO) ile yapılan çalışmalar

Shi ve ark. tarafından (2007) optimum çözümlere yakınsamayı hızlandırmak amacıyla, GSP ve genelleştirilmiş GSP problemleri için belirsiz arama stratejisi ve çaprazlama teknikleri kullanılmasıyla PSO algoritmasını ayrıklaştırılmış ve 19 farklı GSP için test etmiş ve başarılı sonuçlar elde etmiştir.

Wang ve ark. tarafından (2005) belirli bir olasılığa göre parçacıkların hareket hızlarını kendisi ve sürüdeki en iyiye göre de belirleyerek değiştirilen PSO yöntemi ile GSP çözümlerinde test edilmiştir.

Zhang ve ark. tarafından (2009) tek düğüm düzeltmesi (single node adjustment) algoritması ile geliştirilen PSO yöntemi GSP çözümlerinde kullanılmıştır.

Jin-rong (2011), greedy algoritmasından esinlenip genetik algoritmadaki çaprazlama ve mutasyon operatörlerini de kullanarak çok nesneli GSP' lerin çözümünde kullanılan bir PSO yöntemi geliştirmiştir.

Fan (2010), takas (swap) operatöründen esinlenerek PSO daki hız hesaplamasını yeniden tanımlamıştır. Ayrıca GSP çözümleri için önerdiği ayrık PSO algoritmasına sezgisel faktör, çaprazlama operatörü ve uyarlamalı gürültü faktörünü de eklemiştir.

Liao ve ark. (2012) GSP çözümlerinde kullanılmak üzere parçacıklar arası bilgi değişimini geliştirmek için genetik tabanlı bir PSO yöntemi önermişlerdir.

Yan ve ark. (2012) yerel optimuma kolayca takılan genetik algoritmanın dezavantajlarını önlemek için PSO algoritmasını GSP çözümlerinde kullanmışlar ve etkili sonuçlar elde etmişlerdir. Önerilen PSO yöntemiyle bulunan sonuçlar klasik genetik algoritma sonuçları ile karşılaştırılmıştır.

Chen ve Chien (2011) GSP çözümleri için parçacık sürü optimizasyonu teknikleriyle genetik benzetimli tavlamalı karınca koloni sistemi olarak isimlendirdikleri yeni bir yöntem geliştirmişlerdir.

2.3. Hibrit ve Hiyerarşik yöntemlerle yapılan çalışmalar

Dong ve ark. (1994), denetimsiz öğrenme algoritması (unsupervised learning algorithm) ve özyinelemeli hopfield sinir ağı yöntemlerini hiyerarşik olarak birleştirmiş ve büyük ölçekli GSP çözümlerinde kullanmıştır.

Lee (2004), uygun çözümler bulan ancak erken yakınsama ile yerel minimumlara takılabilen KKO yöntemini GA ile yerel minimumlara takılmaktan kurtarmayı amaçlayarak KKO ve GA tabanlı hibrit bir yöntem önermişler ve etkili optimum sonuçlar bulduklarını göstermişlerdir.

White ve Yen (2004), GSP çözümleri için GA nın şehir seçimleri sırasında mesafe koruma çaprazlaması (distance preserving crossover) yöntemini kullanarak evrimsel hibrit bir yöntem önermişlerdir.

Tsai ve ark. (2004) büyük ölçekli GSP lerin çözümü için çoklu karınca klanları ile KKO yönteminin hibritleştirildiği sezgisel bir yaklaşım önermişlerdir.

Marinakis ve ark. (2005) genişleyen komşu arama (Expanding Neighborhood Search) tekniği ve Lagrangean Relaxation yöntemlerinin GA ile hibritleştirilmesiyle elde ettikleri ve GRASP ismini verdikleri yöntemlerini GSP çözümlerinde kullanmışlardır.

Gomez ve ark. (2007) A-B Domain olarak isimlendirdikleri bir framework tabanlı ACS ve PSO tabanlı bir yöntem geliştirerek GSP çözümlerinde kullanmışlardır.

Shelokar ve ark. (2007) geliştirilmiş sürekli optimizasyon (non-convex) problemleri için PSO ve KKO tabanlı hibrit bir yöntem geliştirmişlerdir.

Takahashi (2009), genişletilmiş değiştirici çaprazlama operatörleri yardımıyla düzenlenen genetik algoritmaların KKO yöntemiyle birlikte oluşturdukları hibrit yöntemi GSP çözümlerinde kullanmışlardır.

Marinakis ve ark. (2010) birden fazla sürü kullanarak klasik PSO' nun keşif yeteneklerini geliştirmek için geri bildirim prosedürü kullanan yeni bir hibrit yöntem geliştirmişlerdir.

Kıran ve ark (2012) PSO ve KKO tabanlı hibrit bir algoritma geliştirerek numerik optimizasyon problemlerinde global minimumun bulunması için çalışmalar yapmışlardır.

Kuo ve ark. (2012) dinamik kümeleme probleminin çözümü için PSO ve GA tabanlı hibrit bir yöntem geliştirmişlerdir.

Deng ve ark. (2012a) genetik algoritmalarındaki evrim olgusu ile GA, PSO ve KKO tabanlı iki aşamalı yeni bir sürü zekası optimizasyon algoritması önermişlerdir.

Dong ve ark. (2012b) GSP problemleri çözümlerinde kullanılmak üzere CGAS ismini verdikleri kooperatif genetik karınca sistemi hibrit algoritmasını önermişlerdir. Önerilen hibrit yöntemle GSP çözümlerinde kullanılan KKO yönteminin performansının artırılması amaçlanmıştır.

Kıran ve Gündüz (2013), Yapay Arı Kolonisi (Artifical Bee Colony, ABC) ve PSO algoritmaları hibritleştirilerek HPA kısaltması ile yeni bir hibrit yöntem önermişlerdir. 12 tane numerik benchmark fonksiyonu üzerinde test edilen bu yöntem ile global ve yerel arama yetenekleri artırıldığından sonuçların iyi olduğu gösterilmiştir.

Yöntem sürekli optimizasyon problemlerinin çözümünde bir alternatif olarak önerilmiştir.

Yun ve ark. (2013) Genişletilmiş Armoni Araması (Advanced Harmony Search) algoritması ile KKO yöntemini GSP çözümleri için hibritleştirerek etkili bir yöntem önermiştir. Önerilen yöntem TSPLIB den alınan 26 benchmark problemi üzerinde test edilmiştir.

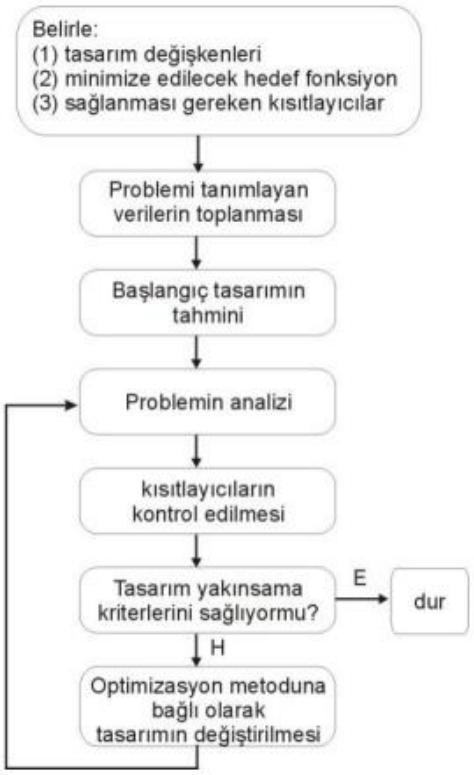
Gündüz ve ark. (2014) tur inşasında KKO, oluşturulan turların geliştirilmesinde ise ABC yöntemlerini hiyerarşik olarak bir arada kullanarak GSP problemleri üzerinde deneyler yaparak iyi sonuçlar elde etmişlerdir.

Literatürde görüldüğü üzere araştırmalar sonucu, KKO ve PSO yu hiyerarşik olarak komşuluk operatörleri yardımıyla GSP çözümünde kullanılan bir çalışma bulunmamaktadır. Bu yüzden bu tez çalışması özgün bir çalışma olarak değerlendirilmektedir.

3. TEMEL KAVRAMLAR

3.1. Optimizasyon

Optimizasyon, bir sistemin tasarlanmasında olası tüm çözümlerin arasından en iyisinin bulunması olarak ifade edilebilir. Belirli kısıtları olan, bir problemin sonucunu etkileyen parametre değerlerinin bulunarak en kârlı sonucun minimum maliyetlerle belirlenmesini hedeflemek, problemin optimize edilmesi anlamını taşır. Her bir gerçek dünya probleminde gerekli çaba, sermaye, malzeme ve işçiliğin minimum seviyede belirlenmesi ve kazancın maksimum düzeyde olması en gerçekçi amaç olmuştur. Optimizasyon işleminde problemin çözümünü belirleyen karar değişkenlerinin belirlenmesi, sonrasında ise bu karar verici parametreler ışığında minimize edilecek maliyet fonksiyonu ya da maksimize edilecek kâr fonksiyonları tanımlanmalıdır (Amaç fonksiyonu). Bunların tanımlanmasında problemi sınırlayan, karar değişkenlerinin alabileceği değer ya da değer aralıklarını ifade eden sınırlamaların belirtilmesi gerekmektedir. Probleme göre bazı kısıtlamalar eşitsizlik, bazıları ise eşitlikler şeklinde olabilmektedir (kısıtlayıcılar). Şekil 3.1 de bir sistem tasarlanırken optimizasyona ait yukarıda açıklanan adımlar şematize edilmiştir.



Şekil 3.1 Bir tasarımın optimizasyonu

Optimizasyon, geniş bir yelpazede ele alınacak problemlerin kesin optimum sonuçlarının araştırılması amaçlarını içermektedir. Bu yüzden optimizasyon problemlerinin çok farklı isimlendirilmeleri ve sınıflandırılmaları mevcuttur. Genellikle optimizasyon teknikleri problemden probleme önemli ölçüde değişebilmektedir. Her bir optimizasyon probleminin çözümü için tek bir yaklaşım söz konusu olamamaktadır. Çünkü her bir problemin karmaşıklığı, önemli oranda kısıtlayıcıları ve amaç fonksiyonlarına bağlı olduğundan çok farklılık gösterebilmektedir.

Bir çok optimizasyon problemi, genel olarak matematiksel bir gösterime şu şekilde sahiptir:

$f_i(x)$, $\Phi_j(x)$, $\psi_k(x)$, tasarım vektörünün fonksiyonları iken;

$$\text{Minimize et } x \in \Re \qquad f_i(x), (i = 1, 2, ..., M), \tag{3.1}$$

$$\text{Amaç} \qquad \Phi_j(x) = 0, (j = 1, 2, ..., J), \tag{3.2}$$

$$\psi_k(x) \leq 0, (k = 1, 2, ..., K), \tag{3.3}$$

$$x = (x_1, x_2, ..., x_n)^T . \tag{3.4}$$

Bu gösterimde, x tasarımının herhangi bir x_i bileşeni *tasarım değişkeni* olarak adlandırılmaktadır. Tasarım değişkeni, *sürekli* (continuous), *ayrık* (discrete) ya da bu ikisinin *karışımı* (mixed) olabilmektedir. $\Re$ arama uzayında, Φ_j ve ψ_k optimizasyon probleminin kısıtlayıcıları olarak anılmaktadır. Bu kısıtlayıcılardan Φ_j eşitlik kısıtlayıcıları, ψ_k ise eşitsizlik kısıtlayıcılarıdır (Yang, 2010).

Optimizasyon problemlerindeki sınıflandırma, hedef sayısı, kısıtlayıcıların sayısı, fonksiyonların yapısı, hedef fonksiyonunun yapısı, tasarım değişkenlerinin türü, değerlerdeki belirsizlik ve hesaplama zorluğuna göre olmaktadır (Yang, 2010).

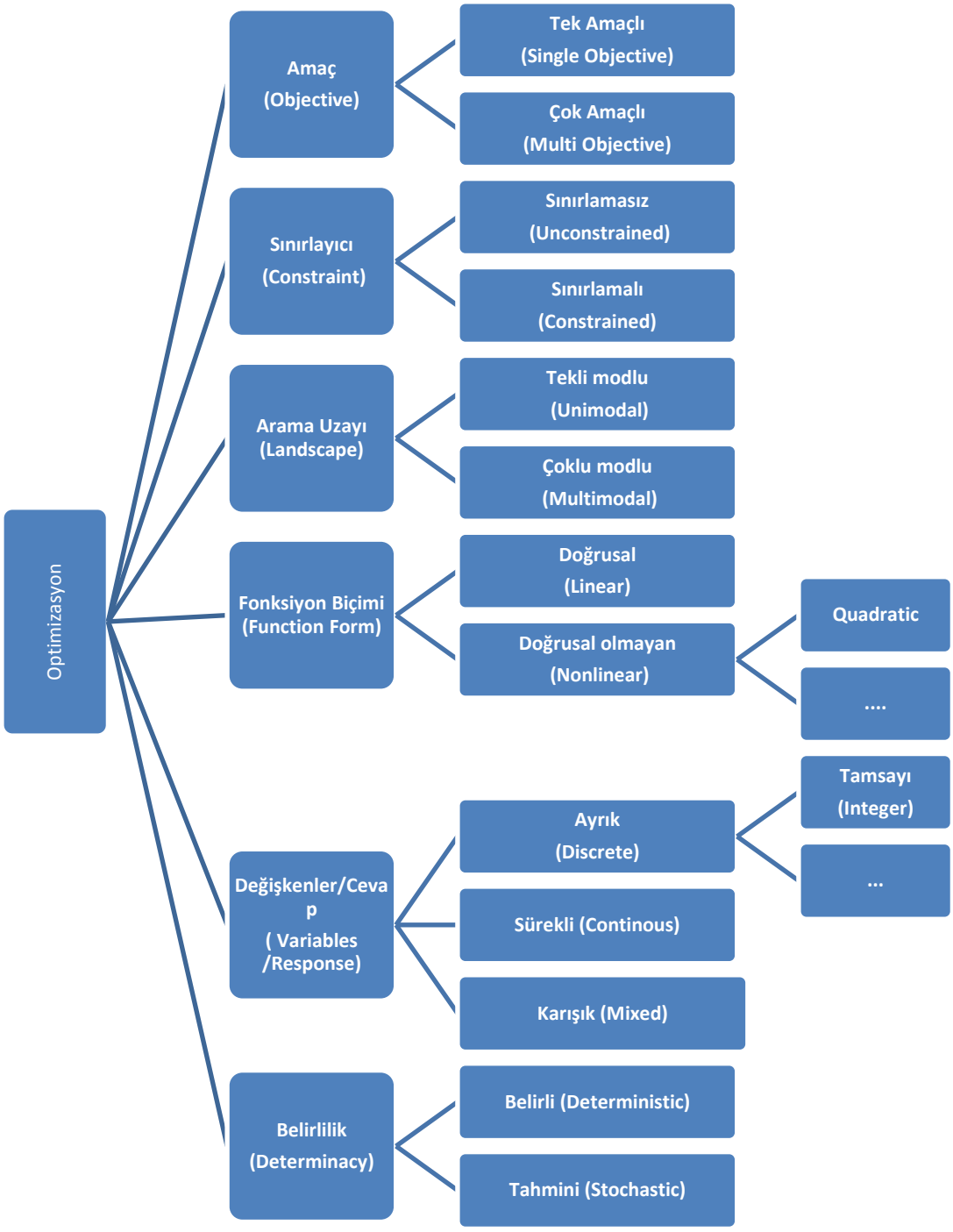
Eşitlik 3.1' de minimize edilecek $f_i(x)$ fonksiyonu hedef sayısı bakımından sınıflandırıldığında M=1 ise *tek* bir amaç, M>1 ise *çok* amaçlı olmaktadır. Gerçek dünya problemlerinin bir çoğu *çok amaçlı optimizasyon* problemleri olarak değerlendirilmektedir. Örneğin bir araba motoru tasarlanmasında yakıt verimliliğini maksimize etmek, karbon dioksit emisyonunu en aza indirmek ve gürültü seviyesini düşürmek gibi çok kriter göz önüne alınmalıdır.

Benzer şekilde, optimizasyon problemlerinin kısıtlayıcıları bakımından bir sınıflandırma yapmak mümkündür. Eşitlik 3.2 ve 3.3 teki ' $\Phi_j(x)$ ve $\psi_k(x)$ optimizasyon probleminin kısıtlayıcı sayıları J ve K toplamı 0 ise problem *kısıtlamasız optimizasyon*

problemidir. $K=0$ ve $J \geq 1$ ise eşitlik kısıtlayıcı, $K \geq 1$ ve $J=0$ ise eşitsizlik *kısıtlayıcı optimizasyon* problemi olarak isimlendirilmektedir (Yang, 2010).

Fonksiyonların yapısı bakımından optimizasyon problemleri *lineer* ve *lineer olmayan* (nonlineer) şekilde ikiye ayrılmaktadır. $\Phi_j(x)$ ve $\psi_k(x)$ optimizasyon probleminin kısıtlayıcılarının her ikisi de lineer ise problem lineer (doğrusal) kısıtlı olmaktadır. Amaç fonksiyonu $f_i(x)$ ve kısıtlayıcıların hepsi doğrusal ise doğrusal optimizasyon, bunlardan herhangi birinin doğrusal olmaması durumunda ise problem doğrusal olmayan optimizasyon problem (nonlinear optimization problem) olarak adlandırılmaktadır.

Tasarım değişkenlerinin değer türü de sınıflandırma için kullanılabilir. Tüm tasarım değişkenlerinin değerleri ayrık ise, optimizasyon ayrık optimizasyon problemi (discrete optimization problem) olarak isimlendirilir. Tasarım değişkenlerinin ondalıklı değer yerine tam sayı değerlerin aldığı ayrık optimizasyon problemlerine; graf teorisi ve rotalama kapsamında ele alınan GSP, minimum yayılan ağaç problemi (minimum spanning tree problem), araç rotalama (vehicle routing), havayolu planlama (airline scheduling) ve sırt çantası problemi (knapsack problem) örnek olarak verilebilir. Eğer tasarım değişkenleri sürekli ya da ondalıklı değerlere sahip oluyorsa optimizasyon, sürekli optimizasyon problemi (continuous optimization problem) olarak isimlendirilmektedir. Karışık optimizasyon problemleri ise (mixed optimization problems) tasarım değişkenlerinin hem ayrık hem de sürekli olduğunda bir sınıflandırma olarak nitelendirilebilir (Yang, 2010). Şekil 3.2' de optimizasyon problemlerinin sınıflandırılması gösterilmektedir.



Şekil 3.2 Optimizasyon problemlerinin sınıflandırılması, (Yang, 2010).

Optimizasyon problemlerinin çözümünde kullanılan algoritmalar iki kategoride değerlendirilebilir: Bunlardan ilki belirli bir prosedürü takip eden, takip edilen yolun, tasarım değişkenleri ve fonksiyon değerlerinin tekrarlanabildiği deterministik (rastgele olmayan, belirli) algoritmalardır. Yani algoritma ne zaman çalıştırılırsa çalıştırılsın aynı girdiler için aynı sonuçları her zaman üretebilen algoritmalar *deterministik* yaklaşımla

açıklanmaktadır. Diğeri ise rastsallığı içinde barındıran *stokastik* (olasılıksal) algoritmalardır. Ayrıca, deterministik ve stokastik algoritmaların bir arada kullanıldığı *hibrit* yaklaşımlarda üçüncü bir tür optimizasyon algoritması olarak ele alınmaktadır. Optimizasyon problemlerinin çözümünde genellikle melez yaklaşımlar sıklıkla tercih edilebilmektedir. Şekil 3.3' te optimizasyon problemlerinin çözümünde kullanılan algoritmaların sınıflandırılması gösterilmektedir.



Şekil 3.3 Optimizasyon algoritmalarının sınıflandırılması, (Yang, 2010).

Tez kapsamında ele alınan yöntemler olasılıksal algoritmalar sınıfında değerlendirildiğinden stokastik algoritmalar alt başlığı kısaca açıklanmıştır.

Stokastik algoritmalar genellikle aralarında küçük farklar olan sezgisel (heuristic) ve metasezgisel (metaheuristic) olarak iki sınıfta incelenir. Sezgisel (heuristic) kavramı deneme yanılma yoluyla çözümün bulunması olarak ifade edilebilir. Optimizasyon problemlerinde deterministik algoritmalara nazaran çok daha makul zamanlarda kaliteli çözümleri bulabilen sezgisel algoritmalar, her zaman optimal çözümü bulma garantisi verememektedirler.

3.2. Sezgisel Optimizasyon

Sezgisel optimizasyon, herhangi bir problemin optimizasyonunda alternatif çözüm yollarından en optimum sonuçlara ulaşabilmek için doğal fenomenlerden ilham alan optimizasyon teknikleridir. Sezgisel optimizasyon çözümünde kullanılan algoritmalar, yakınsama özelliğine sahip olup, her zaman optimum çözümü garanti edememektedirler. Yakınsama özelliğinden dolayı optimum sonuca yakın sonuçlar üretebilmektedirler.

Sezgisel algoritmaların değerlendirilmesinde kullanılan kriterler aşağıdaki gibi olmalıdır (Karaboğa, 2011):

- **Çözüm Kalitesi ve Hesaplama Zamanı:** Çözüm kalitesi ve hesaplama zamanı bir algoritmanın etkinliğinin değerlendirilmesi için önemli kriterlerdir. Bundan dolayı bir algoritma ayarlanabilir parametreler setine sahip olmalı ve bu parametreler kullanıcıya etkinlik açısında hesaplama maliyeti ile çözüm kalitesi arasında bir vurgulamanın yapılabilmesine imkân vermelidir. Diğer bir deyişle çözüm kalitesi ile hesap zamanı arasındaki ilişki kontrol edilebilmelidir.
- **Algoritma Basitliği ve Gerçeklenebilirlik:** Algoritma prensipleri basit olmalı ve genel olarak uygulanabilir olmalıdır. Bu durum problem yapısı ile ilgili başlangıçta çok az bilgiye sahip olunması halinde bile algoritmanın yeni alanlara kolaylıkla uygulanabilmesini sağlar.
- **Esneklik**: Algoritmalar modelde, sınırlamalarda ve amaç fonksiyonlarında yapılacak değişiklikleri kolayca karşılayabilmelidir.
- **Dinçlik**: Yöntem başlangıç çözümünün seçimine sahip olmaksızın her zaman yüksek kaliteli, kabul edilebilir çözümleri üretebilme kabiliyetine sahip olmalıdır.
- **Basitlik ve Analiz Edilebilirlik:** Karmaşık algoritmalar, esneklik ve çözüm kalitesi açısından basit algoritmalardan daha zor analiz edilebilmektedir. Algoritma kolayca analiz edilebilir olmalıdır.
- **Etkileşimli Hesaplama ve Teknoloji Değişimleri:** Algoritma içinde insan-makine etkileşimini kullanma fikri çoğu sistemde yaygın olarak gerçekleştirilmektedir. Herkesçe bilindiği gibi iyi bir kullanıcı arayüzü herhangi bir bilgisayar sistemini veya algoritmayı daha çekici

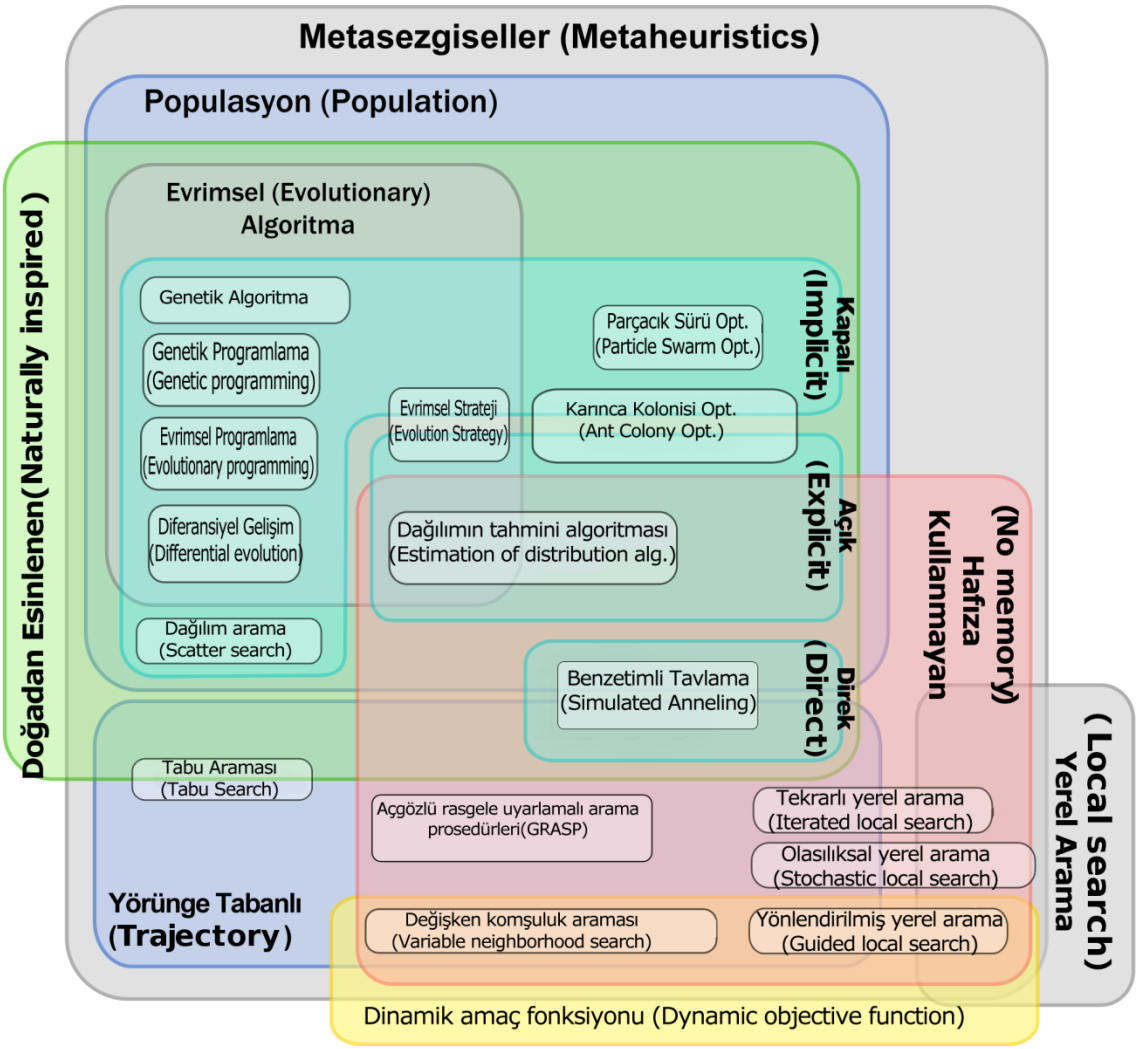
yapmaktadır. Bunun en önemli avantajı çözümleri grafiksel olarak sergilenebilmesidir.

Sezgisel yöntemlerin en önemlileri, GA, benzetimli tavlama (Simulated Annealing), Tabu Arama (Tabu Search), Yapay Sinir Ağları (Neural Networks) ve KKO algoritmalarıdır.

3.3. Metasezgisel Algoritmalar

Metasezgisel algoritmalar, sezgisel optimizasyon algoritmalarının gelişmişi olarak ifade edilir. Temel sezgisel yöntemlerin bir arada kullanılmasıyla ortaya çıkmışlardır. Yüksek seviye anlamında kullanılan meta ifadesi ile bu algoritmalar basit sezgisel algoritmalardan daha iyi performans sergilemektedirler. Ayrıca tüm metasezgisel algoritmalar, rastgelelik ve yerel aramayı değişimli olarak kullanmaktadırlar. Rastgelelik, optimizasyon algoritmalarının çözümlerini yerel aramalardan kurtararak küresel ölçekli aramalar yaparak iyileştirmeyi sağlamaktadırlar (Yang, 2010). Şekil 3.4' te metasezgisel algoritmaların sınıflandırılması gösterilmektedir.

Çoğu sezgisel algoritmalar probleme bağımlı algoritmalardır. Bir problem için en iyi performansı gösterirken diğer bir problem için aynı şekilde başarılı olmayabilir. Metasezgisellerin tüm problemleri kapsayıcı olması konusunda yeni yöntemler geliştirilmektedir. Bu yaklaşımlar sosyal, biyoloji, zooloji, fizik, bilgisayar ve karar verme gibi bilimler temel alınarak türetilmiştir. Bundan dolayı bu tür yaklaşımlara *modern sezgisel yaklaşımlar* ya da *yapay zekâ yaklaşımları* adı verilmektedir (Karaboğa, 2004).



Şekil 3.4 Metasezgisel algoritmaların cinslerine göre sınıflandırılması (Wikipedia, 2014)

Metasezgisel algoritmaların karakteristik özellikleri aşağıdaki gibi sıralanabilir (Blum ve Roli, 2003):

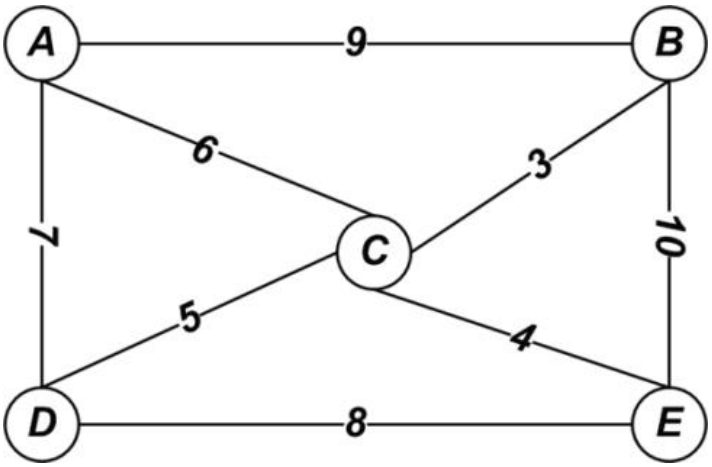
- Metasezgiseller arama süreçlerine rehberlik eden stratejilerdir.
- Arama uzayındaki keşiflerle en iyi ya da ona en yakın çözümlerin bulunması amacını taşır.
- Metasezgisel algoritmaları oluşturan teknikler, basit yerel arama prosedüründen karmaşık öğrenim süreçleri aralığında bir değişim gösterir.
- Metasezgisel algoritmalar tahmine dayalı rastgelelik içerirler.
- Bulunan bir çözüme takılıp kalmayı önleyici mekanizmalar barındırırlar.
- Metasezgiseller probleme özgü değildir.

- Farklı yöntemlerin kullanılmasıyla arama uzayını keşfeden yüksek seviyeli stratejilerdir.

Bu tez çalışması kapsamında ele alınan KKO ve PSO optimizasyon algoritmaları Şekil 3.4' te görüldüğü gibi, doğa esinli-popülasyon tabanlı metasezgiseller sınıfında değerlendirilmektedir.

3.4. Gezgin Satıcı Problemi (GSP)

GSP, ayrık optimizasyon tekniklerinin çözümünde kullanılan tümleşik (*combinatorial*) benchmark fonksiyonlarından biridir. GSP, bilgisayar bilimlerinde, verilerin modellenmesinde kullanılan Hamilton döngüsü olarak da ifade edilmektedir. Graf teorisinde kullanılan yol biçimi belirlemesi olarak ifade edilen Hamilton döngüsü, her düğüme bir kez uğrayarak en kısa kapalı yolun bulunmasıdır. GSP, gerçek dünya problemi olup, belirli N adet şehir (düğüm) için, her şehre bir kez uğramak şartı ile seyyar satıcının minimum maliyetle şehirleri seyahat ederek başlangıç noktasına geri dönerek izlenen rotanın bulunması olarak tanımlanmaktadır. GSP, tanım gereği kolay bir problem gibi gözükmekte olup literatürde NP-Zor problemler sınıfında değerlendirilmektedir. Şekil 3.5' te 5 düğüm ve aralarındaki uzaklıkları belirtilen bir Hamilton döngüsünde bulunan en kısa tam tur toplam 31 birim maliyetle A-B-C-E-D-A şeklindedir.



Şekil 3.5 Örnek bir graf

GSP ler iki ayrı normda incelenebilir:

Öklitsel GSP (Euclidean TSP):

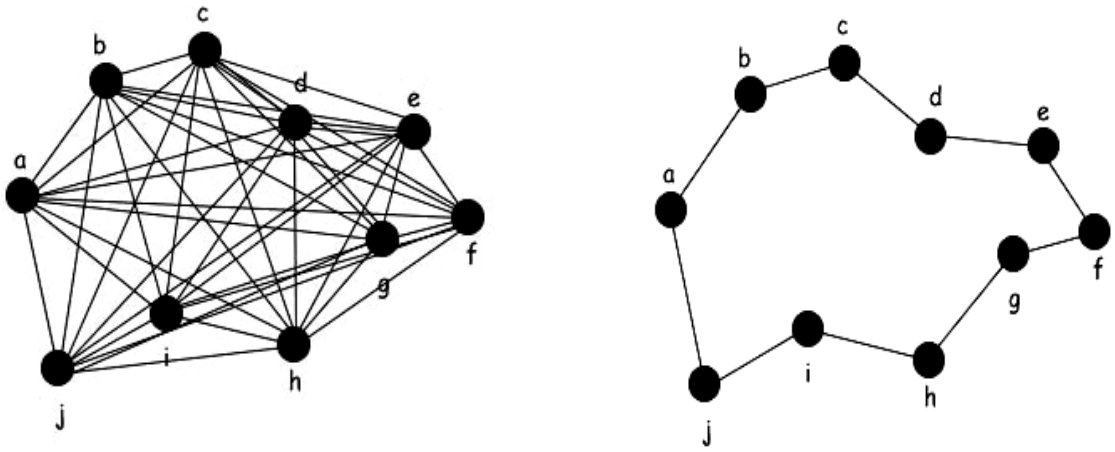
İki nokta arasındaki uzaklığın hesabı için Öklid kullanılması durumunda, i (x_i, y_i) ve j (x_j, y_j) şehirleri arasındaki Öklid mesafesi $d_{ij} = \left[\left(x_i - x_j \right)^2 + \left(y_i - y_j \right)^2 \right]^{1/2}$ ile bulunur.

Öklitsel GSP ise iki farklı alt başlıkta incelenmektedir:

- ***Simetrik GSP (SGSP)*** ; i ve j iki şehir olmak üzere iki şehir arasındaki karşılıklı uzaklıkların birbirine her zaman eşit olduğu GSP lerdir. Problemdeki tüm şehirler için $d_{ij} = d_{ji}$ eşitliği her zaman söz konusudur.
- ***Asimetrik GSP (AGSP)*** **;** i ve j iki şehir olmak üzere iki şehir arasındaki karşılıklı uzaklıkların birbirine her zaman eşit olmadığı GSP lerdir. Problemdeki tüm şehirler için $d_{ij} = d_{ji}$ eşitliği her zaman söz konusu değildir. Yani en az iki şehri birbirine bağlayan farklı yollar söz konusudur. Örneğin Konya' dan İstanbul'a gidecek bir gezgin satıcı, giderken kuzey hattını (Ankara-Bolu üzeri) kullanırken, dönüşte güney hattından (Eskişehir üzeri) Konya'ya gelebilir.

Öklitsel olmayan GSP (non-Euclidean TSP): İki nokta arasındaki uzaklığın Öklid bağlantısı dışında farklı matematiksel formüllerle hesaplanması ile bulunur. Örneğin bir küre yüzeyindeki iki nokta arasındaki mesafe öklit bağlantısı ile hesaplanamamaktadır (Arora, 1998 ve Uğur, 2008).

SGSP lerde olası tur sayıları *(n-1)!/2*, AGSP lerde ise *(n-1)!* adettir. GSP de gezilecek şehir sayısı arttıkça olası turların sayısı oldukça artacağından kesin çözüm sunan yöntemlerin bu problemleri çözme süreleri çok uzun olacaktır. Örneğin Şekil 3.6' daki GSP düğüm sayısı fazla olduğundan Şekil 3.5' teki problemden daha uzun sürede hesaplanabilecektir. Metasezgisel algoritmaların GSP çözümlerinde kullanılmasıyla hesaplama süreleri makul seviyelerde olmaktadır.



Şekil 3.6 10 düğümlü bir GSP örneği ve çözümü

3.5. Sürü Zekâsı

Sürü (Swarm), birbirleriyle etkileşim halinde bulunan dağınık yapıya sahip, bireylerin oluşturduğu yapılara denir. Canlılar topluluğunu oluşturan arılar, karıncalar, koyunlar, kuşlar vb. canlılar bu sürülere örnek gösterilebilir.

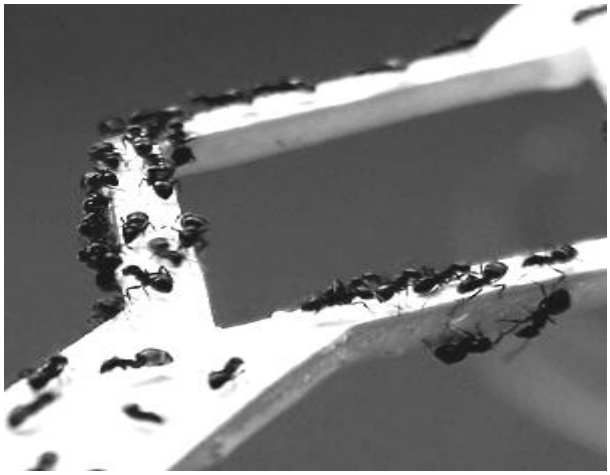
Zekâ, insanların öğrenme, anlama, soyut düşünme, sebeplendirme, planlama, problem çözme ve kanıya varma gibi zihinsel yeteneklerine verilen genel bir ifadedir.

Sürü zekâsı ise karıncalar ve diğer böcek kolonileri gibi deneyimsiz ajanların, kolektif davranışları sayesinde çevreleri ile etkileşim halinde gösterdikleri yem bulma, taşıma işinde yardımlaşma, kümelenme gibi davranışlarından esinlenerek bir zekâ geliştirmeleridir.

Karıncaların incelenmesinde sosyal bir yapı içerisinde kendi vücutlarından kurdukları köprülerle geniş aralıkları kolayca aşabildikleri gözlemlenmiştir (Şekil 3.7). Tek başlarına bir başarı gösterememelerine rağmen, sürü halinde hareket etmeleri sonucu zekice davranışlar sergileyebilmektedirler. Topluluktaki bireyler, en iyi bireyin ya da diğer bireylerin davranışından faydalanarak yorum yapmakta, gerektiğinde davranışını değiştirmekte, yuvadaki diğer bireylerle etkileşime devam etmekte ve bu edindikleri deneyimleri olası durumlarda çözüm yeteneği olarak ortaya koyabilmektedirler. Karıncaların ve diğer böcek kolonilerinin bu davranış biçimleri gözlemlenerek sürü zekâsı tabanlı doğa esinli metasezgisel optimizasyon algoritmaları geliştirilmiştir.







Şekil 3.7 Karıncaların oluşturduğu köprüye ait örnekler

Klasik yapay zekâ tekniklerindeki modelleme odaklı, planlı yaklaşımların aksine sürü zekâsı, basit olması, özerk yapısı, önceden planlama olmadan sürü halinde gösterdikleri bu davranışları sayesinde karmaşık problemlerin çözümlerinde başarılı olduklarını göstermiştir.

3.6. Karınca Optimizasyon Yöntemleri

3.6.1. Tanım

KKO algoritması, graflar üzerindeki olası yolları araştırarak hesaplama problemlerinin çözümü için kullanılan olasılıksal bir tekniktir. Gerçek karıncaların beslenme davranışlarından ilham alan bir yöntemdir. İlk olarak 1992 yılında Marco Dorigo tarafından doktora tezi olarak önerilmiştir (Dorigo, 1992). Orijinalinde amacı bir

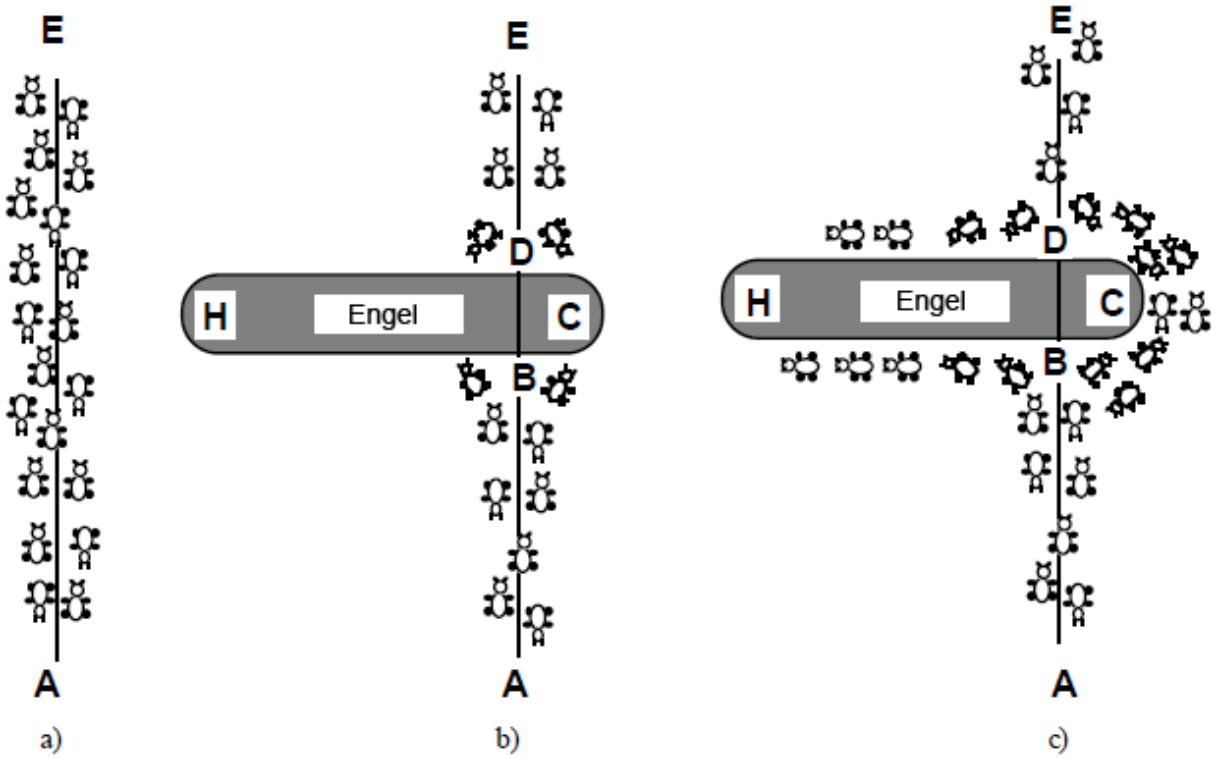
graf üzerinde ulaşılabilecek optimal yolları araştırmak üzere önerilmiş olan algoritma temel olarak şu özeliklere sahiptir:

- ✓ Çok-yönlü bir algoritmadır. Aynı problemin farklı türevlerine uygulanabilir. Örneğin gezgin satıcı probleminden asimetrik gezgin satıcı problemine basit bir uzantısı vardır.
- ✓ Güçlü bir algoritmadır. Sadece ufak değişikliklerle diğer kombinasyonel optimizasyon problemlerine uygulanabilir. İkinci dereceden atama problemi (QAP) ve makine yerleştirme problemi (JSP) gibi.
- ✓ Popülâsyon-temelli bir yaklaşımdır. Bir arama mekanizması olarak olumlu geri-dönüşlerin kullanılmasına izin verilmesi bakımından bu ilginç bir özelliktir.

İstenen bu özellikler özelleştirilmiş algoritmalarla daha iyi yapılabilen karınca sistemi (Ant System) sayesinde başarılabilir. Arama etkinliği "*karıncalar"* olarak isimlendirilen ajanlar tarafından yapılır. Bu ajanlar bazı ilavelerle beraber gerçek karıncaların basit temel davranışlarını taklit eder. Etolojistler tarafından çalışılan problemlerden biri karıncalar gibi hemen hemen kör olan hayvanların kendi kolonileri ve besin kaynağı arasında nasıl en kısa yolu belirleyip geri döndüklerinin anlaşılmasıdır. Araştırmacılar feromon izlerinden oluşan ortamın izlenen yollarla ilgili olarak bireyler arasında bilgiyi iletmek ve nereye gidileceğine karar vermekte kullanıldığını buldular. Hareket eden bir karınca izlediği yola bir miktar (değişken miktarda) feromon maddesi bırakır. Böylece bu karınca tarafından izlenen yol, bu madde ile işaretlenmiş olur. Diğerlerinden ayrı tutulan bir karınca temelde rastgele bir şekilde hareket ederken, önceden bırakılmış bir izle karşılaşan bir karınca ise bunu fark eder ve bu izi büyük olasılıkla takip etme yönünde karar verir. Böylece bu izi kendi feromonu ile de takviye etmiş olur. Ortaya çıkan kolektif davranış bir otokatalitik davranış biçimidir. Yani ne kadar çok karınca bir izi takip ederse, bu iz takip etmek için o kadar çok çekici hale gelir. Bu işlem pozitif geri-besle döngüsü olarak nitelenir. Bir karıncanın kullandığı yolun seçilme olasılığı aynı yolu daha önce seçmiş karıncaların sayısıyla orantılıdır. (Dorigo,2004)

Şekil 3.8'de gösterilen deney düzeneği durumu özetlemektedir. Burada bir yol boyunca yürüyen karıncalar vardır (örnek olarak besin kaynağı A noktasından yuvaya E noktasına doğru ve tersine doğru olan hareket, Şekil 3.8.a). Aniden bir engel koyulsun ve yol kesilsin. Bu durumda B konumunda A'dan E'ye yürüyen karıncalar (veya D

konumunda E'den A'ya yürüyen karıncalar) sağa mı sola mı döneceklerine karar vermek zorundadırlar (Şekil 3.8.b). Seçim önde gelen karıncaların bıraktığı feromon izlerinin yoğunluğundan etkilenir. Sağ yol üzerinde daha yüksek seviyedeki feromon bir karıncaya daha yüksek bir uyarıcı etki yapar ve böylece daha yüksek olasılıkla sağ yol tercih edilir. B noktasına (veya D noktasına) ulaşan ilk karınca, iki alternatif yol için daha önce hiç feromon maddesi olmadığından, sağa veya sola dönmek için eşit olasılığa sahiptir. BCD yolu daha kısa olduğu için bu yolu takip eden ilk karınca, BHD yolunu takip eden ilk karıncadan daha önce D noktasına ulaşır (Şekil 3.8.c). E'den D'ye dönen bir karınca DCB yolu üzerinde daha güçlü bir iz bulacaktır. Çünkü karıncaların yarısı tesadüfî olarak engele DCBA yolu üzerinden gitmeye karar verir ve BCD yolundan gelenler zaten gidecekleri yere daha önceden ulaşmış olurlar. Bu yüzden DCB yolu DH yoluna tercih edilecektir. Sonuç olarak, birim zaman BCD yolunu takip eden karıncaların sayısı BHD yolunu takip eden karıncaların sayısından daha yüksektir. Bu, kısa yoldaki feromon miktarının uzun yoldakinden daha hızlı şekilde artmasına neden olur. Böylece herhangi bir karınca tarafından takip edilecek yolun seçilme olasılığı hızlı bir şekilde daha kısa olan için biaslanmış olur.



Şekil 3.8 Gerçek karıncalar ile bir örnek

Sonraki bölümde tanımlanacak algoritmalar gerçek karınca kolonilerinin çalışmasından türetilmiş modellerdir. Bu yüzden bu sistem *Karınca Sistemi (Ant System)* olarak isimlendirilir ve algoritmalar *karınca algoritmaları (ant algorithms)* olarak tanıtılır. Burada karınca kolonilerinin benzetimi ile değil yapay karınca kolonilerinin bir optimizasyon aracı olarak kullanılmasıyla ilgilenilmektedir. Önerilen sistemin gerçeği ile bazı önemli farklılıkları vardır:

- ✓ Yapay karıncalar bir miktar hafızaya sahiptir,
- ✓ Bunlar tamamen kör değildirler,
- ✓ Ayrık zamanlı bir çevrede yaşarlar.

Aşağıda zamanla geliştirilen ve literatürde sıklıkla rastlanılan karınca optimizasyon yöntemleri açıklanmıştır.

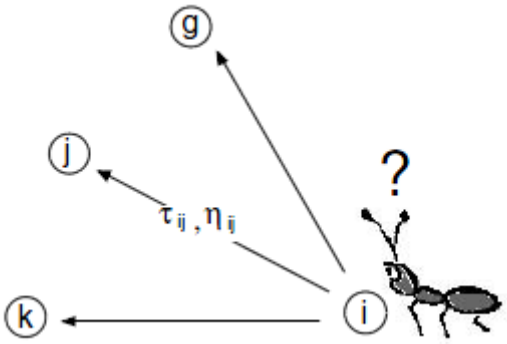
3.6.2. Karınca Sistemi (Ant System)

KKO algoritmalarının ilki olan Karınca Sistemi (AS), ilk olarak GSP üzerinde test edilmiştir (Dorigo, 1992). GSP, verilen *n* tane şehir için her birinin bir kez ziyaret edilerek oluşturulacak en kısa kapalı yolun belirlenmesi olarak açıklanabilir. *i* ve *j* şehirleri arasındaki yolun uzunluğu d_{ij} olarak ifade edilir. Uzunluk hesabı için Öklid kullanılması durumunda, *i* ve *j* şehirleri arasındaki Öklid mesafesi $d_{ij} = \left[\left(x_i - x_j \right)^2 + \left(y_i - y_j \right)^2 \right]^{1/2}$ ile bulunur. GSP bir (*N*,*E*) grafı ile gösterilebilir. Burada *N* şehir kümesi, *E* ise şehirlerarasındaki kenarlar kümesidir.

$b_i(t) = (i = 1,...,n)$ bir *t* anında *i* şehrindeki karıncaların sayısı ve $m = \sum_{i=1}^{n} b_i(t)$ toplam karınca sayısıdır. Her bir karınca aşağıdaki özelliklere sahip basit bir ajandır:

- ✓ Şehrin uzaklığına ve kenarlar üzerindeki mevcut iz miktarına bağlı bir fonksiyondan elde edilen bir olasılıkla gidilecek şehri seçer. (Şekil 3.9)
- ✓ Karıncaların legal turlar yapmasını kuvvetlendirmek için, önceden ziyaret edilmiş şehirlere tur tamamlanıncaya kadar izin verilmez (bu bir tabu listesi ile kontrol edilir).

- ✓ Bir tur tamamlandığında, ziyaret edilen her bir (i,j) kenarı üzerinde trail (feromon-iz) denilen bir madde bırakır.



Şekil 3.9 Karıncanın şehir seçimi - τ_{ij} *ve* η_{ij} *parametreleri*

$\tau_{ij}(t)$ bir *t* anında kenarda bırakılan iz yoğunluğudur. Her bir *t* anındaki karınca *t+1* anında olacağı şehri seçer. Bu yüzden AS algoritmasının bir iterasyonunu ifade edersek, (*t*, *t+1*) zaman aralığında *m* tane karınca tarafından *m* adet hareket yürütülür. Algoritmanın her bir *n* iterasyonundan sonra (buna bir döngü denir) her bir karınca bir turu tamamlamış olur. Bu noktada feromon yoğunluğu aşağıdaki formüle göre güncellenir.

$$\tau_{ij}(t+n) = \rho.\tau_{ij}(t) + \Delta\tau_{ij} \qquad (3.5)$$

Burada ρ bir katsayıdır. Öyle ki $(1-\rho)$ *t* ve *t+n* zaman aralığında feromon maddesindeki buharlaşma oranını gösterir. Buharlaşma sayesinde karıncaların bulduğu kötü çözümlerin unutulması sağlanmakla birlikte, iyi bulunan çözümlerde ise fazla feromon birikmesi önlenerek, karıncaların yeni turları aramasına fırsat sağlanmaktadır.

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad (3.6)$$

Burada $\Delta\tau_{ij}$ t ve $t+n$ zaman aralığında k. karınca tarafından (i,j) kenarına bırakılan iz maddesinin her bir birim için büyüklük miktarıdır (gerçek karıncalardaki feromon maddesi). Şu şekilde verilir:

$$\Delta\tau_{ij} = \begin{cases} \dfrac{Q}{L_k} & \text{Eğer k. karınca (i,j) kenarını kendi turunda kullanırsa (t ve t+n zaman aralığı)} \\ 0 & \text{Aksi halde} \end{cases} \qquad (3.7)$$

Burada Q bir sabittir ve L_k k. karıncanın tur uzunluğudur. ρ katsayısı sınırsız feromon biriktirmeyi önlemek için 1'den küçük bir değere ayarlanmalıdır. Tecrübeyle elde edilmiş bilgiye göre, 0 anındaki feromon miktarı küçük pozitif bir c sabitine ayarlanmalıdır.

Bir karıncanın tüm şehirleri ziyaret etme şartını yerine getirmek için, her bir karınca tabu listesi adı verilen bir veri yapıyla ilişkilendirilir. Tabu listesi t anına kadar ziyaret edilmiş olan şehirleri tutar ve n adet iterasyon tamamlanmadan önce bu şehirlerin tekrar ziyaret edilmesini yasaklar. Bir tur tamamlandığında, karıncanın mevcut çözümünü hesaplamak için kullanılır (ör. karınca tarafından izlenen yolun uzunluğunu hesaplamak için). Daha sonra tabu listesi boşaltılır ve karınca seçim yapmakta özgür kalır. **$\mathbf{tabu_k(s)}$** k. karınca tarafından ziyaret edilmiş s. şehri gösterir.

Görünürlük denilen η_{ij} değeri $1/d_{ij}$ oranıdır. Eşitlik (3.5)'e göre değişen feromon maddesi miktarının aksine, bu sayı karınca sisteminin çalışması boyunca değişmez. k. karınca için i şehrinden j şehrine geçiş olasılığı şu şekilde tanımlanır:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)^{\alpha} . \left[\eta_{ij}\right]^{\beta}\right]}{\sum\left[\tau_{ik}(t)\right]^{\alpha} . \left[\eta_{ik}\right]^{\beta}} & \text{if j} \in \text{izinverilen}_k \end{cases} \qquad (3.8)$$

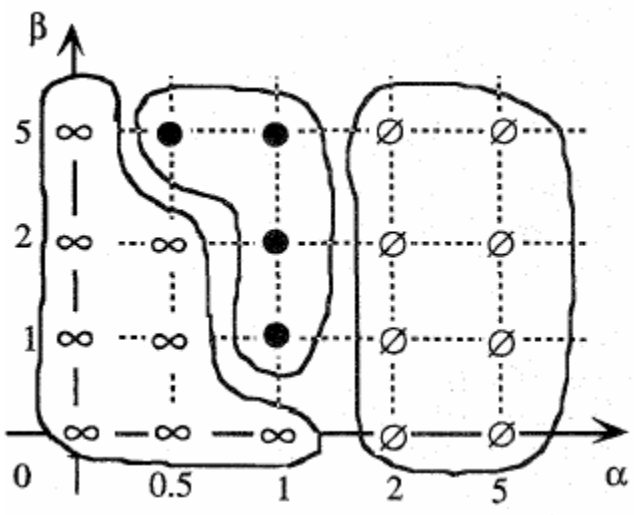
Burada izinverilen_k = { N – tabu_k } dır. Eşitlik 3.8, geçiş kuralı (transition rule) olarak bilinir ve k karıncası için i şehrinden j' ye t' ninci turunu yaparken j' yi seçme ihtimali, başka bir deyişle i ve j şehirleri arasındaki yolun seçilme ihtimalinin hesabıdır.

α ve β görünebilirliğe karşı ilişkisel olarak feromonun önemini kontrol eden parametrelerdir. Bu yüzden geçiş olasılığı görünebilirlik ve t anındaki iz yoğunluğu arasında ince çizgidir.

α ve β parametrelerinin farklı değerlerinde algoritmanın bulduğu sonuçlar arasında farklılıklar bulunmaktadır. Bunlar şu şekilde ifade edilebilir:

- ✓ α değerinin çok yüksek değerleri alması durumunda (feromon izine verilen değerin fazla olması) algoritma çok iyi çözümler bulamaz ve durgunluk (stagnation) davranışı gözlemlenir. Şekil 3.10' da Ø ile gösterilmektedir.
- ✓ α değerinin çok düşük değerleri alması durumunda (feromon izine verilen değerin az olması) algoritma durgunluk (stagnation) davranışı göstermez ancak yine de çok iyi çözümler bulamaz. Şekil 3.10' da ∞ ile gösterilmektedir.
- ✓ α ve β değerlerinin uygun aralıkları ise • sembolü ile gösterilmektedir. Şekil 3.10' da belirtilen bu uygun aralıktaki parametre değerlerinde algoritma aynı performansı sergilemektedir.

α = 0 olması durumu feromon izinin hesaba katılmaması anlamına gelir. Dolayısıyla olasılık hesaplamasında en yakın şehirlerin seçilmesi şansı çok yüksek olur. Yalnızca görünürlüğe göre seçim yaptığından algoritma stokastik açgözlü arama (greeedy search) algoritmasına dönüşür. β = 0 olması durumu ise sadece feromon izine göre bir tahmin yürütmek demek olup, sezgiselliğin ortadan kalkarak yetersiz sonuçlar elde edilmesini sağlar.



Şekil 3.10 α ve β parametrelerinin farklı değerlerinde karınca-döngü davranışı

Her bir karınca için Eşitlik 3.8' e göre karıncanın önceden gitmediği (tabu listesinde bulunmayan) şehirlerin hepsi için hesaplanan olasılıksal değerlere göre Rulet Tekerleği (Roulette Wheel) seçim mekanizması ile bir sonraki aşamada gidilecek şehir belirlenir. Rulet Tekerleği (Roulette Wheel) kullanılmadan olasılığı en yüksek olan şehirlerin seçilmesinde algoritma, optimum sonuçları bulamamaktadır.

Rulet Tekerleği seçim mekanizmasına göre çember, şehirlerin olasılık karar değerleri ile orantılı aralıklara bölünür. Burada aralığın genişliği, temsil ettiği şehrin seçilme olasılığını göstermektedir. Çemberdeki aralıkların toplamı ise, karıncanın daha önce uğramadığı tüm şehirlerin olasılık karar değerleri toplamına eşittir. Seçim aşamasında çember çevrilir ve sıfır ile toplam olasılık karar değeri aralığında rastgele bir sayı üretilir. Üretilen sayının düştüğü aralıktaki şehir, karıncanın bir sonraki adımda uğrayacağı şehir olarak belirlenir.

Karınca algoritmasındaki diğer önemli parametreler ise karınca sayısı ve tur sayısıdır. Karınca sayısının fazla olması algoritmanın yerel minimumlara takılmasına, az olması ise algoritmanın yeterince etkin sonuçlar üretmemesine neden olmaktadır. Bundan dolayı karınca sayısının şehir sayısına eşit seçilmesi uygun çözümler üretilmesini sağlamaktadır. Tur sayısı ise fazla olması durumunda daha iyi sonuçların elde edilmesine ancak buna karşın algoritmanın çalışma süresinin artmasına neden olmaktadır. Tur sayısı az olduğunda süre kısa olmakla birlikte sonuçlar iyi olmayabilir.

3.6.3. Elit Karınca Sistemi (Elitist Ant System)

Başlangıçtaki AS yapısına eklenen ilk gelişme, Karınca sistemi için Elitist Strateji olarak adlandırılan sistemdir. Elitist Karınca Sistemi Dorigo tarafından 1992' de geliştirilmiştir. Elitist karınca sistemi modelinde en iyi tura ait kenarların feromon güncellemesi standardın üzerinde yapılarak, en iyi turu yapan karıncanın izlediği yol kuvvetli bir şekilde seçkinleştirilir. En iyi tur T^{bs} , ve bu en iyi tura eklenen feromon miktarı ise e/ C^{bs} dir. e sabit bir parametre, C^{bs} ise en iyi turun uzunluğunu göstermektedir.

3.6.4. Max-Min Karınca Sistemi (Max-Min Ant System)

Thomas Stützle ve Holger H. Hoos (1996, 2000) mevcut algoritmanın feromon miktarının belirli sınırlar arasında (üst sınır, alt sınır aralığında) kalmasını sağlayacak şekilde algoritmada değişiklik yapmışlardır. Bu algoritmaya sınırlamalardan dolayı Maksimum-Minimum Karınca Sistemi (MAX-MIN Ant System - MMAS) adını vermişlerdir.

MMKS'yi AS' den ayıran iki temel özellikten ilki, her iterasyonda sadece bir önceki iterasyondaki en iyi çözümü veren karıncanın feromon yenilemesine izin verilmesidir. İkinci özellik ise, optimum araştırmalarındaki dalgalanmanın önlenmesi için, feromon izlerinin sınırlı bir aralıkta [τ_{min}, τ_{max}] tutulmasıdır. Feromon izleri üst limitten başlatılarak başlangıçta yüksek oranda iyileşme sağlanır. MMKS' de alt ve üst limitler uygun seçilmediği takdirde tüm karıncalar aynı yoldan gitmekte ve bu da iyi bir çözümün bulunmasını engellemektedir. Alt ve üst sınırlar Eşitlik 3.9 ve Eşitlik 3.10 daki formüllerle hesaplanır (Keskintürk ve Söyler, 2006). MMKS algoritmasında ilk aşamalarda iterasyondaki en iyi karıncanın turlarının kullanılır. Diğer aşamalarda ise global en iyi karıncanın turlarının kullanılması artarak iyileştirmeler yapılmaktadır.

$$\tau_{max} = \frac{1}{\rho} x \frac{1}{L^b} \tag{3.9}$$

$$\tau_{min} = \frac{\tau_{max}}{2n} \tag{3.10}$$

L^b bulunan en iyi tur uzunluğunu, ρ, feromon buharlaşma katsayısını, n ise nokta sayısını vermektedir.

Alt ve üst sınırları belirlemek deneysel olarak yapılmakla birlikte problemin yapısına göre değişiklik gösterebilmektedir.

Bu noktada feromon yoğunluğu aşağıdaki formüle göre güncellenir:

$$\tau_{ij} \leftarrow \left[(1-\rho).\tau_{ij} + \Delta\tau_{ij}^{best} \right]_{\tau_{min}}^{\tau_{max}} \tag{3.11}$$

$$\Delta\tau_{ij}^{best} = \begin{cases} \dfrac{1}{L_{best}} & \text{Eğer (i,j) en iyi turda ise} \\ 0 & \text{Aksi halde} \end{cases} \qquad (3.12)$$

3.6.5. Rütbeye Dayalı Karınca Sistemi (Rank-Based Ant System)

Bullnheimer, Hartl ve Strauss tarafından (1999) elitist yaklaşımın gelişmişi olarak rütbe yada dereceye bağlı olarak artan miktarda feromon bırakıldığı, Rütbeye Dayalı Karınca Sistemi (Rank Based Ant System - AS_{rank}) geliştirilmiştir. Bu yapıda, her karınca rütbesine göre artan bir miktarda feromon bırakır. İlave olarak, elit karınca sisteminde olduğu gibi, o ana kadarki en iyi karınca her tekrarda daima en yüksek feromonu bırakır. Bu sistemde her turda bir rütbelendirme söz konusudur.

Her bir turda, tur uzunluklarına göre en iyi çözümü bulan belirli bir miktar karınca için, bu en iyi tur uzunluklarına ait kenarlara orantısal feromon bıraktırılarak karıncalar arasında başarı olarak örneğin %10 en iyi karınca böylece ödüllendirilir.

3.6.6. Karınca-Yoğunluk, Karınca-Çevrim ve Karınca - Nicelik (Ant Density, Ant Cycle and Ant Quantity

AS olarak bilinen yapıdan türetilmiş üç algoritma söz konusudur. Bu yapılar Ant Cycle (Karınca Çevrim), Ant Density (Karınca Yoğunluk) ve Ant Quantity (Karınca Nicelik) olarak isimlendirilmiştir. Sistemler arasındaki tek farklılık ise feromon izi bırakılırken ortaya çıkmaktadır.

Ant Cycle (Karınca Çevrim) modelinde, feromon izi bırakımı her bir karıncanın bir turunu tamamlaması sonrası, hafızalarında (tabu liste) tuttukları o tura ait rota üzerindeki yolların feromonu miktarı güncellenmektedir. *n* bir karıncanın bir iterasyonu (döngüsü) olmak üzere:

Denemeler sonucu bu model için en iyi parametre değerleri;

$\alpha = 1$, $\beta = 5$ ve $\rho = 0.50$

$$\Delta\tau_{ij}^{k}(t,t+n) = \begin{cases} \dfrac{Q}{L_k} & \text{Eğer k. karınca (i,j) kenarını kendi turunda kullanırsa (t ve t+n aralığında)} \\ 0 & \text{Aksi halde} \end{cases}$$

(3.13)

Ant Density (Karınca Yoğunluk) modelinde ise her karınca *i* noktasından *j* noktasına gittiğinde Q miktar feromon bırakır. Toplam tur uzunluğunun ya da *i – j* noktaları arasındaki mesafenin bu modelde önemi yoktur.

Denemeler sonucu bu model için en iyi parametre değerleri;

$\alpha = 1$, $\beta = 5$ ve $\rho = 0.99$

$$\Delta\tau_{ij}^{k}(t,t+1) = \begin{cases} Q & \text{Eğer k. karınca (i,j) kenarını kendi turunda kullanırsa (t,t+1 arasında)} \\ 0 & \text{Aksi halde} \end{cases}$$

(3.14)

Ant Quantity (Karınca Nicelik) modelinde ise her karınca *i* noktasından *j* noktasına gittiğinde Q / d_{ij} miktar feromon bırakır. *i – j* noktaları arasındaki mesafe bu modelde önem kazanmakta ancak toplam tur uzunluğunun bir önemi yoktur.

Denemeler sonucu bu model için en iyi parametre değerleri;

$\alpha = 1$, $\beta = 5$ ve $\rho = 0.99$

$$\Delta\tau_{ij}^{k}(t,t+1) = \begin{cases} \dfrac{Q}{d_{ij}} & \text{Eğer k. karınca (i,j) kenarını kendi turunda kullanırsa (t ve t+1 aralığında)} \\ 0 & \text{Aksi halde} \end{cases}$$

(3-15)

Yukarıda da belirtildiği gibi karınca-yoğunluk ve karınca-nicelik algoritmalarında karıncalar her bir şehir geçişinde, karınca-çevrim algoritmasında ise her bir karınca bir turunu (döngüsünü) tamamladıktan sonra feromon güncellemesi yapılmaktadır. Başka bir değişle karınca-yoğunluk ve karınca-nicelik algoritmaları *lokal bilgi* kullanırken, karınca-çevrim algoritması ise *global bilgi*yi kullanmaktadır.

Global bilgiyi kullanan karınca-çevrim algoritması test problemlerinde diğer iki algoritmaya nazaran daha iyi sonuçlar verdiğinden AS ile ilgili bilgilerde bu algoritma ele alınmaktadır.

karınca-çevrim algoritması biçimsel olarak aşağıdaki gibidir:

1. Başlat:
   t = 0 { t zaman sayacıdır. }
   NC = 0 { NC döngü sayacıdır. }

   for tüm (i,j) kenarları için bir başlangıç değeri ata
   $\tau_{ij}(t) = c$
   $\Delta\tau_{ij} = 0$
   m tane karıncayı n düğüm noktasına yerleştir.

2. s = 1
   for k=0 to m ***do***
   k. karıncanın başlama şehrini $tabu_k(s)$ e ata.

3. ***Repeat until*** tabu listesi doluncaya kadar
   s = s + 1
   for k=1 to m ***do***
   denklem(3-4)'te verilen olasılıkla seçilen bir j şehrine taşı.
   $tabu_k(s)$ e j şehrini ekle.

4. ***for*** k=1 to m ***do***
   k. karıncayı $tabu_k(n)$ den $tabu_k(1)$ e taşı.
   k. karınca tarafından belirlenen tur uzunluğu L_k yı hesapla.
   Bulunan en kısa yolu güncelle.

   for tüm (i,j) kenarları için ***do***
   for k=1 to m ***do***

$$\Delta\tau_{ij} = \begin{cases} \frac{Q}{L_k} & \text{Eğer k. karınca (i,j) kenarını kendi turunda kullanırsa (t ve t+n zaman aralığında)} \\ 0 & \text{Aksi halde} \end{cases}$$

$$\Delta\tau_{ij} = \Delta\tau_{ij} + \Delta\tau_{ij}^{k}$$

5. ***for*** tüm (i,j) kenarları için
   $\tau_{ij}(t+n) = \rho.\tau_{ij}(t) + \Delta\tau_{ij}$ denklemine göre $\tau_{ij}(t+n)$ değerini hesapla.
   t = t+1
   NC = NC + 1
   for tüm (i,j) kenarları için $\Delta\tau_{ij} = 0$ atamasını yap.

6. ***IF*** (NC < NC_{MAX}) ve (durgunluk davranışı değilse) ***THEN***
   Tüm tabu listelerini boşalt.
   Adım 2 ye git.
   ELSE
   En kısa yolu yaz.
   Dur.

3.6.7. Ant Q

Gambardella ve Dorigo tarafından (1995) Q-Learning algoritmasındaki öğrenme katsayısının (γ) katkısıyla feromon güncellemesinin hesaplandığı Ant-Q geliştirilmiştir. Karınca Koloni Sistemi ile aynı performanslarda sonuçlar üretmektedir. Bu yüzden çalışmalarda Ant-Q bırakılarak Karınca Koloni Sistemi üzerine yoğunlaşılmıştır.

Karınca Koloni Sistemi (ACS) ile Ant-Q arasındaki tek fark, karınca koloni sistemindeki τ_0 teriminin Ant-Q da $\tau_0 = \gamma \max_{j \in N_i^k} \{\tau_{ij}\}$ şeklinde tanımlanmasıdır.

3.6.8. Karınca Koloni Sistemi (Ant Colony System)

Karınca Koloni Sistemi (ACS- Ant Colony System) , 1996' da Dorigo ve Gambardella tarafından tanıtılmıştır. Sınırlı sayıda ve az şehirli GSP ler için makul bir sürede iyi çözümler sunabilen AS nin performansının geliştirilmesi amaçlanmıştır. Karınca sisteminden bazı noktalarda farklılık gösterir.

İlki, karınca sisteminden daha atılgan bir seçme kuralı uygulanmasıyla, karınca sistemindeki karıncaların araştırma birikimi olağanüstü bir başarıya dönüştürülmektedir. İkincisi, feromon buharlaşımı ve feromon bırakımı, sadece o ana kadarki en iyi tura ait olan arklarda gerçekleşir. Bir tur sonunda bütün karıncalar bir çözüm oluşturduklarında, feromon izi, tekrarın başlangıcından o yana en iyi turu bulan karıncanın kullandığı arklara ilave edilir. Üçüncüsü, karıncaların i şehrinden j şehrine hareket ederken kullandıkları bir (i, j) arkındaki feromonun bir kısmı, her defasında, alternatif yolların keşfini arttırmak için, azaltılır. Bu şekilde farklı lokal bir feromon güncelleme gerçekleştirilir. Dördüncüsü, ziyaret edilecek bir sonraki şehrin seçimini sınırlamak için aday listeleri kullanılır (Öztürk, 2006).

Karınca Koloni Sistemi, küçük GSP problemlerinde karınca sistemine nazaran iyi sonuçlar üretmese de, büyük GSP problemlerinde daha başarılı sonuçların üretilmesini sağlamaktadır.

- ✓ **Sözde–Rastsal Orantılı Seçim Kuralı (Pseudo-Random Proportional Rule):**

i sehrindeki k karıncası, j' ye geçişini aşağıdaki ifadedeki kurala göre seçimini yaparak gerçekleştirir:

$$p_{ij}^{k}(t)=\begin{cases} J, & aksi\ halde \\ \arg\max_{l\in N_i^k}\{\tau_{il}.\left[\eta_{ij}\right]^{\beta}\} & eğer\ \ q\le q_0\ ise \end{cases} \qquad (3\text{-}16)$$

$$J=\begin{cases} \dfrac{\left[\tau_{ij}(t)^{\alpha}.\left[\eta_{ij}\right]^{\beta}\right]}{\sum\left[\tau_{ik}(t)\right]^{\alpha}.\left[\eta_{ik}\right]^{\beta}} & \text{if j } \in \text{ izinverilen}_k \end{cases} \qquad (3\text{-}17)$$

q, [0-1] aralığında rastgele seçilen bir değişkendir. q_0 ise $(0\le q_0\le 1)$ aralığında seçilebilen ve genellikle 0.9 olarak alındığında en uygun çözümü veren bir parametredir. Eğer $(q\le q_0)$ ise yeni bir kurala göre (Eşitlik 3.16) seçim yapılmakta, aksi durumda mevcut karınca sistemindeki olasılıksal seçme kuralı α = 1 alınarak yapılmaktadır.

- ✓ **Global Feromon Güncelleme Kuralı:**

Global feromon güncelleme, her bir iterasyon sonunda o ana kadar en iyi tur uzunluğunu elde eden karıncanın turundaki rotadaki kenarlara ait feromon bırakılmasıdır. Feromon güncellemesi şu şekilde yapılmaktadır:

$$\tau_{ij} \leftarrow (1-\rho).\tau_{ij}+\rho.\Delta\tau_{ij}^{best}, \qquad \forall(i,j)\in T^{best} \qquad (3\text{-}18)$$

T^{best} en kısa yol turu, τ_{ij}^{best} en iyi turdaki ij kenarına ait feromon miktarını (τ_{ij}^{best} = $1/C^{best}$), $1/C^{best}$ ise en iyi yolun tur uzunluğudur. Karınca sisteminde bütün kenarlara uygulanan feromon izi güncellemesi, karınca koloni sisteminde sadece en iyi karıncaya uygulanmaktadır. Bırakılan feromon miktarı, karınca sisteminin aksine ρ katsayısı oranında azaltılır.

- ✓ **Lokal Feromon Güncelleme Kuralı:**

Lokal feromon güncelleme ile turların dinamik olarak değiştirilmesi ve bütün karıncaların bir önceki turdan kalan en iyi tura bağlı kalmadan yeni aramalar yapmasını sağlamaktadır. Global feromon güncellemesine ek olarak *ij* kenarını ziyaret eden karıncanın lokal feromon güncellemesi şu şekilde yapılmaktadır:

$$\tau_{ij} \leftarrow (1-\xi).\tau_{ij} + \xi.\tau_0 \qquad (3\text{-}19)$$

ξ, (0,1) aralığında olan deneysel sonuçları sonrası 0.1 değerinin uygun olduğu feromon buharlaştırma katsayısını, τ_0 ise başlangıç feromon miktarlarını göstermektedir. Deneysel olarak τ_0 için 1/ n.C^{nn} bulunmuştur. n, GSP örneğindeki şehir sayısı, C^{nn} ise, en yakın komşu tur uzunluğudur.

Lokal feromon güncellemesi, turları karıştırarak, yolların öğrenilen istekliliğinin dinamik bir şekilde değiştirilmesini sağlamaktadır. Bu şekilde ziyaret edilmeyen kenarlar varsa, bunların keşfedilme olasılığının artırılarak algoritmanın durgunluk göstermesi engellenir.

3.7. Parçacık Sürü Optimizasyonu

3.7.1. Tanım

Parçacık Sürü Optimizasyonu (Particle Swarm Optimization) (PSO) 1995'te Dr. Eberhart ve Dr. Kennedy tarafından (Kennedy ve Eberhart, 1995) geliştirilmiş popülasyon temelli sürü zekası kategorisinde değerlendirilen sezgisel bir optimizasyon tekniğidir. İnsanların konuşarak ya da bir başka şekilde bilgi paylaşımı yapmaları gibi, kuşlar ve balıklarında bilgi paylaşması sosyal bir zekâya işaret etmektedir. PSO, kuşların yön tayininde birbirlerinden faydalanmalarından ve balık sürülerinin sosyal davranışlarından esinlenilerek geliştirilmiştir. (Şekil 3.11)



Şekil 3.11 Balıklar ve kuşlar kolektif davranış sergilerler.

PSO, Genetik Algoritmalar gibi evrimsel hesaplama teknikleri ile birçok benzerlikler gösterir. Sistem rastgele çözümlerden oluşan bir popülasyonla başlatılır ve en iyi çözüm için jenerasyonları güncelleyerek arama yapar. Buna karşın, GA' nın tersine, PSO' da çaprazlama ve mutasyon gibi evrimsel operatörler yoktur. PSO' da parçacık (particles) denilen potansiyel çözümler, mevcut en iyi çözümleri takip ederek problem uzayında gezinirler (uçuş yaparlar).

GA ile karşılaştırılırsa; PSO' nun avantajı, gerçekleştirilmesinin kolay olmasıdır ve ayarlanması gereken çok az parametresi vardır. PSO pek çok alanda başarılı bir şekilde uygulanmıştır. Bunlardan bazıları; fonksiyon optimizasyonu, yapay sinir ağları eğitimi, bulanık sistem kontrolü ve GA' nın uygulanabildiği diğer alanlardır.

Biyolojik sistemlerden esinlenen birçok hesaplama tekniği mevcuttur. Örneğin yapay sinir ağları insan beyninin basitleştirilmiş bir modelidir. Genetik algoritmalar biyolojideki evrimsel süreçten esinlenir. Burada ise ele alınan konu biyolojik sistemlerin farklı bir türü olan sosyal sistemlerdir. Özellikle birbiriyle ve çevresiyle etkileşim içinde olan basit bireylerin birliktelik davranışları incelenmektedir. Bu kavram sürü zekâsı olarak isimlendirilir. Sürü zekâsı alanında parçacıklardan esinlenen metotlardan biri Karınca Kolonisi Optimizasyonu (KKO) iken bir diğeri ise Parçacık Sürü Optimizasyonudur (PSO) (Şekil 3.4).

Parçacık sürüsü kavramı basitleştirilmiş sosyal sistemin benzetimi olarak oluşturulmuştur. Asıl amaç, bir kuş veya balık sürüsü koreografisinin grafiksel olarak benzetiminin yapılmasıdır. Ancak sonraları parçacık sürüsü modelinin bir optimizasyon aracı olarak kullanılabileceği düşünülmüştür.

3.7.2. PSO Algoritması

Yukarıda bahsedildiği gibi, PSO Algoritması kuşların uçuşu veya balıkların sürü halindeki hareketinden esinlenerek ortaya atılan bir yöntemdir. Senaryo olarak bir alanda rastgele yiyecek arayan bir kuş grubunun olduğunu ve arama yapılan alanda yalnızca bir parça yiyecek olduğunu varsayalım. Kuşların hiçbiri yiyeceğin nerede olduğunu bilmesin. Fakat her bir iterasyon sonunda yiyeceğin ne kadar uzakta olduğunu bilsinler. Bu durumda en iyi strateji, yiyeceğe en yakın olan kuşu takip etmektir.

PSO bu senaryoya göre çalışır ve optimizasyon problemlerini çözmek için kullanılır. PSO' da her bir çözüm arama uzayındaki bir kuşu temsil eder ve bunlar birer "parçacık" olarak isimlendirilir. Tüm parçacıkların, optimize edilecek uygunluk fonksiyonu tarafından değerlendirilen bir uygunluk değeri ve uçuşlarını yönlendiren hız bilgileri vardır. Parçacıklar, problem uzayında mevcut optimum parçacıkları takip ederek uçarlar.

PSO, bir grup rastgele üretilmiş çözümle (parçacıkla) başlatılır (başlangıç popülasyonunun oluşturulması) ve jenerasyonlar güncellenerek en uygun değer araştırılır. Başlangıç popülasyonu, rastgele noktalara yerleştirilen ve rastgele hızlara sahip bir sürü olarak ifade edilir. Her iterasyondaki, her bir parçacık iki "*en iyi*" değere göre güncellenir.

- Bunlardan birincisi bir parçacığın o ana kadar bulduğu en iyi uygunluk değeridir. Ayrıca bu değer daha sonra kullanılmak üzere hafıza tutulur ve "*pbest"* yani parçacığın en iyi değeri olarak isimlendirilir.
- Diğer en iyi değer ise popülâsyondaki herhangi bir parçacık tarafından o ana kadar elde edilmiş en iyi uygunluk değerine sahip çözümdür. Bu değer popülasyon için global en iyi değerdir ve *"gbest"* olarak isimlendirilir.

$$x = \begin{pmatrix} a_{11} & \cdots & a_{1D} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nD} \end{pmatrix}_{nxD}$$

D adet parametreden oluşan n adet parçacık için popülasyon matrisi yukarıdaki gibi ifade edilir. Matrise göre, i. parçacık $x_i = [x_{i1}, x_{i2}, ..., x_{iD}]$ için bu değerler $pbest_i = [p_{i1}, p_{i2}, ..., p_{iD}]$ $gbest_i = [p_1, p_2, ..., p_D]$ şeklinde gösterilir. i. ninci parçacığın

her konumdaki değişim miktarını gösteren hız vektörü $v_i = [v_{i1}, v_{i2}, ..., v_{iD}]$ olarak ifade edilir. Bu iki en iyi değer bulunduktan sonra; parçacık, hızını ve konumu sırasıyla aşağıdaki Eşitlikler (3-20) ve (3-21)' e göre günceller.

Atalet Kişisel Hafıza Sosyal Hafıza

$$v_i^{k+1} = w.v_i^k + c_1.rand_1^k.\left(pbest_i^k - x_i^k\right) + c_2.rand_2^k.\left(gbest^k - x_i^k\right) \quad (3\text{-}20)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (3\text{-}21)$$

Burada ***rand*** (0,1) arasında üretilen rastgele bir değeri, ***i*** parçacık numarasını, ***k*** ise iterasyon sayısını gösterir. c_1 ve c_2 öğrenme faktörleridir. Bunlar parçacıkları ***pbest*** (personal best) ve ***gbest*** (global best) konumlarına doğru yönlendiren sabitlerdir. c_1 parçacığın kendi tecrübelerine göre (kişisel hafıza), c_2 ise sürüdeki diğer parçacıkların tecrübelerine göre (sosyal hafıza) hareketi yönlendirir. Düşük değerler seçilmesi parçacıkların hedef bölgeye doğru çekilmeden önce, bu bölgeden uzak yerlerde dolaşmalarına imkân verir. Ancak hedefe ulaşma süresi uzayabilir. Diğer yandan, yüksek değerler seçilmesi, hedefe ulaşmayı hızlandırırken, beklenmedik hareketlerin oluşmasına ve hedef bölgenin es geçilmesine sebep olabilir. Genellikle $c_1=c_2=2$ olarak almanın iyi sonuçlar verdiği belirtilmiştir. ***w*** değeri ise her bir iterasyonda değeri düşürülerek bir önceki hız bilgisinin etkisini azaltmaya yarayan atalet ağırlığı (inertia weight) dır.

Eşitlik 3.20' deki "Atalet" olarak belirtilen $w.v_i^k$ ifadesi hız değerinin atalet ağırlığını göstermektedir. Bu parametre parçacıkların hız değerlerinde ani değişme olmamasını, parçacıkların bir önceki hızlarına bağlı kalarak hız güncellemesi yapmasını sağlayarak çözüm uzayında parçacıkların ani hız değişimlerini engelleyerek uygun arama yapabilmesini sağlamaktadır. "Kişisel Hafıza" olarak belirtilen $c_1.rand_1^k.\left(pbest_i^k - x_i^k\right)$ ile parçacıkların c_1 ve rand() ile ölçeklendirilerek parçacıkların yeni hızlarının belirlenmesinde geçmişteki en iyi konuma (pbest) yakınsaması sağlanmaktadır. "Sosyal Hafıza" olarak belirtilen $c_2.rand_2^k.\left(gbest^k - x_i^k\right)$ kısmı ile ise bir öncekine benzer şekilde parçacıkların c_2 ve rand() ile ölçeklendirilerek parçacıkların

sürünün en iyisine (gbest) yakınsaması sağlanmaktadır. Görüldüğü gibi parçacığın mevcut konumundan bir sonraki konumuna geçişini sağlayan hızı hesaplanırken iterasyonlardaki her bir parçacığın kendi en iyi değeri ve popülasyonun en iyisinden faydalanılmaktadır.

PSO' nun avantajlarından birisi reel sayılarla çalışıyor olmasıdır. Genetik algoritmalardaki gibi hesaplama yapabilmek için ikili kodlamadan dönüştürme yapılması ya da bazı kullanılması zorunlu özel operatörlere ihtiyaç duyulmaz. Örneğin $f(x) = x_1^2 + x_2^2 + x_3^2$ fonksiyonu için çözümü bulmayı deneyelim. Burada 3 bilinmeyen olduğundan dolayı D=3 boyutludur ve parçacıklar (x_1, x_2, x_3) şeklinde ayarlanır. $f(x)$ fonksiyonu da uygunluk fonksiyonu olarak kullanılır. Daha sonra yukarıda verilen standart prosedür optimumu bulmak için uygulanır. Sonlanma kriteri olarak maksimum iterasyon sayısı veya minimum hata koşulu sağlanması kullanılır. Görüldüğü gibi PSO' da ihtiyaç duyulan çok az sayıda parametre vardır. Bu parametrelerin listesi ve tipik olarak aldıkları değerler aşağıda verilmektedir:

Parçacık Sayısı: genellikle 20 ila 40 arasında alınır. Aslında çoğu problem için sayıyı 10 almak iyi çözümler elde etmek için yeterlidir. Bazı zor veya özel problemler için 100 veya 200 parçacık kullanılması gerekebilir.

Parçacık boyutu: optimize edilecek probleme göre değişmektedir.

Parçacık değer aralığı (range): Optimize edilecek probleme göre değişmekle birlikte farklı boyutlarda ve aralıklarda parçacıklar tanımlanabilir.

Vmax: Bir iterasyonda, bir parçacıkta meydana gelecek maksimum değişikliği (hız) belirler. Genellikle parçacık aralığına göre belirlenir. Örneğin X1 parçacığı [-5,5] aralığında ise Vmax=10 sınırlandırılabilir.

Öğrenme Faktörleri: c_1 ve c_2 genellikle 2 olarak seçilir. Fakat farklı da seçilebilir. Genellikle c_1, c_2 ye eşit ve [0, 4] aralığında seçilir.

Durma Koşulu: Maksimum iterasyon sayısına ulaşıldığında veya değer fonksiyonu istenilen seviyeye ulaştığında algoritma durdurulabilir. Ayrıca belirtilen bir süre, belirli

iterasyonlar boyunca gbest çözümünün değişmemesi, belirli sayıda uygunluk fonksiyonunun hesaplanması ve belirtilen bir oranda minimum hata oranına göre gbest in elde edilmesi durdurma kriterleri olarak belirlenebilir.

Atalet Ağırlığı: Genellikle [0.9,0.4] aralığında alınarak, algoritmanın işleyişi süresince değişmeyerek kullanılabilen, ya da her bir iterasyonda azaltılarak dinamik olarak kullanılabilen parametredir.

Pozisyon Değerleri: Parçacıkların pozisyonları (alabileceği değerler) herhangi bir çözüm aralığında ise [x_{min}, x_{max}] bu sınırların dışına çıkacak yeni bir pozisyon bilgisi hesaplanırsa parçacığa verilecek bazı kısıtlamalar ile bu düzenlenir. Kısıtlamalar şu şekildedir:

$$\begin{aligned} &eğer\, x_i > x_{\max} \;\; ise \;\; x_i = x_{\max} \\ &eğer\, x_i < x_{\min} \;\; ise \;\; x_i = x_{\min} \end{aligned} \qquad (3\text{-}22)$$

Hız Değerleri: Hız değerleri pozitif ve negatif değerler alabilmekte, bu sayede çok yönlü hareketlilik sağlanmaktadır. Parçacıkların çözüm uzayı dışına çıkmasını engellemek hız değerlerinin kontrolü ile sağlanabilmektedir. Yukarıda da belirtildiği gibi parçacık aralığına göre belirlenen V_{max} ın yanı sıra V_{min} değeri de bulunmaktadır. Bunun için hız değerleri için sınırlar belirlenmiştir. Bu sınırlara ait kısıtlamalar şu şekildedir:

$$\begin{aligned} &V_{\min} = -V_{\max} \quad olmak\, üzere \\ &eğer\, V_i > V_{\max} \;\; ise \;\; V_i = V_{\max} \\ &eğer\, V_i < V_{\min} \;\; ise \;\; V_i = V_{\min} \end{aligned} \qquad (3\text{-}23)$$

Atalet Ağırlığı : Atalet ağırlığı parçacığın bir önceki hızının yeni hızına etkisini belirleyen bir parametredir. Bu parametre parçacıkların hız değerlerinde ani değişme olmamasını, parçacıkların bir önceki hızlarına bağlı kalarak hız güncellemesi yapmasını sağlayarak çözüm uzayında parçacıkların ani hız değişimlerini engelleyerek uygun arama yapabilmesini sağlamaktadır.

Atalet ağırlığının büyük değerler alması parçacığın çözün uzayında global aramalar yapmasına, küçük değerler alması ise çözüm uzayında bölgesel aramalar yapmasını sağlamaktadır.

Genel olarak uygulamalarda w_{max} = 0.9, w_{min} = 0.4 olarak alınarak her bir iterasyonda en büyük değerinden (başlangıç değeri) en küçük değerine (bitiş değeri) lineer bir şekilde azalması sağlanır.

$$w_i = w_{\max} - \frac{w_{\max} - w_{\min}}{toplam\,iterasyon}.k \tag{3-24}$$

$$k = mevcut\,iterasyon$$

PSO algoritması biçimsel olarak aşağıdaki gibidir:

BEGIN

Başlangıç parametrelerini ayarla.

For her parçacık için
    Parçacığı başlangıç konumuna getir
End

Do
  For her parçacık için
    Uygunluk değerini hesapla
    Eğer uygunluk değeri pbest ten daha iyi **ise,**
      Şimdiki değeri yeni pbest olarak ayarla (pbest güncellemesi)
  End

  Tüm parçacıkların bulduğu pbest değerlerinin en iyisini, tüm parçacıkların gbest'i olarak ayarla (gbest güncellemesi)

  For her parçacık için
    Eşitlik (3-20)'ye göre parçacık hızını hesapla
    Eşitlik (3-21)'e göre parçacık konumunu güncelle
  End

While maksimum iterasyon sayısına veya minimum hata koşulu sağlanana kadar devam et.

END

4. KOMŞULUK OPERATÖRLERİ İLE KKO-PSO TABANLI HİYERARŞİK YÖNTEM

Bu bölümde tez çalışması kapsamında önerilen hiyerarşik yöntem için gerekli argümanlar açıklanarak komşuluk operatörleri ile KKO-PSO tabanlı yeni hiyerarşik yaklaşımın detayları anlatılmıştır. Yöntemin performansı TSPLIB den alınan 10 farklı GSP üzerinde test edilmiştir.

4.1. Komşuluk Operatörleri

Komşuluk operatörleri, mevcut bir çözümden yeni bir çözüm sağlamak için kullanılır. 7 adet komşuluk operatörü bulunmaktadır (Szeto ve ark., 2011). Tez çalışmasında kullanılan komşuluk operatörleri, Araya Rastgele Ekleme (Random insertions - insert), Araya Rastgele Alt Küme Ekleme (Random insertions of subsequences - insert_sub), Alt Kümenin Ters Çevrilmesi (Reversing a subsequence - reverse), Ters Çevrilmiş Alt Kümenin Rastgele Araya Eklenmesi (Random insertions of reversed subsequences - reverse_insert), Ters Çevrilmiş Rastgele İki Alt Kümenin Yer Değiştirmesi (Random swaps of reversed subsequences - reverse_swap), Rastgele Yer Değiştirme (Random swaps - swap) ve Rastgele İki Alt Kümenin Yer Değiştirmesi (Random swaps of subsequences - swab_sub) dir. Aşağıda detaylarıyla belirtilen komşuluk operatörlerinden herhangi biri, karıncaların oluşturdukları rotaya (tabu liste) uygulanır ve yeni uygunluk değeri hesaplanır. Komşuluk operatörü sonucu oluşan tur uzunluğu, orijinal rotanın tur uzunluğundan daha iyi ise karıncanın o turu bu sonuçla değiştirilerek iyileşir. Komşuluk operatörlerinin KKO-PSO algoritmasında kullanılmasındaki temel amaç; bir süre sonra yerel minimumlara takılıp durgunluk hareketi göstererek aynı sonuçların tekrar ettiği KKO sonuçlarında, çeşitliliğin artırılarak çözümü global optimum sonuçlara taşıyabilmektir. Tez kapsamında KKO sonuçlarından örnek olarak 8 düğümlü bir tur uzunluğu üzerinde komşuluk operatörlerinin detayları verilmiştir.

- ✓ **Araya Rastgele Ekleme (Random insertions - insert)**

  Bu operatör rastgele seçilmiş herhangi bir i noktasının j. ci pozisyona eklenerek diğerlerinin sağa kaydırılması ile yapılmaktadır.

Önce	4	2	5	7	1	8	6	3

Sonra	4	2	3	5	7	1	8	6

- ✓ **Araya Rastgele Alt Küme Ekleme (Random insertions of subsequences - insert_sub)**

  Bu operatör *Araya Rastgele Ekleme* operatörünün uzantısı olup, herhangi bir bölümün j. ci pozisyona eklenerek diğerlerinin sağa kaydırılması ile yapılmaktadır.

Önce	4	2	5	7	1	8	6	3

Sonra	4	1	8	6	2	5	7	3

- ✓ **Alt Kümenin Ters Çevrilmesi (Reversing a subsequence - reverse)**

  Bu operatör rastgele uzunlukta seçilen bir alt parçanın kendi içinde terslenmesi ile yapılmaktadır.

Önce	4	2	5	7	1	8	6	3

Sonra	4	1	7	5	2	8	6	3

- ✓ **Ters Çevrilmiş Alt Kümenin Rastgele Araya Eklenmesi (Random insertions of reversed subsequences - reverse_insert)**

  Alt Kümenin Ters Çevrilmesi ve *Araya Rastgele Eklenmesi* operatörlerinin birleştirilmesi ile yapılmaktadır.

Önce	4	2	5	7	1	8	6	3

Sonra	4	6	8	1	2	5	7	3

- ✓ **Ters Çevrilmiş Rastgele İki Alt Kümenin Yer Değiştirmesi (Random swaps of reversed subsequences - reverse_swap)**

  Bu operatör rastgele uzunlukta seçilen iki parçanın terslenerek yer değiştirmeleri ile yapılmaktadır.

Önce	4	2	5	7	1	8	6	3

Sonra	6	8	1	5	7	2	4	3

- ✓ **Rastgele Yer Değiştirme (Random swaps - swap)**

  Bu operatör rastgele seçilen *i* ve *j* elemanlarının (i ≠ j) yer değiştirmesi ile yapılmaktadır.

Önce	4	2	5	7	1	8	6	3

Sonra	4	2	6	7	1	8	5	3

- ✓ **Rastgele İki Alt Kümenin Yer Değiştirmesi (Random swaps of subsequences - swab_sub)**

  Bu operatör rastgele uzunlukta seçilen iki alt parçanın yer değiştirmeleri ile yapılmaktadır.

Önce	4	2	5	7	1	8	6	3

Sonra	4	6	3	1	8	2	5	7

4.2. PSO Algoritmasının Ayrıklaştırılması

Optimizasyon problemlerinin tasarım değişkenleri değer türüne göre dikkate alındığında, ayrık optimizasyon problemlerinin çözümünde ayrık optimizasyon algoritmaları, sürekli optimizasyon problemlerinin çözümünde ise sürekli optimizasyon algoritmaları kullanılmaktadır. Tez kapsamında ele alınan GSP ayrık bir benchmark problemidir. Bu yüzden ayrık problemin çözümünde sürekli bir yöntem test edilmek istendiğinde çözüm yönteminin ayrıklaştırılarak problemin yapısına uygun bir hale getirilmesi gerekmektedir. Benzer şekilde sürekli formdaki bir optimizasyon probleminin çözümünde ayrık yöntemlerin süreklileştirilmesi de söz konusudur (Socha ve Dorigo 2008; Liao ve ark. 2014; Shayeghia ve ark, 2010).

Bu çalışmada ayrık problem GSP nin çözümünde kullanılan yöntemlerden KKO, yapısı gereği ayrık problem çözümlerine uygundur. Ancak hiyerarşik yapıda KKO ile birlikte çalışacak PSO yönteminin ayrıklaştırılarak, KKO ile uyumlu bir şekilde çalışması ve GSP çözümünde kullanılması sağlanmıştır.

Temel kavramlarda temel PSO algoritmasının detaylarına yer verilmiştir. PSO yönteminde parçacık olarak isimlendirilen çözümlerin uygunluk değerleri ve hız bilgileri bulunmaktadır. Hiyerarşik yaklaşımda KKO yönteminin ürettiği çözümler, yerel minimuma takılan çözümleri çeşitlendirmek ve rastgeleliği artırmak amacıyla PSO yöntemi tarafından başlangıç popülasyonu olarak alınır. Yani PSO, rastgele üretilmiş parçacıklar yerine KKO da her bir karıncanın bulduğu çözümlerle (tabu liste) çalışmaya başlar. Her bir karıncanın kapalı turları PSO daki parçacıkları (x) temsil etmektedir. Eşitlik 3.20 deki hız vektörü; karınca çözümlerinin her bir iterasyondaki en iyisi (pbest) ve tüm zamanların en iyi tur uzunluğunu bulan karınca (gbest) her bir iterasyonda dikkate alınarak hesaplanır. Çözüm kümesinde şehir numaralarının dışında bir şehir numarası (x) parçacık değerleri arasında yer alamayacağından hız vektörünün sınırlarına ait kısıtlama aşağıdaki gibidir:

$$V = \left[-\frac{Karınca\ Sayısı}{2}, \frac{Karınca\ Sayısı}{2}\right] \tag{4-1}$$

Dizinin eleman sayısının her bir karıncanın turundaki şehir sayısı kadar olduğu bir dizi tanımlanır. Karıncanın uygunluk değerlerini değiştirecek hız değerleri bu dizide

tutulur. Eşitlik 3.20 ye göre önce yeni hız değerleri sonrasında ise Eşitlik 3.21 e göre yeni çözüm değerleri x ler hesaplanır. Bu hesaplamada bir karıncanın şehir listesinde (tabu liste) aynı şehirlerin olmasıyla tekrarlanan değerler olabilir. Bu durumda ise karıncanın tur listesinde tekrar eden değerler kullanılmayan değerlerle yer değiştirerek KKO algoritmasındaki bu hafıza durumuna sadık kalınır.

Son olarak PSO nun çeşitlendirdiği sonuçların uygunluk değerinin KKO dakinden daha iyi olması halinde yeni değerler saklanır ve algoritmanın koşturulması sonucunda bu değerler gösterilir. Aksi halde PSO nun kötüleştirdiği değerler ihmal edilir ve bu şekilde KKO çözümlerinin iyiliği muhafaza edilir.

4.3. KKO & PSO Hiyerarşik Yaklaşım

Tez kapsamında önerilen KKO ve PSO algoritmalarının hiyerarşik bir şekilde çalışması ile, GSP ler üzerinde optimum sonuçların elde edilmesindeki başarısının artırılması hedeflenmektedir. Optimizasyon problemlerinden GSP, yapısal olarak ayrık bir benchmark problemidir. Ayrık problemlerin çözümlerinde kullanılan yöntemler, ayrık ya da sonradan ayrıklaştırılmış yöntemlerdir. Önerilen hiyerarşik yaklaşım; KKO ve ayrıklaştırılmış PSO nun hiyerarşik olarak entegre bir şekilde çalışmasıdır. Hiyerarşik yöntem, yöntemleri art arda çalışması anlamına gelmektedir. GSP yapısına çok daha uygun olan KKO nun PSO dan önce çalışarak uygun başlangıç çözümleri bulması izlenen yoldur. PSO, komşuluk operatörlerinden sonra en iyi çözüm kabul edilen yerel minimuma takılmış çözümlerin elenmesi işlevini görmüştür.

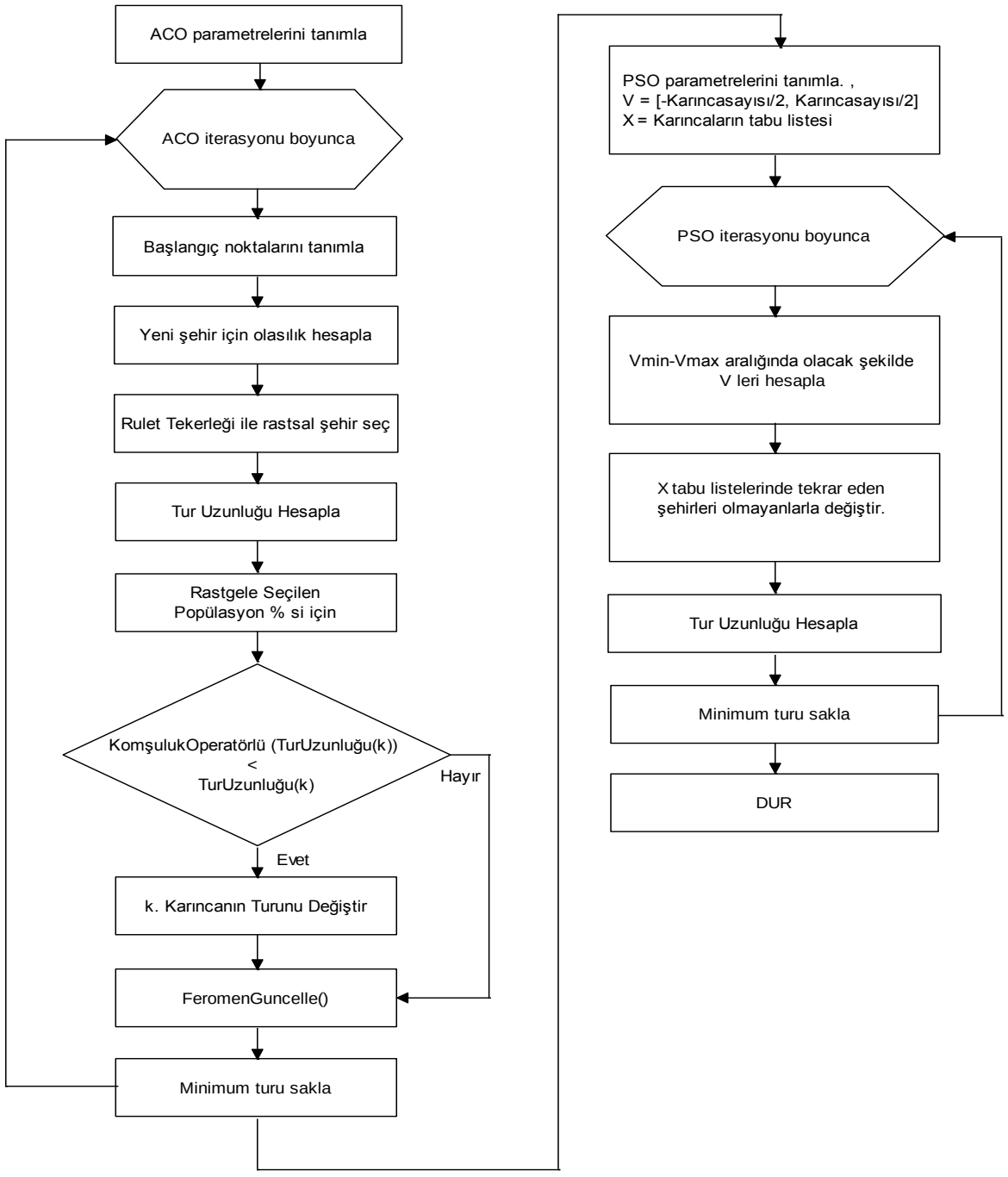
Önerilen yaklaşımda, KKO nun kendi iterasyonu boyunca bulduğu çözüm havuzundaki popülasyon bireylerinden belirli oranda rastgele seçilen bireyler, komşuluk operatörüne tabi tutulur. Komşuluk operatörü çözümünün uygunluk değeri kendi çözümünden iyiyse o karıncanın turu değiştirilir. Komşuluk operatörü yardımıyla iyileştirilen KKO çözümleri, PSO algoritmasına başlangıç popülasyonu olarak verilir. Her bir karıncanın tabu listesi PSO daki parçacıkları temsil eder. PSO daki parçacık sayısı KKO daki şehir sayısına denktir. PSO yapısı gereği hız ve pozisyon güncellemeleri yapılır. Son olarak PSO da KKO daki gibi tabu liste benzeri hafıza özelliği olmadığından, PSO çözümünden sonra tekrar eden şehirler listede olmayanlarla değiştirilir. PSO çözümü KKO çözümüyle kıyaslanır. Çözüm daha iyi ise PSO çözümü, değilse KKO çözümü bulunan en optimum sonuç olarak saklanır. Böylece elde edilen global en iyi sonuç her bir iterasyonda güncellenerek saklanmakta olup her iki yöntemin

iterasyonu tamamlandıktan sonra en iyi sonuç gösterilir. Önerilen KKO ve PSO metasezgisel yöntemlerinin hiyerarşik olarak bir arada kullanılmasına ait genel algoritmik yapının sözde kodu Şekil 4.1’ de ve akış diyagramı ise Şekil 4.2’ de gösterilmektedir.

KKO & PSO Hiyerarşik Yaklaşım

```
KKO parametrelerini tanımla.
KKO iterasyonu boyunca
{
        Başlangıç noktalarını tanımla.
        Her bir karınca için gideceği diğer şehir için olasılık hesapla.
        Hesaplanan olasılığı Rulet Tekerleği ile rastsal şehir seç.
        Turları oluşturarak tur uzunluklarını hesapla.
        Rastgele Seçilen Popülasyon % si için
        {
                if(Komşuluk Operatörü(TurUzunluğu(k)) < TurUzunluğu(k) )
                {
                        Karıncanın turunu değiştir.
                }
        }
        FeromenGuncelle();
        Minimum turu sakla.
}
PSO parametrelerini tanımla. (V = [-Karıncasayısı/2, Karıncasayısı/2 ] X = Karıncaların
tabu listesi )
PSO iterasyonu boyunca
{
        Vmin-Vmax aralığında olacak şekilde V leri hesapla.
        X konumlarını güncelle.
        X tabu listelerinde tekrar eden şehirleri olmayanlarla değiştir.
        Tur uzunluklarını hesapla.
        Minimum turu sakla.
}
```

Şekil 4.1 KKO-PSO Hiyerarşik Yaklaşım Sözde Kodu



Şekil 4.2 KKO-PSO Hiyerarşik Yaklaşım Akış Diyagramı

Yukarıda açıklanan komşuluk operatörleri KKO nun her bir iterasyonunda karıncaların tabu listelerine uygulanır. Çözüm iyi ise komşuluk operatörü çözümü kalır. Örneğin;

i. karıncanın tabu listesi aşağıdaki gibi ve tur uzunluğu 347 olsun.

4	2	3	5	7	1	8	6

Rastgele Araya Ekleme komşuluk operatörünün sonucu oluşan tabu liste aşağıdaki gibi ve tur uzunluğu 328 kabul edilsin.

4	2	8	3	5	7	1	6

Komşuluk operatörünün bulduğu tur uzunluğu karıncanın tur uzunluğundan daha iyi olduğundan bu i. karıncanın tabu listesi komşuluk operatörünün bulduğu ile değişecektir.

4.4. Deneysel Çalışmalar

Deneyler, Intel Core 2 Duo P8700 2.53 GHz işlemcili, 4 GB RAM li bir bilgisayarda gerçekleştirilmiştir. Yöntemin kodlamasında Matlab R2010a programlama dili kullanılmıştır. KKO daki karınca sayısı ve PSO daki parçacık sayısı, GSP lerdeki düğüm sayılarına eşit olarak belirlenmiştir. Deneylerde standart KKO ve ayrık PSO algoritmaları 500 iterasyon çalıştırılmıştır. Hiyerarşik yöntemde bu iterasyonun yarısı KKO diğer yarısı ise PSO algoritmasının çalışması için ayırılmıştır. Tez çalışması kapsamında TSPLIB kütüphanesinden 10 GSP seçilmiştir. Seçilen problemler oliver30, eil51, berlin52, st70, eil76,pr76, kroa100, eil101, ch150 ve tsp225 dir. Bu veri kümelerinin isimleri, düğüm sayıları ve bilinen optimum tur uzunlukları Çizelge 4.1 de sunulmuştur (TSPLIB, 2014).

Çizelge 4.1 Problemlerin düğüm sayıları ve optimum tur uzunlukları

	Problem	**Düğüm (şehir) sayıları**	**Optimum Tur Uzunlukları**
1	Oliver	30	423.74
2	Eil	51	428.87
3	Berlin	52	7544.37
4	St	70	677.11
5	Eil	76	545.39
6	Pr	76	108159.44
7	Kroa	100	21285.44
8	Eil	101	642.31
9	CH	150	6532.28
10	TSP	225	3859.00

Başlangıçta, KKO ve ayrık PSO algoritmalarının standart yalın halleriyle sonuçlar elde edilmiştir. Bulunan sonuçlar, problemlerin her biri için Çizelge 4.2 de sunulmuştur. Sonrasında Çizelge 4.1' de sunulan GSP ler, önerilen hiyerarşik yöntemle tekrar çözülmüş ve performans değerlendirmeleri yapılmıştır. Performans değerlendirmelerinde GSP çözümlerinin her biri için program 20 defa çalıştırılarak bulunan sonuçların *en iyisi, en kötüsü, ortalaması* ve *ortalamaya ait bağıl hata* (*OBH*) sonuçları elde edilmiştir. Elde edilen bu sonuçlar tablolar halinde EK' te sunulmuştur (EK 1 – EK 10). Bu hesaplamalar, problemlerin tümünde ilgili komşuluk operatörüne gönderilen (%10, %20, %30, %40 ve %50) popülasyon oranlarının her biri için yapılmıştır. Popülasyon oranlarının farklı yüzdelerde seçilmesiyle komşuluk operatörüne tabi tutulan popülasyon ile sonuçların iyileşmesi arasında bir bağ kurulması amaçlanmıştır.

OBH, aşağıdaki denklem ile bulunmaktadır.

$$OBH(\%) = \frac{B - O}{O} \times 100 \tag{4-2}$$

Burada *O* Çizelge 4.1' de belirtilen optimum tur uzunluğudur. *B* ise önerilen hiyerarşik yöntem tarafından bulunan en iyi turların ortalamasıdır.

Çizelge 4.2 Standart KKO ve ayrık PSO ile bulunan tur uzunlukları.

	Yöntem	En iyi	En Kötü	Ortalama	OBH (%)	CPU Süresi (s)
Oliver 30	KKO	423,91	426,54	424,52	0,18	76,25
	PSO	426,68	495,78	459,4	8,42	2,49
Eil 51	KKO	440,40	451,54	446,44	4,10	203,06
	PSO	808,11	1034,78	914,72	113,28	6,09
Berlin 52	KKO	7544,37	7659,25	7610,20	0,87	213,81
	PSO	12762,61	15323,02	14435,78	91,35	6,44
St 70	KKO	695,98	709,70	702,22	3,71	393,02
	PSO	1066,51	1663,2	1317,8	94,62	11,41
Eil 76	KKO	555,85	562,56	559,87	2,65	476,93
	PSO	1317,64	1679	1456	166,96	13,76
Pr 76	KKO	113774,15	116547,82	115124,32	6,44	494,85
	PSO	127212,44	198327,5	153158,4	41,60	13,60
kroA 100	KKO	22455,89	22927,47	22690,09	6,60	891,74
	PSO	70459,404	89798,2	83341,94	291,54	23,42
Eil 101	KKO	669,76	692,23	685,33	6,70	889,75
	PSO	1469,4427	2414,916	1764,317	174,68	24,14
Ch 150	KKO	6647,13	6719,61	6680,73	2,27	2182,97
	PSO	18251,697	22850,31	21380,08	227,30	58,57
Tsp 225	KKO	4122,81	4171,58	4149,34	7,52	6024,34
	PSO	6873,7	16982	9942,6	157,65	160,08

Çizelge 4.2 incelendiğinde, GSP nin yapısına uygun olan standart KKO metodunun çözümlerinin, ayrık PSO metodunun çözümlerinden oldukça iyi olduğu görülmektedir. Her bir problemin KKO-PSO Hiyerarşik Yaklaşımla elde edilen çözümlerinin sunulduğu EK 1-10 arasındaki sonuçlar incelendiğinde, hiyerarşik yaklaşım, tüm komşuluk operatörlerinin her bir yüzdesi için, Çizelge 4.2 deki KKO ve ayrık PSO nun standart halleriyle buldukları sonuçlardan her zaman iyi sonuçlar üretmiştir. Örneğin *Eil101* probleminde bulunan ortalama bağıl hata KKO da *%6,70* PSO da *%174,68* iken, EK 8'deki *Eil101* problemi için bulunan en kötü bağıl hata *swap_%40* ta %1,69, en iyi bağıl hata ise *insert_%10* da %1,28 dir. Bağıl hatalardaki yorumlarda en büyük bağıl hatalar "en kötü", ve en küçük bağıl hatalar "en iyi" şeklinde ifade edilmiştir.

Önerilen KKO-PSO Hiyerarşik yöntemin standart KKO ve standart PSO ile adil bir şekilde kıyaslanabilmesi için; algoritmaların çalışmalarında geçirdikleri evrim sayılarının aynı olması gerekmektedir. Her bir iterasyonda her bir karınca tarafından bulunan çözüm (tabu listelerindeki şehirlerin sıralanması) ve bulunan bu çözümün tur uzunluğunun hesaplanması bu karıncanın geçirdiği *bir evrim* olarak düşünülebilir. Dolayısıyla KKO-PSO Hiyerarşik yöntemin standart algoritmaların geçirdikleri evrime denk bir iterasyona tabi tutulması şarttır. Standart KKO ve standart PSO ile KKO-PSO Hiyerarşik Yöntemin iterasyon sayıları Eşitlik 4.3' teki gibi hesaplanmaktadır.

$$KKO_{EvrimSayısı=KKO_{iter}} \text{ x } KarıncaSayısı$$

$$PSO_{EvrimSayısı=PSO_{iter}} \text{ x } ParçacıkSayısı \quad (4.3)$$

$$KKOPSO_Hiyerarşik_{EvrimSayısı=KKO_{iter}} \text{ x } (KarıncaSayısı+Seçilen\ Pop.\%\ si)+PSO_{iter} \text{ x } KarıncaSayısı$$

Çizelge 4.1' de verilen GSP' ler için, komşuluk operatörlerinin her birinde belirtilen popülasyon yüzdelerindeki bulunan sonuçların yorumlanmasına aşağıda yer verilmiştir. Tüm GSP' lerde tek tek yorumlanan sonuçlar sayesinde komşuluk operatörü ve popülasyon yüzdesi açısından bir genelleme yapılarak hiyerarşik yaklaşıma en uygun komşuluk operatörü ve yüzdesi belirlenmiştir.

Önerilen yaklaşımın *Oliver30* problemi için çözüm sonuçları EK 1' de sunulmuştur.

Oliver30 probleminde bulunan ortalama bağıl hata Çizelge 4.2' de KKO için %0,18 PSO için %8,42 iken EK 1' de görüldüğü gibi bulunan en büyük ortalama bağıl hatalar *insert_%10, insert_%40, insert_%50, insertsub_%10* için %0,04 iken en küçük bağıl hatalar *insertsub_%50, swap_%30* ve *swapsub_%10* için %0 dır. Yani önerilen KKO-PSO hiyerarşik yaklaşım, bu komşuluk operatörlerinin belirtilen yüzdeleri için optimum sonuca ulaşmıştır.

Oliver30 problemi sonuçlarında, komşuluk operatörleri incelendiğinde en başarılı olanlar %0 bağıl hata ile *insertsub, swap* ve *swapsub* dir. Bağıl hata oranını %0,04 olarak en kötü bulan komşuluk operatörleri ise *insert* ve *insertsub* dır.

Ayrıca bu problemde komşuluk operatörüne giren popülasyon yüzdeleri kıyaslandığında *%10, %30* ve *%50* oranlarında %0 la en başarılı sonuçlar elde edilirken, bağıl hata oranını %0,04 olarak en kötü bulan yüzdeler ise %10, %40 ve %50 dir.

EK 1' deki sonuçlarda popülasyon yüzdesi arttıkça komşuluk operatörlerinden *insertsub* (%0,04' dan %0' a) ve *reverse* (%0,03' dan %0,01' a) için tutarlı olarak sonuçların iyileştiği görülmektedir. Diğer komşuluk operatörlerinde dağınık bir performans sergilenmektedir.

Sonuçta *Oliver30* problemi için önerilen KKO-PSO Hiyerarşik yaklaşımda en iyi sonucu veren (%0) ve popülasyon yüzdesi arttıkça iyileşmenin oransal arttığı *insertsub* operatörünün problemin çözümüne uygun olduğu söylenebilir.

Önerilen yaklaşımın *Eil51* problemi için çözüm sonuçları EK 2' de sunulmuştur.

Eil51 probleminde bulunan ortalama bağıl hata Çizelge 4.2' de KKO için %4,10 PSO için %113,28 iken EK 2' de görüldüğü gibi bulunan en kötü ortalama bağıl hatalar *insert_%50, insertsub_%10, insertsub_%40* ve *insertsub_%50* için %0,42 iken en iyi bağıl hatalar *reverse_%10* ve *reverse_%20* için %0,19 dur.

Eil51 problemi sonuçlarında, *komşuluk operatörleri* incelendiğinde en başarılı olan %0,19 bağıl hata ile *reverse* dir. Bağıl hata oranını %0,42 olarak en kötü bulan komşuluk operatörleri ise *insert* ve *insertsub* dır.

Ayrıca bu problemde komşuluk operatörüne giren *popülasyon yüzdeleri* kıyaslandığında *%10* ve *%20* oranlarında %0,19 la en başarılı sonuçlar elde edilirken, bağıl hata oranını %0,42 olarak en kötü bulan yüzdeler ise *%10, %40* ve *%50* dir.

EK 2' deki sonuçlarda popülasyon yüzdesi arttıkça tutarlı olarak sonuçların iyileştiği komşuluk operatörü bulunmamaktadır.

Sonuçta *Eil51* problemi için en iyi sonucu veren (%0,19) ve bağıl hatası en kötü olan komşuluk operatörlerinden olmayan *reverse* operatörünün probleme uygun olduğu söylenebilir.

Önerilen yaklaşımın *Berlin52* problemi için çözüm sonuçları EK 3' te sunulmuştur.

Berlin52 probleminde bulunan ortalama bağıl hata Çizelge 4.2' de KKO için %0,87 PSO için %91,35 iken EK 3' te görüldüğü gibi bulunan en kötü ortalama bağıl hatalar *insert_(%10, %20, %30), insertsub_(%10, %20, %30), reverse_(%10, %20, %30, %40, %50), reverseinsert_(%10, %20, %30, %40, %50), reverseswap_(%10, %20, %30, %40, %50), swap_(%10, %20)* ve *swapsub_(%10, %20, %40, %50)* için %0,01 iken en iyi bağıl hatalar *insert_(%40,%50), insertsub_(%40,%50), swap_(%30,%40,%50)* ve *swapsub_(%30)* için %0 dır. Yani KKO-PSO hiyerarşik yaklaşım bu komşuluk operatörlerinin belirtilen yüzdeleri için optimum sonuca ulaşmıştır.

Berlin52 problemi sonuçlarında, *komşuluk operatörleri* incelendiğinde en başarılı olanlar %0 bağıl hata ile *insert, insertsub, swap* ve *swapsub* dir. %0,01 olan en kötü bağıl hata oranı tüm komşuluk operatörlerinde bulunmuştur.

Ayrıca bu problemde komşuluk operatörüne giren *popülasyon yüzdeleri* kıyaslandığında *%30, %40* ve *%50* oranlarında %0 la en başarılı sonuçlar elde edilirken, bağıl hata oranını %0,01 olarak en kötü bulan yüzdeler ise hepsidir.

EK 3' teki sonuçlarda popülasyon yüzdesi arttıkça tutarlı olarak sonuçların iyileştiği komşuluk operatörleri *insert* (%0,01'den %0' a), *insertsub* (%0,01' den %0' a)

ve *swap* (%0,01' den %0' a) dir. Diğer komşuluk operatörlerinde dağınık bir performans sergilenmektedir.

Sonuçta *Berlin52* problemi için önerilen KKO-PSO Hiyerarşik yaklaşımda en iyi sonucu veren (%0) ve popülasyon yüzdesi arttıkça iyileşmenin oransal arttığı *insert* ve *insertsub* operatörlerinin probleme uygun olduğu söylenebilir.

Önerilen yaklaşımın *St70* problemi için çözüm sonuçları EK 4' te sunulmuştur.

St70 probleminde bulunan ortalama bağıl hata Çizelge 4.2' de KKO için %3,71 PSO için %94,62 iken EK 4' te görüldüğü gibi bulunan en kötü ortalama bağıl hatalar *insertsub_%20* ve *insertsub_%30* için %0,93 iken en iyi bağıl hata *insert_%10* için %0,74 dür.

St70 problemi sonuçlarında, *komşuluk operatörleri* incelendiğinde en başarılı olan %0,74 bağıl hata ile *insert* dir. Bağıl hata oranını %0,93 olarak en kötü bulan komşuluk operatörü ise *insertsub* dır.

Ayrıca bu problemde komşuluk operatörüne giren *popülasyon yüzdeleri* kıyaslandığında *%10* oranında %0,74 la en başarılı sonuç elde edilirken, bağıl hata oranını %0,93 olarak en fazla bulan yüzdeler ise *%20* ve *%30* dir.

EK 4' teki sonuçlarda popülasyon yüzdesi arttıkça tutarlı olarak sonuçların iyileştiği komşuluk operatörü bulunmamaktadır.

Sonuçta *St70* problemi için en iyi sonucu veren (%0,74) ve bağıl hatası en fazla olan komşuluk operatörlerinden olmayan *insert* operatörünün probleme uygun olduğu söylenebilir.

Önerilen yaklaşımın *Eil76* problemi için çözüm sonuçları EK 5' te sunulmuştur.

Eil76 probleminde bulunan ortalama bağıl hata Çizelge 4.2' de KKO için %2,65 PSO için %166,96 iken EK 5' te görüldüğü gibi bulunan en kötü ortalama bağıl hatalar *insertsub_%50, reverse_%50, reverseinsert_%20* ve *reverseswap_%30* için %0,75 iken en iyi bağıl hata *reverseinsert_%10* için %0,57 dir.

Eil76 problemi sonuçlarında, *komşuluk operatörleri* incelendiğinde en başarılı olan %0,57 bağıl hata ile *reverseinsert* dir. Bağıl hata oranını %0,75 olarak en kötü bulan komşuluk operatörleri ise *insertsub, reverse, reverseinsert* ve *reverseswap* dır.

Ayrıca bu problemde komşuluk operatörüne giren *popülasyon yüzdeleri* kıyaslandığında *%10* oranında %0,57 la en başarılı sonuç elde edilirken, bağıl hata oranını %0,75 olarak en kötü bulan yüzdeler ise *%20, %30* ve *%50* dir.

EK 5' teki sonuçlarda popülasyon yüzdesi arttıkça tutarlı olarak sonuçların iyileştiği komşuluk operatörü bulunmamaktadır.

Sonuçta *Eil76* problemi için en iyi sonucu veren (%0,57), ancak bağıl hatası en fazla komşuluk operatörlerinden olan *reverseinsert* operatörünün probleme uygun olduğu söylenebilir. Bu problemde komşuluk operatörlerinin buldukları en büyük yüzdeler birbirine çok yakındır.

Önerilen yaklaşımın *Pr76* problemi için çözüm sonuçları EK 6' da sunulmuştur.

Pr76 probleminde bulunan ortalama bağıl hata Çizelge 4.2' de KKO için %6,44 PSO için %41,60 iken EK 6' da görüldüğü gibi bulunan en kötü ortalama bağıl hatalar *swap_%50* için %3,47 iken en iyi bağıl hata *reverse_%10* için %2,47 dir.

Pr76 problemi sonuçlarında*, komşuluk operatörleri* incelendiğinde en başarılı olan %2,47 bağıl hata ile *reverse* dir. Bağıl hata oranını %3,47 olarak en kötü bulan komşuluk operatörü ise *swap* dır.

Ayrıca bu problemde komşuluk operatörüne giren *popülasyon yüzdeleri* kıyaslandığında *%10* oranında %2,47 la en başarılı sonuç elde edilirken, bağıl hata oranını %3,47 olarak en kötü bulan yüzde ise *%50* dir.

EK 6' daki sonuçlarda popülasyon yüzdesi arttıkça tutarlı olarak sonuçların iyileştiği komşuluk operatörü bulunmamaktadır.

Sonuçta *Pr76* problemi için en iyi sonucu veren (%2,47) *reverse* operatörünün probleme uygun olduğu söylenebilir. Ayrıca *reverse* operatörü bağıl hatası en fazla olan komşuluk operatörlerinden biri değildir.

Önerilen yaklaşımın *kroA100* problemi için çözüm sonuçları EK 7' de sunulmuştur.

kroA100 probleminde bulunan ortalama bağıl hata Çizelge 4.2' de KKO için %6,60 PSO için %291,54 iken EK 7' de görüldüğü gibi bulunan en kötü ortalama bağıl hata *insert_%50* için %2,97 iken en iyi bağıl hata *insertsub_%20* için %2,03 dür.

kroA100 problemi sonuçlarında, komşuluk operatörleri incelendiğinde en başarılı olan %2,03 bağıl hata ile *insertsub* dir. Bağıl hata oranını %2,97 olarak en kötü bulan komşuluk operatörü ise *insert* dır.

Ayrıca bu problemde komşuluk operatörüne giren popülasyon yüzdeleri kıyaslandığında *%20* oranında %2,03 la en başarılı sonuç elde edilirken, bağıl hata oranını %2,97 olarak en kötü bulan yüzde ise %50 dir.

EK 7' deki sonuçlarda popülasyon yüzdesi arttıkça tutarlı olarak sonuçların iyileştiği komşuluk operatörü bulunmamaktadır.

Sonuçta *kroA100* problemi için en iyi sonucu veren (%2,03) ve bağıl hatası en fazla olan komşuluk operatörlerinden olmayan *insertsub* operatörünün probleme uygun olduğu söylenebilir.

Önerilen yaklaşımın *Eil101* problemi için çözüm sonuçları EK 8' de sunulmuştur.

Eil101 probleminde bulunan ortalama bağıl hata Çizelge 4.2' de KKO için %6,70 PSO için %174,68 iken EK 8' de görüldüğü gibi bulunan en kötü ortalama bağıl hata *swap_%40* için %1,69 iken en iyi bağıl hata *insert_%10* için %1,28 dir.

Eil101 problemi sonuçlarında, *komşuluk operatörleri* incelendiğinde en başarılı olan %1,28 bağıl hata ile *insert* dir. Bağıl hata oranını %1,69 olarak en kötü bulan komşuluk operatörü ise *swap* dır.

Ayrıca bu problemde komşuluk operatörüne giren *popülasyon yüzdeleri* kıyaslandığında *%10* oranında %1,28 la en başarılı sonuç elde edilirken, bağıl hata oranını %1,69 olarak en kötü bulan yüzde ise %40 dir.

EK 8' deki sonuçlarda popülasyon yüzdesi arttıkça tutarlı olarak sonuçların iyileştiği komşuluk operatörü bulunmamaktadır.

Sonuçta *Eil101* problemi için en iyi sonucu veren (%1,28) ve bağıl hatası en fazla olan komşuluk operatörlerinden olmayan *insert* operatörünün probleme uygun olduğu söylenebilir.

Önerilen yaklaşımın *Ch150* problemi için çözüm sonuçları EK 9' da sunulmuştur.

Ch150 probleminde bulunan ortalama bağıl hata Çizelge 4.2' de KKO için %2,27 PSO için %227,30 iken EK 9' da görüldüğü gibi bulunan en kötü ortalama bağıl hata *swap_%40* için %1,36 iken en iyi bağıl hata *insert_%10* için %1,04 dır.

Ch150 problemi sonuçlarında, *komşuluk operatörleri* incelendiğinde en başarılı olan %1,04 bağıl hata ile *insert* dir. Bağıl hata oranını %1,36 olarak en kötü bulan komşuluk operatörü ise *swap* dır.

Ayrıca bu problemde komşuluk operatörüne giren *popülasyon yüzdeleri* kıyaslandığında *%10* oranında %1,04 la en başarılı sonuçlar elde edilirken, bağıl hata oranını %1,36 olarak en kötü bulan yüzde ise %40 dir.

EK 9' daki sonuçlarda popülasyon yüzdesi arttıkça tutarlı olarak sonuçların iyileştiği komşuluk operatörü *reverse* (%1,35' den %1,34' e) dir. Diğer komşuluk operatörlerinde dağınık bir performans sergilenmektedir.

Sonuçta *Ch150* problemi için en iyi sonucu veren (%1,04) ve bağıl hatası en fazla olan komşuluk operatörlerinden olmayan *insert* operatörünün probleme uygun olduğu söylenebilir.

Önerilen yaklaşımın *Tsp225* problemi için çözüm sonuçları EK 10' da sunulmuştur.

Tsp225 probleminde bulunan ortalama bağıl hata Çizelge 4.2' de KKO için %7,52 PSO için %157,65 iken EK 10' da görüldüğü gibi bulunan en kötü ortalama bağıl hatalar *insert_(%30,%50), insertsub_(%10,%30)* ve *reverse_%50, swap_%20* için %5,55 iken en iyi bağıl hata *reverse_%30* için %4,99 dir.

Tsp225 problemi sonuçlarında, *komşuluk operatörleri* incelendiğinde en başarılı olan %4,99 bağıl hata ile *reverse* dir. Bağıl hata oranını %5,55 olarak en kötü bulan komşuluk operatörleri ise *insert, insertsub, reverse* ve *swap* dır.

Ayrıca bu problemde komşuluk operatörüne giren *popülasyon yüzdeleri* kıyaslandığında *%30* oranında %4,99 la en başarılı sonuç elde edilirken, bağıl hata oranını %5,55 olarak en kötü bulan yüzdeler ise *%10, %20, %30* ve *%50* dir.

EK 10' daki sonuçlarda popülasyon yüzdesi arttıkça tutarlı olarak sonuçların iyileştiği komşuluk operatörü bulunmamaktadır.

Sonuçta *Tsp225* problemi için en iyi sonucu veren (%4,99), ancak bağıl hatası en fazla komşuluk operatörlerinden olan *reverse* operatörünün probleme uygun olduğu söylenebilir. Bu problemde komşuluk operatörlerinin buldukları en büyük yüzdeler birbirine çok yakındır.

Ayrıca, 10 adet GSP' nin her bir komşuluk operatörüne belirli yüzde oranlarıyla tabi olmasıyla elde edilen EK 1-10 arasındaki sonuçların grafiksel gösterimleri EK 11' de sunulmuştur.

EK 1-10 arasında çözümleri sunulan problemlerin her biri için yukarıda yapılan açıklamalardan özet bir sonuç çıkarmak mümkündür: Toplam 10 GSP' de, komşuluk operatörü bakımından en iyi sonuçlar en fazla *insert* (4 problem) en az *reverseswap* (0 problem) operatörlerinde, en kötü sonuçlar ise en fazla *insertsub* (6 problem) en az *reverseinsert, reverseswap* ve *swapsub* (2 problem) operatörlerinde elde edilmiştir. Aynı şekilde komşuluk operatörüne tabi olan yüzdeler bakımından EK 1-10 lardaki çizelgeler incelendiğinde; en iyi sonuçlar en fazla *%10* (8 problem), en az *%40* (1 problem) oranlarında elde edilmiştir (Çizelge 4.3 ve Çizelge 4.4). Komşuluk operatörüne tabi olan yüzdeler bakımından EK 1-10 lardaki çizelgeler incelendiğinde maksimum sonuçlar en fazla *%50* (7 problem), en az *%10,%20,%30* (4 problem)

oranlarında elde edilmiştir (Çizelge 4.5 ve Çizelge 4.6). Dolayısıyla testlerde *Berlin52, St70, Eil101* ve *Ch150* olmak üzere 4 problemde en iyi sonuçları üreten *insert* en başarılı komşuluk operatörü iken 6 problemde en büyük sonuçları üreten *insertsub* operatörü başarısı en düşük olarak değerlendirilebilir. Testlerde 8 problemde (Oliver30, Eil51, St70, Eil76, Pr76, Eil101, Ch150 ve Tsp225) %10 oranı en başarılı komşuluk operatör yüzdesi olarak, %50 oranı ise en başarısız yüzde olarak değerlendirilebilir. Genel olarak bakıldığında tüm GSP' lerde komşuluk operatörüne tabi olan popülasyon yüzdeleri ne kadar az olursa sonuçlar o oranda başarılı olmaktadır (Bakınız EK 11).

EK 1-10 arasında çözümleri sunulan problemlerin sonuçlarının hesaplama sürelerine bakıldığında, her bir problemde standart KKO algoritmasının hesaplama süreleri ayrık PSO' dan ve önerilen KKO-PSO hiyerarşik yöntemden fazladır. Ayrık PSO algoritması ise her bir problemde en kısa süreli çalışan yöntemdir. Önerilen hiyerarşik yöntem, GSP çözümlerinde PSO' ya nazaran daha iyi sonuçlar üreten KKO dan daha iyi sonuçlar ürettiği gibi, hesaplama süresi bakımından da KKO ya göre çok daha kısa sürelerde çalışmaktadır.

Çizelge 4.3 Komşuluk Operatörü Bakımından En İyi Sonuçlar

insert	**insert_sub**	**reverse**	**reverse_insert**	**reverse_swap**	**swap**	**swab_sub**
Berlin52	Oliver30	Eil51	Eil76		Oliver30	Oliver30
St70	Berlin52	Pr76			Berlin52	Berlin52
Eil101	kroA100	Tsp225				
Ch150						
4	3	3	1	0	2	2

Çizelge 4.4 Komşuluk Operatörü Bakımından En Kötü Sonuçlar

insert	**insert_sub**	**reverse**	**reverse_insert**	**reverse_swap**	**swap**	**swab_sub**
Oliver30	Oliver30	Berlin52	Berlin52	Berlin52	Berlin52	Berlin52
Eil51	Eil51	Eil76	Eil76	Eil76	Pr76	Eil76
Berlin52	Berlin52	Tsp225			Eil101	
kroA100	St70				Ch150	
Tsp225	Eil76				Tsp225	
	Tsp225					
5	6	3	2	2	5	2

Çizelge 4.5 Komşuluk Operatörüne Tabi Olan Yüzdeler Bakımından En İyi Sonuçlar

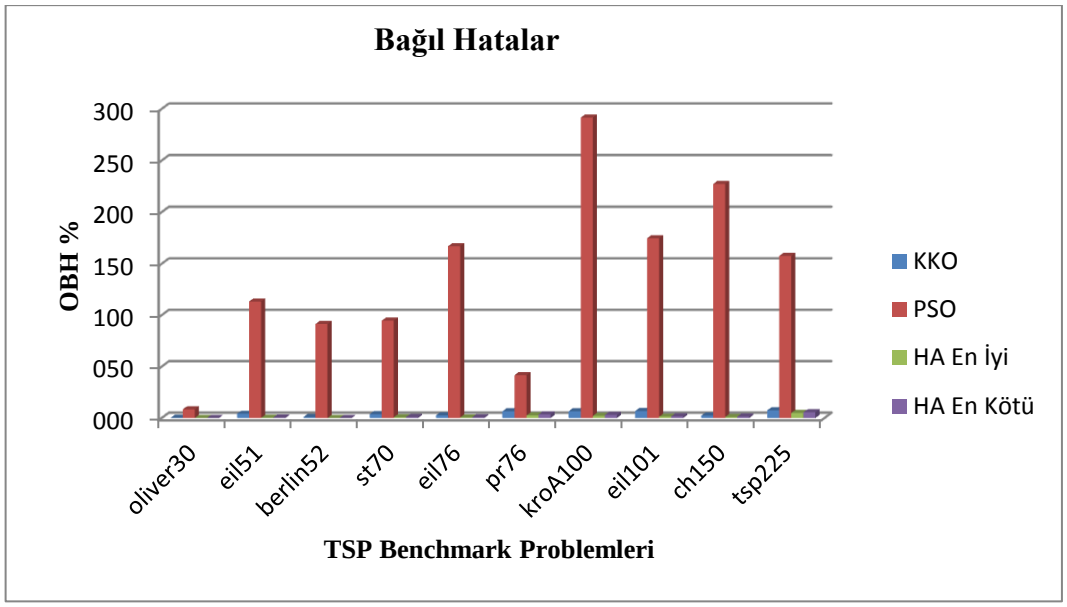
	Oliver30	**Eil51**	**Berlin52**	**St70**	**Eil76**	**Pr76**	**kroA100**	**Eil101**	**Ch150**	**Tsp225**	
%10	+	+		+	+	+		+	+	+	**8**
%20		+					+				**2**
%30	+		+								**2**
%40			+								**1**
%50	+		+								**2**

Çizelge 4.6 Komşuluk Operatörüne Tabi Olan Yüzdeler Bakımından En Kötü Sonuçlar

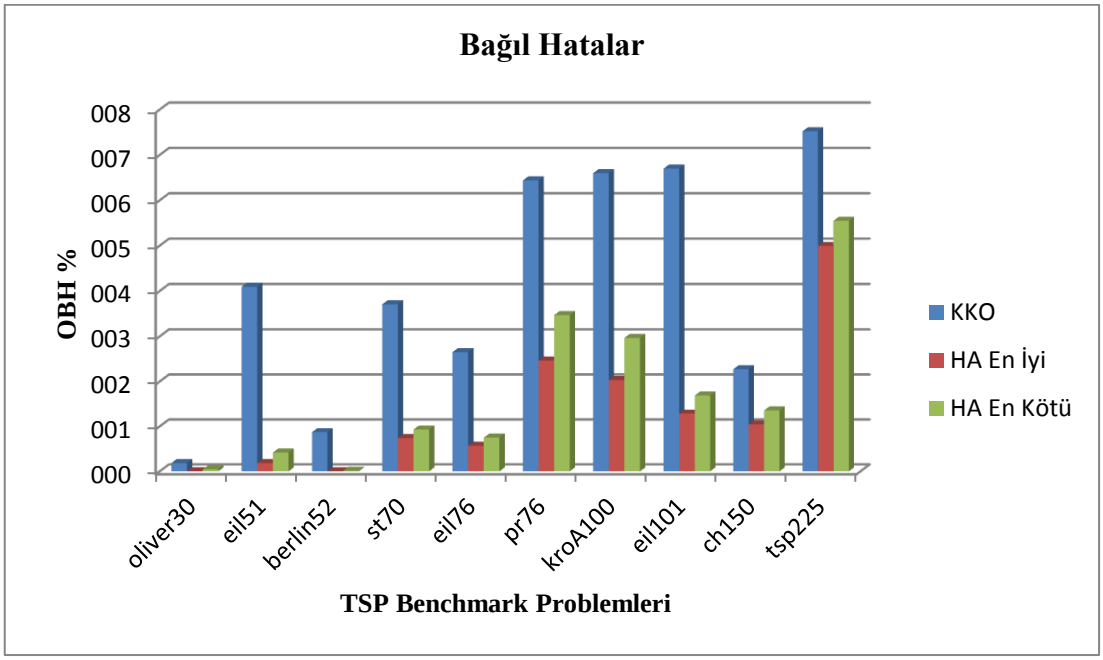
	Oliver30	**Eil51**	**Berlin52**	**St70**	**Eil76**	**Pr76**	**kroA100**	**Eil101**	**Ch150**	**Tsp225**	
%10	+	+	+							+	**4**
%20			+	+	+					+	**4**
%30			+	+	+					+	**4**
%40	+	+	+					+	+		**5**
%50	+	+	+		+	+	+			+	**7**

Komşuluk operatörüne tabi tutulan bireylerin yüzde oranları ile herhangi bir kıyaslama söz konusu olamamaktadır. Örneğin Oliver30 için *insertsub* komşuluk operatörüne tabi tutulan karınca yüzdesi arttıkça çözüm oransal olarak optimuma yaklaşmış hatta %50 popülasyon yüzdesinde optimum çözüm bulunmuştur (OBH %0). Ancak EK 1-10 lardaki çizelgeler incelendiğinde komşuluk operatörüne tabi olan popülasyon oranı arttıkça bağıl hatalarda oransal bir iyileştirmeye her zaman rastlanılmamıştır. Komşuluk operatörüne tabi tutulan bireylerin artırılmasıyla çeşitliliğin artması ve optimum sonuca yapacağı katkının fazlalaşması hedeflenmesine rağmen KKO algoritmasının yapısındaki olasılıksal seçim buna her zaman olanak tanımamıştır.

Problemin yapısına uygun olan KKO yönteminin PSO ya nazaran daha iyi sonuçlar verdiği yukarıda açıklanmıştır (Çizelge 4.2). Ayrıca önerilen hiyerarşik yöntemin de KKO dan her zaman daha iyi sonuçlar ürettiği ispatlanmıştır (Çizelge 4.2 ve EK 1-10 çizelgeleri). Aşağıdaki şekil 4.3 ve şekil 4.4' te de görüldüğü üzere önerilen hiyerarşik yöntemin ürettiği en kötü sonuç dahi her zaman KKO ve PSO algoritmalarından daha iyi olmaktadır.



Şekil 4.3 KKO, PSO, Hiyerarşik Yöntemin En iyi, Hiyerarşik Yöntemin en kötü bağıl hata oranları karşılaştırması



Şekil 4.4 KKO, Hiyerarşik Yöntemin En iyi, Hiyerarşik Yöntemin en kötü bağıl hata oranları karşılaştırması

Tez kapsamında önerilen hiyerarşik yöntem, literatürdeki çalışmalardan Gündüz ve ark. tarafından önerilen çalışmadaki sayısal sonuçlarla da karşılaştırılmıştır. Bu çalışmada KKO-ABC tabanlı hiyerarşik bir yaklaşımla GSPler çözülerek ortalama bağıl hatalar sunulmuştur (Gündüz ve ark., 2014). Literatürden alınan KKO-ABC Hiyerarşik

yaklaşım ile bu tez çalışmasında önerilen KKO-PSO Hiyerarşik yaklaşım ların ortalama bağıl hataları Çizelge 4.7 de verilmiştir.

Çizelge 4.7 KKO-PSO Hiyerarşik Yöntem ile KKO-ABC Hiyerarşik Yöntem Bağıl Hata Yüzdeleri

Problem	**KKO-ABC Hiyerarşik Yaklaşım (Gündüz ve ark., 2014)**	**KKO-PSO Hiyerarşik Yaklaşım**
Oliver30	***0.00***	***0.00***
Eil51	3.39	**0.19**
Berlin52	***0.00***	***0.00***
St70	3.47	**0.74**
Eil76	2.31	**0.57**
Pr76	6.39	**2.47**
KroA100	5.40	**2.03**
Eil101	6.39	**1.28**
Ch150	2.21	**1.04**
Tsp225	7.74	**4.99**

Çizelge 4.8 KKO-PSO Hiyerarşik Yöntem ile KKO-ABC Hiyerarşik Yöntem Bağıl Hata Yüzdelerinin Karşılaştırması

		KKO-ABC
KKO-PSO	İyi	8
	Eşit	2
	Kötü	0

Çizelge 4.8’ de görüldüğü gibi tez çalışması kapsamında önerilen KKO-PSO hiyerarşik yönteminin performansı KKO-ABC dan daha iyidir.

5. SONUÇLAR VE ÖNERİLER

5.1 Sonuçlar

Donanımı yüksek olan bilgisayarlar kullanılsa da matematiksel yöntemlerle orta boyuttaki bir GSP' nin çözümü günlerce sürebilmektedir. Metasezgisel yöntemler her zaman optimum sonucu garanti edemeseler de işlem süresinin çok kısa olması ve sonuca daha az arama ile ulaşmaları kendilerine olan ilgiyi arttırmıştır.

Sürü zekâsı temeline dayanan metasezgisel algoritmalardan Karınca Kolonisi Optimizasyonu (KKO) ve Parçacık Sürü Optimizasyonu (PSO) nin yapılarına uygun problemleri optimize etme başarıları için iki farklı yöntem izlenmiştir. Bunlardan ilki mevcut algoritmanın davranışı, parametreleri vb. kriterleri üzerinde iyileştirme yapılmasıdır. Diğer bir yöntem ise doğa esinli bu iki yöntemin birlikte organize bir şekilde çalışmasını sağlayabilmektir.

Gezgin Satıcı Problemi, günlük hayattaki farklı problemlere uyarlanabilen, bu yüzden etkin çözümler elde edilmesi sağlandığında günlük hayatı kolaylaştıracak olan bir eniyileme problemidir. Bu çalışmada, literatürde optimizasyon problemlerinin performans değerlendirmelerinde sıkça kullanılan Gezgin Satıcı Problemi (Traveling Salesman Problem - TSP) ne ait test fonksiyonlarından bazıları üzerinde Karınca Kolonisi Optimizasyonu (KKO) , Parçacık Sürü Optimizasyonu (PSO) ve bu iki yöntemin hiyerarşik bir yaklaşımla birlikte çalışmaları test edilerek sonuçları gözlenmiştir.

Ayrık bir problem olarak nitelendirilen GSP problemlerinin çözümünde etkin olarak kullanılan ve optimum sonuçlar üretebilen ayrık yöntemlerden KKO, kendisi gibi doğada yaşayan balık, kuş vb. canlıların yaşamlarını devam ettirebilmek için sergiledikleri hareketlerden yola çıkarak ortaya çıkan yeni metasezgisel yöntemlerden PSO ile bir arada hiyerarşik bir yaklaşımla kullanılması ile verimli sonuçlar elde edilmiştir. PSO, yapısı gereği sürekli optimizasyon problemlerinin çözümünde başarılı olurken, ayrık bir problem olan GSP problemlerine uyarlanması için yeniden düzenlenmesi (ayrıklaştırılması) gerekmektedir.

KKO ve PSO bir arada çalışarak bir GSP probleminin çözümünde yalın halleri ile buldukları çözümlerin iyileştirilmesi için iki yöntemin bir arada hiyerarşik bir şekilde kullanılması bu tez kapsamında önerilmiştir. Şöyle ki iterasyon sayısınca KKO çalışır ve kendi sonuçlarını üretir. Ürettiği bu sonuçlar (karıncaların bulduğu en iyi çözümler)

ayrıklaştırılan PSO yöntemine parametre olarak gönderilir. GSP problemlerinin yapısına uygun hale getirilen PSO ise kendi iterasyonlarında KKO dan gelen sonuçları kendi yapısına uygun şekilde tekrar bir iyileştirme olup olmayacağını belirler. Dolayısıyla KKO-PSO hiyerarşik yapıda bulunan sonuçlar, bu algoritmaların kendilerinin ürettikleri sonuçlardan her zaman daha iyi sonuçlar bulabilme olasılığı çok yüksektir. Çünkü hiyerarşik yapı her zaman iyi sonucu saklamaktadır. Başka bir deyişle KKO dan sonra PSO nun çalışması ile elde edilen sonuçlar, PSO nun daha iyi bir sonuç üretememesi durumunda bile en kötü ihtimalle KKO sonuçlarına eş değer olacaktır.

Bu tez çalışmasında önerilen KKO-PSO hiyerarşik yöntem, TSPLIB den alınan 10 adet GSP problemlerine ait çözümlerinin iyileştirilmesinde test edilmiştir. Çalışmada hiyerarşik yöntem 7 adet komşuluk operatörünün KKO-PSO hiyerarşik yaklaşıma entegre edilmesiyle optimum çözümlere katkı sağlanmıştır. Kullanılan komşuluk operatörleri, Araya Rastgele Ekleme (Random insertions - insert), Araya Rastgele Alt Küme Ekleme (Random insertions of subsequences - insert_sub), Alt Kümenin Ters Çevrilmesi (Reversing a subsequence - reverse), Ters Çevrilmiş Alt Kümenin Rastgele Araya Eklenmesi (Random insertions of reversed subsequences - reverse_insert), Ters Çevrilmiş Rastgele İki Alt Kümenin Yer Değiştirmesi (Random swaps of reversed subsequences - reverse_swap), Rastgele Yer Değiştirme (Random swaps - swap) ve Rastgele İki Alt Kümenin Yer Değiştirmesi (Random swaps of subsequences - swab_sub) dir.

GSP problemlerinde şehir sayısı arttıkça hangi komşuluk operatörlerinin kullanılmasına dair bir genelleme tam olarak yapılamamaktadır. Ancak 52 düğüme sahip Berlin52 ve 150 düğüme sahip Ch150 de başarılı olan insert operatörü küçük ve büyük ölçekli GSP lerde başarı göstermiştir (Çizelge 4.3). GSP' lere uygulanan komşuluk operatörleri yüzdeleri ile ilgili bir genelleme yapmak mümkündür. Çizelge 4.5 ve Çizelge 4.6 incelendiğinde komşuluk operatörü yüzdelerinin artmasıyla başarı oransal olarak düşmüştür. Önerilen hiyerarşik yöntemde yerel arama KKO algoritmasında kullanılan Rulet Tekerleği seçim mekanizmasına göre yapılmaktadır. Global aramayı artıran komşuluk operatörlerinin kullanılması yerel arama ve global arama arasındaki dengeyi bozduğundan komşuluk operatörlerinin oransal azlığı başarılı sonuçlar elde edilmesini sağlamıştır.

EK 1-10 daki GSP sonuçları, hesaplama sürelerine göre incelendiğinde bütün test problemlerinde en uzun süreler KKO, en kısa süreler ise PSO sonuçlarına aittir. Önerilen hiyerarşik yöntemin hesaplama süreleri KKO dan az, PSO dan fazladır.

Dolayısıyla hiyerarşik yöntem KKO ya göre sonuçlar bakımından başarılı olduğu gibi, hesaplama süreleri yönünden de daha başarılı bir performans sergilemiştir.

Hiyerarşik yöntemin KKO dan iyi olmasının sebebi, KKO nun belirli bir süre kısa yollarda çok fazla feromon birikmesi sonucu durağanlaşma (stagnation) davranışı göstermesidir. Hiyerarşik yöntemde ilk etapta turları oluşturan KKO yönteminin zamanla durağanlaşmadan dolayı birbirine çok yakın sonuçları üretmesi problemi, PSO yönteminin sağladığı rastgelelik katkısı sayesinde turların geliştirilmesi ile ortadan kalkmıştır. Hiyerarşik yöntemin PSO dan daha iyi sonuçlar üretmesinin nedeni ise, PSO da başlangıç popülasyonları rastgele oluşturulurken, hiyerarşik yöntemde KKO tarafından üretilen anlamlı başlangıç popülasyonu PSO ya gönderilmesidir.

Sonuç olarak önerilen KKO-PSO hiyerarşik yaklaşımı, her iki algoritmanın standart versiyonlarında GSP test problemleri için ürettiği sonuçlardan daha iyi sonuçlar elde etmiştir. Geliştirilen hiyerarşik yaklaşım tez kapsamında ele alınan küçük ve büyük ölçekli GSP test problemlerinin tamamında başarılı olmuştur.

5.2 Öneriler

Önerilen bu yeni yöntem, GSP' nin yanı sıra NP-Hard problemleri kapsamında değerlendirilebilecek araç rotalama, çizge tabanlı kombinasyonel eniyileme problemleri, çizge renklendirme ve parçalama problemlerinin çözümü, fonksiyon optimizasyonu, çizelgeleme problemleri, dağıtık bilgisayar ağlarının tasarımı, dağıtık bilgisayar ağlarındaki kaynak ve dosya paylaşım problemleri ve benzeri ayrık optimizasyon problemi sınıfında değerlendirilebilecek gerçek dünya problemlerinin çözümünde uygulanabilir. Ayrıca, yöntem, GSP probleminden yola çıkarak problemin 3 boyutlu gerçek dünya problemlerinin çözümünde kullanılabilir.

Yöntemde çözüm zamanının kısaltılması ya da sonuçların iyileştirilmesi için çeşitli yardımcı yöntemlerin eklenmesi ile her iki algoritma yeniden düzenlenebilir.

Önerilen hiyerarşik yöntemin yanı sıra bu iki algoritmanın melez yapıda birlikte kullanılmasıyla GSP ve diğer problemlerin çözümleri test edilebilir. Karınca kolonisi yönteminin PSO dışında genetik algoritmalar, tavlama benzetimi, yapay arı kolonisi (ABC) gibi diğer yöntemlerle hiyerarşik ve hibrit yapıda ele alınarak kullanılması farklı çalışmalara esin kaynağı oluşturacaktır.

Önerilen hiyerarşik yaklaşımın sürekli fonksiyonlarda kullanılabilecek formu geliştirilerek benchmark fonksiyonları gibi sayısal optimizasyon problemlerinin çözümündeki başarısı irdelenebilir.

KAYNAKLAR

Arora, S. 1998. Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems., Journal of the ACM, 45-5, 753–782.

Bullnheimer, B. , Hartl, R. F. , ve Strauss, C. 1999c , "A new rank-based version of the Ant System: A computational study." Central European Journal for Operations Research and Economics, sayı 7 (1) , 25-28.

Boussaïd I., Julien L., Siarry P., 2013, A survey on optimization metaheuristics, Information Sciences, 237, 82–117.

Blum, C., Roli, A., 2003, Metaheuristics in combinatorial optimization: Overview and conceptural comparison, *ACM Computing Surveys*, 35, 268-308.

Chen S., Chien C., 2011, Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques, Expert Systems with Applications 38, 14439–14450.

Colorni, A., Dorigo, M. and Maniezzo, V., 1991, Distributed Optimization by Ant Colonies. In Proceedings of the European Conference on Artificial Life (ECAL'91, Paris, France), eds F. Varela and P. Bourgine. Elsevier Publishing, Amsterdam, pp. 134-142.

Colorni, A., Dorigo, M. and Maniezzo, V., 1992, An investigation of some properties of an ant algorithm. In Proceedings of the Second Conference on Parallel Problem Solving from Nature (PPSN II, Brussels, Belgium), eds R. Maenner and B. Manderick. North-Holland, Amsterdam, pp. 509- 520.

Cabrero D. Gomez, Armero C., Ranasinghe D.N.,2007, "The traveling salesman's problem: a self-adapting PSO-ACS algorithm", Second International Conference on Industrial and Information Systems, pp. 479-484.

Deng W., Chen R., He B. Liu Y. Yin L., Guo J., 2012, A novel two-stage hybrid swarm intelligence optimization algorithm and Application, 16:1707–1722

Dorigo, M.; Colorni, A. 1996, The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics–Part B, 26*(1), 1-13.

Dorigo, M. 1991, *Positive feedback as a search strategy.Technical Report.* Italy: Politecnico de Milano.

Dorigo, M. 1992, Optimization, Learning and Natural Algorithms.Ph.D. Thesis, Politecnico di Milano, Italy (in Italian).

Dorigo, M. ve Gambardella, L. M. 1997a, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", IEEE Transactıons On Evolutionary Computation, Vol. 1, No. 1, 53-66.

Dorigo, M. ve Gambardella 1997b, L. M., "Ant Colonies for the Travelling Salesman Problem", BioSystems 43, 73–81.

Dorigo M. , Maniezzo V., ve Colorni A, 1996, "The ant system: Optimization by a colony of cooperating agents". IEEE Transactions on Systems, Man, and Cybernetics–Part B, 26(1):29–41.

Dorigo M., Stützle T. , Ant Colony Optimization, 2004, A Bradford Book, The MIT Press Cambridge, Massachusetts, London, England, ISBN 0-262-04219-3.

Dong C. P., Anthony Figueras L., Chen C., 1994, "A Hierarchical Approach for Solving Large-scale Traveling Salesman Problem", Neural Networks. IEEE World Congress

Fan H., 2010, Discrete Particle Swarm Optimization for TSP based on Neighborhood, Journal of Computational Information Systems 6:10, 3407-3414

Gambardella, L.M. ve Dorigo, M. 1995, "Ant-Q: a reinforcement learning approach to the travelling salesman problem", In Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann, Palo Alto, California, USA, pp 252-260.

Gambardella L. M. and Dorigo M., 1996, Solving symmetric and asymmetric TSPs by ant colonies. In Proceedings of the IEEE Conference on Evolutionary Computation, ICEC96, IEEE Pres, 622–627.

Gündüz M., Kıran M. S., Özceylan E., 2014, "A hierarchic approach based on swarm intelligence to solve traveling salesman problem", Turkish Journal Of Electrical Engıneerıng & Computer Sciences, (In Press).

Hertz, A., Widmer, M., 2003, "Guidelines For The Use Of Meta-Heuristics In Combinatorial Optimization",European Journal of Operation Research,151,247-252.

Ilie S., Bădică C., 2013, Multi-agent approach to distributed ant colony optimization, Science of Computer Programming 78, 762–774.

Irnich S., Funke B., Grünert T., 2006, Sequential search and its application to vehicle-routing problems, Comput Oper Res 33:2405–2429.

Jun-man K.,Yi Z., 2012 Application of an Improved Ant Colony Optimization on Generalized Traveling Salesman Problem, 2012 International Conference on Future Electrical Power and Energy Systems, Energy Procedia Vol.17, 319-325,.

Jin-rong S., 2011, Improved Particle Swarm Optimization for Multiobject Traveling Salesman Problems, Seventh International Conference on Natural Computation pp. 1175,1179.

Karaboğa, D., 2011 Yapay Zekâ Optimizasyon Algoritmaları, Nobel Yayın Dağıtım.

Kennedy J. , Eberhart R., 1995 Particle swarm optimization, in Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ, pp. 1942–1948.

Keskintürk, T. ve Söyler, H., 2006 , "Global Karınca Kolonisi Optimizasyonu" , Gazi Üniv. Müh. Mim. Fak. Der., Cilt 21, No 4, 689-698.

Kıran, M. S., Gündüz, M., 2013, A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems, *Applied Soft Computing*, *13*(4), 2188–2203.

Kıran M. S., Gündüz M., Baykan Ö.K., 2012, A novel hybrid algorithm based on particle swarm and ant Colony optimization for finding the global minimum, Applied Mathematics and Computation, Vol. 219, 1515-1521.

Kuo R.J., SyuY.J., Chen Z., Tien F.C., 2012, Integration of particle swarm optimization and genetic algorithm for dynamic clustering, Information Sciences, Vol. 195, 124-140.

Laporte, G., Gendreau, M., Potvin, J. Y., Semet, F., 2000, "Classical And Modern Heuristics For The Vehicle Routing Problem", International Transactions in Operation Research, 7, 285-300.

Lee Z. J., 2004, A hybrid algorithm applied to travelling salesman problem, Networking, Sensing and Control, IEEE International Conference 1, 237-242.

Li L., Ju S., Zhang Y., 2008, Improved Ant Colony Optimization for the Traveling Salesman Problem, International Conference on Intelligent Computation Technology and Automation, 76-80.

Liao Y, Yau D., Chen C., 2012, Evolutionary algorithm to traveling salesman problems, Computers and Mathematics with Applications 64, 788–797.

Liao T., Stützle T. , Marco A, Dorigo M., 2014, A unified ant colony optimization algorithm for continuous optimization, European Journal of Operational Research, 234-3, 597–609.

Liu X., Su J., Han Y., 2007, An improved particle swarm optimization for traveling salesman problem, Lect Notes Comput Sci 4682:803–812.

Marinakis Y., Migdalas A. ve Pardalos P. M., 2005, A Hybrid Genetic-GRASP Algorithm Using Lagrangean Relaxation for the Traveling Salesman Problem, J. Comb. Optim. 10(4), 311-326.

Marinakis Y., Marinaki M., 2010, A Hybrid Multi-Swarm Particle Swarm Optimization algorithm for the Probabilistic Traveling Salesman Problem, Computers & Operations Research, 37-(3),432 – 442.

Öztürk, C., 2006, Karınca ve sürü optimizasyon yöntemlerinin incelenmesi ve yazılım uygulamalarının oluşturulması, Yüksek Lisans Tezi, *Marmara Üniversitesi Fen Bilimleri Enstitüsü*, İstanbul.

Pan Q-K, Tasgetiren MF, Suganthan PN, Chua TJ,2011, A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem, Inf Sci 181:2455–2468.

Sha D.Y., Hsu C-Y., 2006, A hybrid particle swarm optimization for job shop scheduling problem, Comput Ind Eng 51:791–808.

Stützle, T. ve Hoos, H. H., 2000, "Max-Min Ant System" , Future Generation Computer Systems, 16:889-914.

Shi X.H., Liang Y.C., Lee H.P., Lu C., Wang Q.X. 2007, "Particle swarm optimization based algorithms for TSP and generalized TSP", Information Processing Letters, Vol. 103, pp. 169-176.

Shelokar P.S., Siarry P., Jayaraman V.K., Kulkarni B.D., 2007, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, Applied Mathematics and Computation, 188, 129–142.

Socha K., Dorigo M., 2008, Ant colony optimization for continuous domains, European Journal of Operational Research 185, 1155–1173.

Shayeghia H. , Mahdavib, M. Bagherib A. 2010, Discrete PSO algorithm based optimization of transmission lines loading in TNEP problem, Energy Conversion and Management, 51-1, 112–121.

Szeto W.Y., Wu Y. , Ho S.C. , 2011, An artificial bee colony algorithm for the capacitated vehicle routing problem, European Journal of Operational Research 215, 126–135.

Takahashi R., 2009, "A hybrid method of genetic algorithms and ant colony optimization to solve the traveling salesman problem", International Conference on Machine Learning and Applications, pp. 81-88.

Tsai C.F., Tsai C. W., Tseng, C.C., 2004, "A New Hybrid Heuristic Approach for Solving Large Traveling Salesman Problem", Information Sciences, 166(1-4), 67-81.

Tuba M., Jovanovic R., 2013, Improved ACO Algorithm with Pheromone Correction Strategy for the Traveling Salesman Problem, Int J Comput. Commun., 8(3):477-485.

TSPLIB., 2014, *Traveling salesman problem benchmark problems* [online], http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/index.html [Ziyaret Tarihi: 16 Haziran 2012].

Uğur A., 2008, Path Planning On A Cuboid Using Genetic Algorithms, Information Sciences; 178: 3275-3287.

Yan X., Zhang C., Luo W., Li W., Chen W. ve Liu H., 2012, Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 6, No 2, 1694-0814.

Yang, Xin-She. (2010). Engineering Optimization. Wiley & Sons yayınları.

Yang, Xin-She. (2010). Nature-Inspired Metaheuristic Algorithms Second Edition. University of Cambridge,United Kingdom: Luniver Press.

Yun H., Jeong S., Kim K., 2013, "Advanced Harmony Search with Ant Colony Optimization for Solving the Traveling Salesman Problem", Hindawi Publishing Corporation, Journal of Applied Mathematics Volume 2013, Article ID 123738.

Wang C., Zhang J., Yang J., Hu C., Liu J., 2005, "A modified particle swarm optimization algorithm and its application for solving traveling salesman problem", International Conference on Neural Networks and Brain, pp. 689-694.

Wang K.P., Huang L., Zhou C.G., Pang W., 2003, Particle swarm optimization for traveling salesman problem, Int Conf Mach Learn Cybern 3:1583–1585.

White C. M. ve Yen G. G., 2004, A hybrid evolutionary algorithm for traveling salesman problem, Congress on Evolutionary Computation (CEC2004) 2, 1473-1478.

Weise, T., 2009, Global Optimization Algorithms, *Theory an Aplication*.

Wikipedia., 2014, *Metaheuristic* [online], http://en.wikipedia.org/wiki/Metaheuristic [Ziyaret Tarihi: 20 Nisan 2014].

Zhang J., Xiong W., 2009, An Improved Particle Swarm Optimization Algorithm and its Application for Solving Traveling Salesman Problem, Computer Science and Information Engineering, 612-616.

EKLER

EK-1 OLIVER30 için önerilen hiyerarsik yaklaşımla komşuluk operatörüne giren her bir popülasyon yüzdesi için bulunan sonuçlar
OBH: Ortalama Bağıl Hata Yüzdesi

	insert					**insert_sub**					**reverse**					**reverse_insert**				
Popülasyon Yüzdeleri	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
%10	423,74	425,65	423,90	0,04	46,87	423,74	424,90	423,93	0,04	49,00	423,74	424,69	423,88	0,03	44,65	423,74	424,90	423,85	0,02	47,48
%20	423,74	424,69	423,88	0,03	42,79	423,74	424,69	423,88	0,03	48,26	423,74	424,90	423,88	0,03	40,09	423,74	424,69	423,81	0,02	39,86
%30	423,74	424,69	423,85	0,03	38,57	423,74	425,82	423,86	0,03	36,37	423,74	424,69	423,87	0,03	35,28	423,74	424,12	423,78	0,01	36,13
%40	423,74	424,69	423,91	0,04	34,45	423,74	424,90	423,87	0,03	35,49	423,74	424,69	423,85	0,03	34,83	423,74	424,69	423,81	0,02	35,41
%50	423,74	424,69	423,89	0,04	34,32	423,74	423,74	423,74	***0,00***	34,24	423,74	424,12	423,78	0,01	34,04	423,74	424,90	423,86	0,03	34,35

reverse_swap					**swap**					**swab_sub**				
En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
423,74	425,65	423,84	0,02	47,32	423,74	424,90	423,88	0,03	49,90	423,74	423,74	423,74	***0,00***	41,08
423,74	424,90	423,80	0,01	43,21	423,74	424,64	423,80	0,02	40,12	423,74	424,90	423,84	0,02	38,48
423,74	424,90	423,85	0,03	36,09	423,74	423,74	423,74	***0,00***	35,32	423,74	424,90	423,86	0,03	36,83
423,74	424,69	423,82	0,02	34,83	423,74	424,69	423,84	0,02	35,13	423,74	424,90	423,87	0,03	36,40
423,74	423,95	423,78	0,01	34,73	423,74	424,69	423,84	0,02	34,44	423,74	424,69	423,80	0,01	35,18

EK-2 EIL51 için önerilen hiyerarsik yaklaşımla komşuluk operatörüne giren her bir popülasyon yüzdesi için bulunan sonuçlar
OBH: Ortalama Bağıl Hata Yüzdesi

	insert					**insert_sub**					**reverse**					**reverse_insert**				
Popülasyon Yüzdeleri	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
%10	428,87	433,20	430,33	0,34	139,54	428,87	433,89	430,66	0,42	117,00	428,87	431,11	429,66	***0,19***	117,93	428,87	434,44	430,42	0,36	117,35
%20	428,87	431,82	430,30	0,33	135,06	428,87	434,44	430,50	0,38	112,07	428,87	431,11	429,69	***0,19***	116,98	428,87	433,22	430,09	0,28	115,82
%30	428,87	434,33	430,57	0,40	133,07	428,87	432,73	430,48	0,38	109,72	428,87	436,18	430,36	0,35	114,01	428,87	432,49	430,22	0,31	115,47
%40	428,87	431,82	430,64	0,41	130,43	428,87	433,51	430,65	0,42	109,67	428,87	434,44	430,50	0,38	109,51	428,87	432,49	429,88	0,24	113,09
%50	428,87	433,73	430,69	0,42	124,49	428,87	434,44	430,67	0,42	108,11	428,87	433,98	430,22	0,31	103,38	428,87	432,19	430,00	0,26	112,30

reverse_swap					**swap**					**swab_sub**				
En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
428,87	434,44	430,51	0,38	123,12	428,87	432,51	430,22	0,31	114,81	428,87	434,60	430,09	0,28	124,63
428,87	433,89	430,39	0,35	120,68	428,87	434,44	430,27	0,33	112,10	428,87	432,51	430,25	0,32	120,05
428,87	431,82	430,50	0,38	111,43	428,87	434,82	430,18	0,31	107,63	428,87	432,83	430,46	0,37	118,19
428,87	432,51	430,02	0,27	110,93	428,87	434,44	430,26	0,32	103,99	428,87	432,51	430,11	0,29	117,03
428,87	432,51	430,31	0,34	109,06	428,87	434,44	430,28	0,33	102,21	428,87	433,89	430,25	0,32	106,38

EK-3 BERLIN52 için önerilen hiyerarsik yaklaşımla komşuluk operatörüne giren her bir popülasyon yüzdesi için bulunan sonuçlar
OBH: Ortalama Bağıl Hata Yüzdesi

	insert					**insert_sub**					**reverse**					**reverse_insert**				
Popülasyon Yüzdeleri	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
%10	7544,37	7548,99	7545,09	0,01	138,60	7544,37	7549,29	7545,09	0,01	135,69	7544,37	7548,99	7545,29	0,01	125,42	7544,37	7548,99	7545,29	0,01	127,62
%20	7544,37	7548,99	7545,29	0,01	135,36	7544,37	7548,99	7544,84	0,01	125,37	7544,37	7549,29	7545,10	0,01	121,04	7544,37	7548,99	7545,29	0,01	121,17
%30	7544,37	7548,99	7545,06	0,01	126,02	7544,37	7549,29	7545,31	0,01	119,93	7544,37	7548,99	7545,09	0,01	118,97	7544,37	7548,99	7544,84	0,01	120,97
%40	7544,37	7544,37	7544,37	***0,00***	123,37	7544,37	7544,37	7544,37	***0,00***	115,62	7544,37	7548,99	7545,08	0,01	117,80	7544,37	7548,99	7545,07	0,01	118,03
%50	7544,37	7544,37	7544,37	***0,00***	118,98	7544,37	7544,37	7544,37	***0,00***	113,05	7544,37	7548,99	7545,34	0,01	114,12	7544,37	7548,99	7544,84	0,01	113,19

reverse_swap					**swap**					**swab_sub**				
En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
7544,37	7548,99	7545,06	0,01	130,14	7544,37	7549,29	7545,37	0,01	128,13	7544,37	7548,99	7544,84	0,01	127,69
7544,37	7548,99	7544,83	0,01	128,83	7544,37	7548,99	7545,07	0,01	119,48	7544,37	7548,99	7545,29	0,01	126,33
7544,37	7548,99	7545,06	0,01	124,61	7544,37	7544,37	7544,37	***0,00***	116,16	7544,37	7549,89	7544,66	***0,00***	121,52
7544,37	7548,99	7544,86	0,01	120,37	7544,37	7544,37	7544,37	***0,00***	111,78	7544,37	7549,29	7545,32	0,01	116,87
7544,37	7548,99	7545,29	0,01	117,57	7544,37	7544,37	7544,37	***0,00***	110,17	7544,37	7548,99	7545,29	0,01	113,09

EK-4 ST70 için önerilen hiyerarsik yaklaşımla komşuluk operatörüne giren her bir popülasyon yüzdesi için bulunan sonuçlar
OBH: Ortalama Bağıl Hata Yüzdesi

	insert					**insert_sub**					**reverse**					**reverse_insert**				
Popülasyon Yüzdeleri	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
%10	677,12	684,74	682,12	***0,74***	209,50	680,58	683,97	683,36	0,92	204,41	679,61	686,32	683,36	0,92	207,15	680,58	685,63	683,11	0,89	214,51
%20	680,58	685,45	683,26	0,91	205,15	680,58	685,46	683,38	0,93	203,96	680,58	685,31	683,19	0,90	205,02	680,58	686,24	683,26	0,91	200,37
%30	679,61	685,45	682,45	0,79	204,98	680,58	685,45	683,42	0,93	203,20	680,58	686,49	683,14	0,89	204,96	680,58	685,55	683,30	0,91	195,40
%40	680,58	684,71	683,13	0,89	201,54	679,61	685,55	682,81	0,84	198,97	680,58	690,64	683,25	0,91	199,02	682,96	685,71	683,35	0,92	194,27
%50	677,57	684,45	682,33	0,77	201,27	679,61	684,74	682,49	0,79	190,72	679,61	684,62	682,72	0,83	195,26	680,58	684,74	683,25	0,91	192,59

reverse_swap					**swap**					**swab_sub**				
En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
680,58	685,45	683,10	0,89	206,34	679,61	686,32	683,23	0,90	204,99	679,61	685,63	683,18	0,90	204,21
680,58	688,12	683,29	0,91	204,92	680,58	697,36	683,32	0,92	204,62	680,58	685,45	683,25	0,91	203,34
680,58	687,74	683,27	0,91	198,83	680,58	693,69	682,99	0,87	201,99	680,58	687,41	683,29	0,91	197,37
680,58	685,22	683,33	0,92	197,99	680,58	686,16	683,28	0,91	195,91	681,50	685,72	683,30	0,91	196,95
680,58	686,64	683,28	0,91	191,50	680,58	684,74	683,13	0,89	195,70	679,61	688,91	683,24	0,90	191,67

EK-5 EIL76 için önerilen hiyerarsik yaklaşımla komşuluk operatörüne giren her bir popülasyon yüzdesi için bulunan sonuçlar
OBH: Ortalama Bağıl Hata Yüzdesi

	insert					**insert_sub**					**reverse**					**reverse_insert**				
Popülasyon Yüzdeleri	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
%10	545,39	555,15	549,22	0,70	238,05	544,39	554,47	549,37	0,73	238,95	545,39	554,47	548,93	0,65	241,62	545,20	552,13	548,50	***0,57***	238,05
%20	545,20	553,38	549,14	0,69	237,70	544,76	553,57	548,54	0,58	234,45	545,39	553,48	549,24	0,71	239,82	544,76	555,01	549,48	0,75	236,25
%30	544,67	554,29	549,27	0,71	232,59	546,52	551,78	549,39	0,73	230,33	545,39	554,73	549,23	0,70	236,97	545,17	551,78	549,02	0,66	231,38
%40	545,39	554,68	548,82	0,63	230,70	545,39	552,44	548,62	0,59	228,29	544,48	555,15	549,33	0,72	229,09	545,39	553,24	548,66	0,60	230,20
%50	545,39	551,83	549,43	0,74	225,32	545,39	554,10	549,49	0,75	218,69	545,39	552,44	549,50	0,75	221,69	544,76	554,72	549,36	0,73	218,85

reverse_swap					**swap**					**swab_sub**				
En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
544,48	551,23	549,11	0,68	240,41	544,39	553,74	549,37	0,73	240,65	546,19	552,97	549,21	0,70	241,76
545,39	553,64	549,38	0,73	236,24	545,39	553,25	549,21	0,70	236,62	546,52	552,30	549,48	0,75	235,09
545,39	553,74	549,46	0,75	233,12	544,39	554,73	549,16	0,69	237,53	545,39	554,68	549,06	0,67	231,38
544,67	554,29	548,61	0,59	226,90	545,28	554,20	549,02	0,67	234,50	544,48	554,21	549,40	0,74	231,18
545,39	553,55	549,42	0,74	224,22	545,39	554,47	549,16	0,69	233,50	544,48	552,50	549,48	0,75	230,38

EK-6 PR76 için önerilen hiyerarsik yaklaşımla komşuluk operatörüne giren her bir popülasyon yüzdesi için bulunan sonuçlar
OBH: Ortalama Bağıl Hata Yüzdesi

	insert					**insert_sub**					**reverse**					**reverse_insert**				
Popülasyon Yüzdeleri	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
%10	110987,28	112602,04	111353,72	2,95	257,93	110699,96	113124,17	111436,27	3,03	248,60	108200,69	112776,14	110826,09	***2,47***	247,92	110911,62	112911,74	111799,99	3,37	246,92
%20	110987,28	113238,41	111276,16	2,88	256,24	111138,11	113065,30	111754,14	3,32	244,81	110699,96	113016,10	111766,76	3,34	246,58	111128,14	113274,60	111808,36	3,37	241,89
%30	110773,39	112811,53	111404,32	3,00	255,40	111136,68	113458,24	111768,84	3,34	235,51	110773,39	112576,72	111700,99	3,27	241,05	111003,03	112988,49	111649,57	3,23	240,58
%40	110773,39	112652,88	111379,05	2,98	250,69	110699,96	112861,06	111766,92	3,34	235,33	110699,96	112945,61	111885,25	3,44	231,68	110699,96	113620,39	111866,30	3,43	235,89
%50	110773,39	112932,98	111392,33	2,99	245,72	110561,37	112845,36	111748,58	3,32	228,64	110699,96	112746,53	111857,05	3,42	228,67	110699,96	113065,92	111859,57	3,42	232,87

reverse_swap					**swap**					**swab_sub**				
En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
110755,68	113035,87	111785,74	3,35	244,05	111037,98	112587,70	111830,99	3,39	249,88	110711,16	112814,53	111633,18	3,21	250,22
111168,16	113274,56	111898,49	3,46	242,23	110896,78	112945,61	111807,45	3,37	243,57	110733,71	112558,65	111497,08	3,09	247,84
110699,96	112966,30	111808,09	3,37	242,11	110699,96	112817,48	111863,63	3,42	240,60	110699,96	112945,61	111889,75	3,45	246,63
110699,96	113217,78	111850,65	3,41	239,57	110699,96	113456,71	111859,83	3,42	240,02	110699,96	112961,38	111892,90	3,45	241,20
110699,96	113066,96	111884,59	3,44	237,23	110959,25	113065,92	111911,05	3,47	224,92	110699,96	113458,44	111900,22	3,46	226,95

EK-7 KROA100 için önerilen hiyerarsik yaklaşımla komşuluk operatörüne giren her bir popülasyon yüzdesi için bulunan sonuçlar
OBH: Ortalama Bağıl Hata Yüzdesi

	insert					**insert_sub**					**reverse**					**reverse_insert**				
Popülasyon Yüzdeleri	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
%10	21709,99	22019,07	21846,82	2,64	435,25	21594,41	22086,50	21826,80	2,54	424,41	21618,35	22141,77	21891,59	2,85	432,24	21594,41	22028,06	21820,93	2,52	423,21
%20	21594,41	22051,43	21888,76	2,83	427,25	21532,75	21984,23	21718,32	***2,03***	414,89	21566,82	22048,93	21888,10	2,83	430,11	21431,13	22086,24	21819,56	2,51	414,58
%30	21581,96	22169,70	21894,95	2,86	423,39	21594,41	21945,51	21787,69	2,36	410,57	21670,19	22084,76	21904,04	2,91	414,57	21670,19	22090,51	21904,01	2,91	410,95
%40	21603,36	22056,84	21880,31	2,79	417,41	21594,41	22063,34	21806,80	2,45	404,05	21584,17	22079,34	21882,42	2,80	411,88	21594,41	22100,67	21886,98	2,83	400,99
%50	21684,61	22127,05	21917,21	2,97	410,36	21594,41	21981,55	21797,52	2,41	399,87	21570,72	22137,82	21859,35	2,70	397,96	21804,51	22095,03	21898,29	2,88	400,01

reverse_swap					**swap**					**swab_sub**				
En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
21532,75	22043,80	21884,66	2,82	429,76	21594,41	22044,87	21862,50	2,71	430,28	21693,40	22162,02	21897,69	2,88	435,53
21542,57	22099,29	21908,56	2,93	415,48	21670,19	22077,77	21877,89	2,78	423,59	21434,62	22194,02	21867,35	2,73	430,73
21668,66	22048,93	21906,70	2,92	415,08	21594,41	22239,44	21903,49	2,90	409,55	21645,82	22086,24	21894,77	2,86	423,15
21651,89	22128,90	21898,57	2,88	408,86	21581,96	22046,85	21907,38	2,92	408,13	21594,41	22104,46	21899,61	2,89	421,51
21594,41	22193,35	21903,45	2,90	406,31	21594,41	22180,49	21910,54	2,94	400,66	21594,41	22394,06	21865,09	2,72	406,08

EK-8 EIL101 için önerilen hiyerarsik yaklaşımla komşuluk operatörüne giren her bir popülasyon yüzdesi için bulunan sonuçlar
OBH: Ortalama Bağıl Hata Yüzdesi

	insert					**insert_sub**					**reverse**					**reverse_insert**				
Popülasyon Yüzdeleri	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
%10	642,31	656,99	650,56	***1,28***	459,9	645,50	656,15	651,39	1,41	450,41	647,95	655,52	652,10	1,52	444,19	648,74	659,44	653,08	1,68	445,21
%20	643,55	659,82	651,78	1,47	447,3	648,25	656,58	650,85	1,33	447,05	645,35	667,36	652,20	1,54	438,79	646,71	658,94	652,19	1,54	436,85
%30	645,70	657,11	650,62	1,29	439,2	646,95	659,01	652,00	1,51	445,50	645,14	664,28	651,66	1,46	437,36	642,82	659,09	652,88	1,65	434,62
%40	644,94	658,78	651,19	1,38	426,0	647,95	661,34	652,02	1,51	433,32	647,95	661,53	652,91	1,65	428,11	642,82	659,79	653,03	1,67	416,54
%50	645,50	657,25	651,67	1,46	424,7	646,95	657,14	651,78	1,47	425,19	645,50	662,60	651,14	1,38	426,88	642,82	659,82	652,46	1,58	412,67

reverse_swap					**swap**					**swab_sub**				
En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
650,58	656,81	652,35	1,56	441,62	646,68	666,73	652,41	1,57	444,81	648,05	661,15	652,01	1,51	439,47
647,67	665,88	652,87	1,64	436,68	646,95	658,78	652,98	1,66	436,34	648,05	658,89	652,29	1,55	437,85
647,80	660,48	653,04	1,67	436,34	646,05	658,57	652,31	1,56	434,98	647,95	667,09	652,67	1,61	435,59
650,58	661,04	653,08	1,68	433,68	647,30	664,70	653,15	1,69	433,67	648,19	658,11	652,10	1,52	429,39
645,17	659,82	651,91	1,50	433,77	645,15	659,77	651,60	1,45	419,80	645,70	656,91	651,60	1,45	429,00

EK-9 CH150 için önerilen hiyerarsik yaklaşımla komşuluk operatörüne giren her bir popülasyon yüzdesi için bulunan sonuçlar
OBH: Ortalama Bağıl Hata Yüzdesi

	insert					**insert_sub**					**reverse**					**reverse_insert**				
Popülasyon Yüzdeleri	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
%10	6581,50	6629,71	6600,42	***1,04***	1186,15	6581,81	6659,70	6617,47	1,30	1074,00	6616,28	6626,97	6620,74	1,35	1067,49	6581,81	6638,74	6620,13	1,34	1150,34
%20	6581,50	6647,64	6617,81	1,31	1145,60	6595,25	6640,41	6617,65	1,31	1064,31	6616,28	6646,66	6620,63	1,35	1056,46	6581,81	6627,14	6618,76	1,32	1112,63
%30	6581,81	6627,35	6611,53	1,21	1092,76	6581,81	6639,58	6613,66	1,25	1052,94	6618,84	6626,97	6619,95	1,34	1053,20	6584,99	6627,14	6615,62	1,28	1085,62
%40	6584,99	6637,42	6617,33	1,30	1038,80	6581,81	6629,71	6619,46	1,33	1032,65	6608,07	6630,32	6619,51	1,34	1049,68	6608,65	6636,39	6620,12	1,34	1045,56
%50	6609,73	6628,52	6619,11	1,33	1011,88	6614,22	6626,97	6620,76	1,35	1032,53	6616,28	6627,14	6620,01	1,34	1041,33	6608,65	6636,89	6620,43	1,35	1005,72

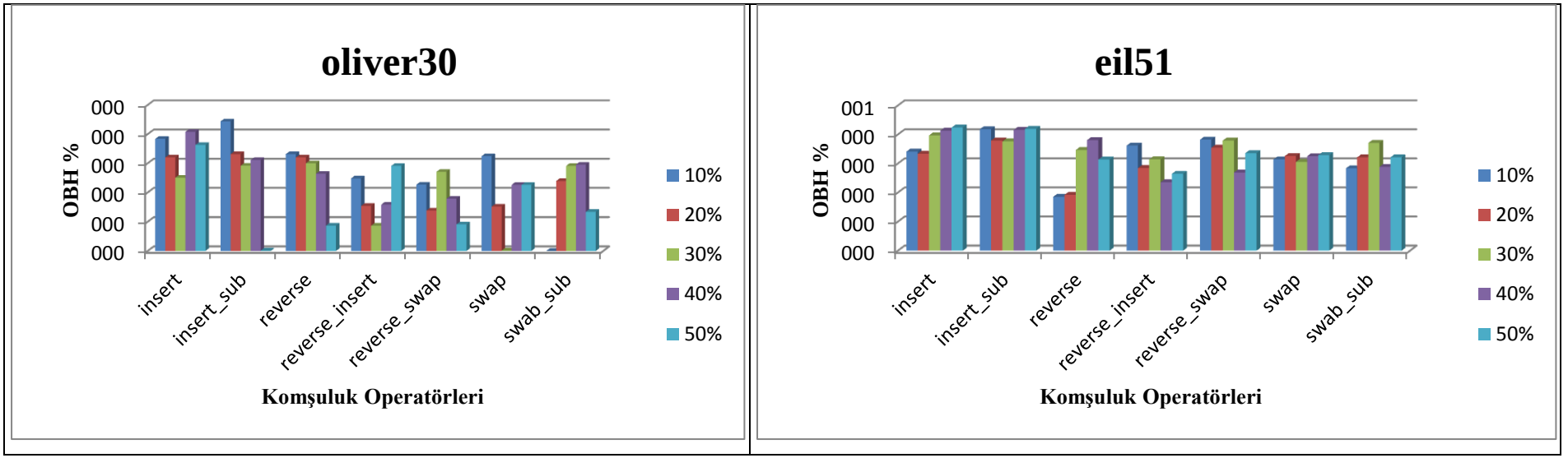
reverse_swap					**swap**					**swab_sub**				
En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
6598,74	6640,58	6618,62	1,32	1093,63	6615,71	6631,53	6620,38	1,35	1173,95	6583,19	6639,63	6616,93	1,30	1184,83
6601,92	6646,26	6618,25	1,32	1090,31	6576,02	6630,15	6619,12	1,33	1093,56	6583,19	6636,50	6620,36	1,35	1124,13
6581,81	6636,89	6618,98	1,33	1062,81	6581,81	6655,72	6614,91	1,26	1083,57	6584,99	6653,94	6616,15	1,28	1085,17
6581,81	6639,58	6620,57	1,35	1037,44	6616,28	6627,82	6620,86	1,36	1055,45	6584,99	6646,26	6619,47	1,33	1046,10
6581,81	6643,59	6618,28	1,32	1033,18	6584,99	6640,60	6619,94	1,34	1036,51	6581,81	6645,39	6619,82	1,34	1018,35

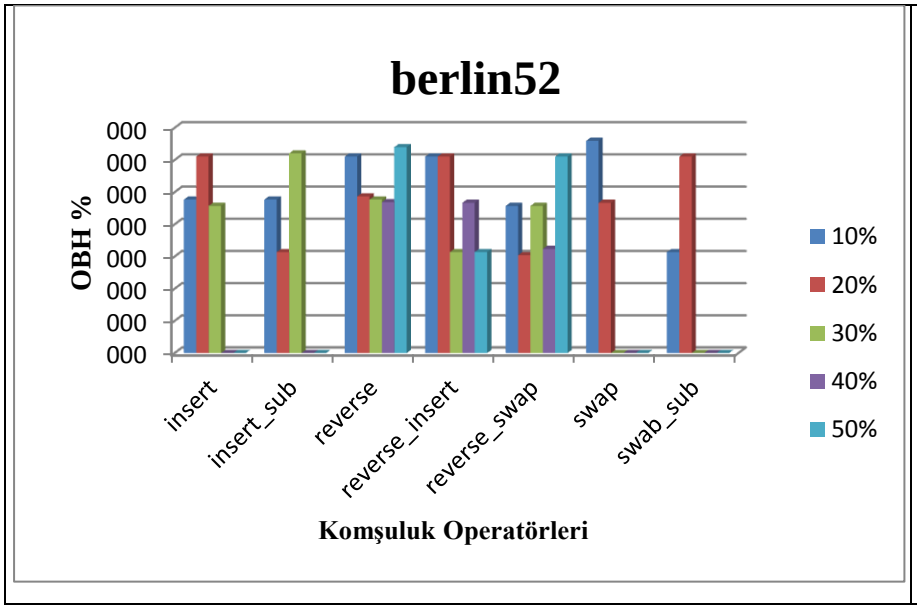
EK-10 TSP225 için önerilen hiyerarsik yaklaşımla komşuluk operatörüne giren her bir popülasyon yüzdesi için bulunan sonuçlar
OBH: Ortalama Bağıl Hata Yüzdesi

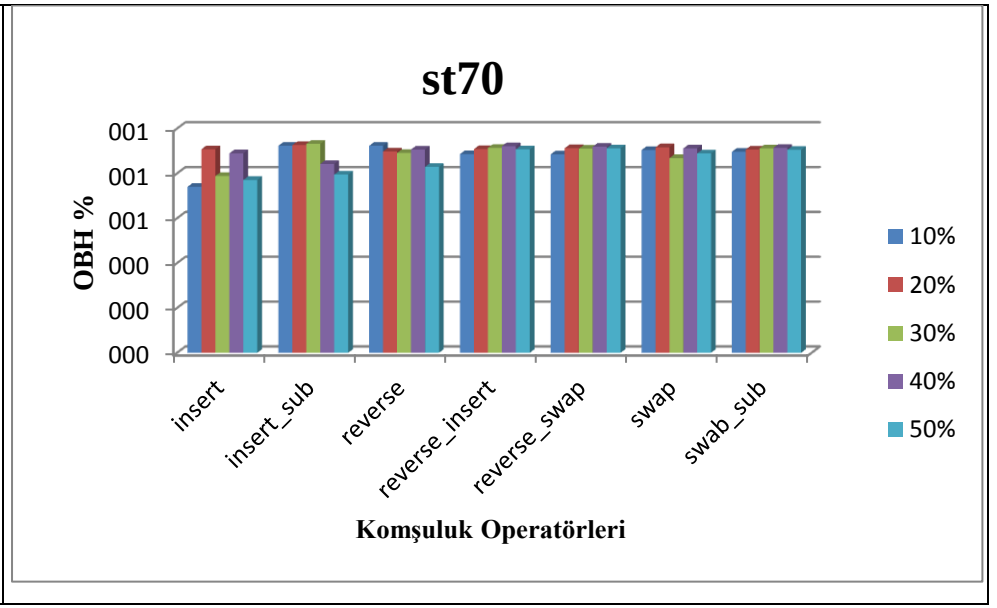
	insert					**insert_sub**					**reverse**					**reverse_insert**				
Popülasyon Yüzdeleri	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
%10	4055,33	4089,35	4065,24	5,34	2664,41	4052,08	4097,11	4073,16	5,55	2627,69	4066,94	4096,40	4072,96	5,54	2647,19	4058,41	4089,26	4068,54	5,43	2712,88
%20	4015,73	4103,27	4062,16	5,26	2655,36	4038,41	4079,03	4065,41	5,35	2608,97	4059,98	4096,85	4072,97	5,54	2617,76	4050,14	4083,71	4066,19	5,37	2678,86
%30	4046,38	4097,37	4073,05	5,55	2654,97	4059,45	4099,25	4073,21	5,55	2600,84	4009,75	4092,38	4051,74	***4,99***	2609,45	4059,98	4094,26	4071,94	5,52	2671,74
%40	4063,88	4097,61	4072,64	5,54	2654,60	4045,59	4096,73	4068,94	5,44	2596,78	4057,19	4094,40	4065,89	5,36	2601,77	4067,10	4087,94	4071,00	5,49	2600,46
%50	4066,08	4088,20	4073,03	5,55	2645,03	4048,45	4091,82	4067,20	5,40	2590,55	4057,19	4094,51	4073,11	5,55	2586,91	4058,32	4098,25	4071,18	5,50	2581,43

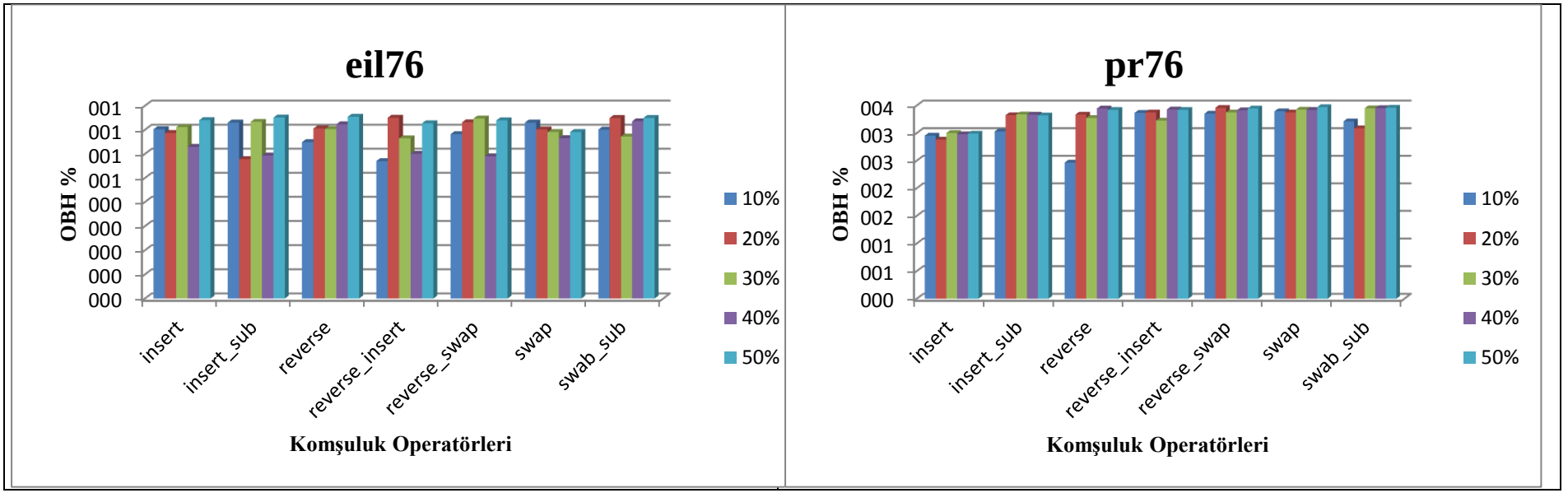
reverse_swap					**swap**					**swab_sub**				
En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)	En İyi	En kötü	**Ortalama**	OBH (%)	CPU Süresi (s)
4065,88	4087,09	4072,25	5,53	2668,64	4038,96	4081,02	4056,63	5,12	2637,53	4059,45	4089,11	4072,46	5,53	2524,14
4060,57	4098,34	4071,22	5,50	2655,01	4063,95	4089,19	4073,29	5,55	2593,28	4039,00	4090,47	4065,39	5,35	2515,46
4063,80	4090,05	4071,71	5,51	2650,20	4061,34	4087,82	4072,91	5,54	2576,17	4053,09	4100,00	4071,64	5,51	2507,80
4063,80	4088,90	4072,51	5,53	2637,52	4061,34	4093,95	4072,20	5,52	2575,62	4059,45	4087,93	4072,51	5,53	2505,65
4064,24	4086,64	4070,90	5,49	2627,35	4053,24	4090,15	4069,84	5,46	2561,03	4031,85	4098,85	4065,59	5,35	2502,35

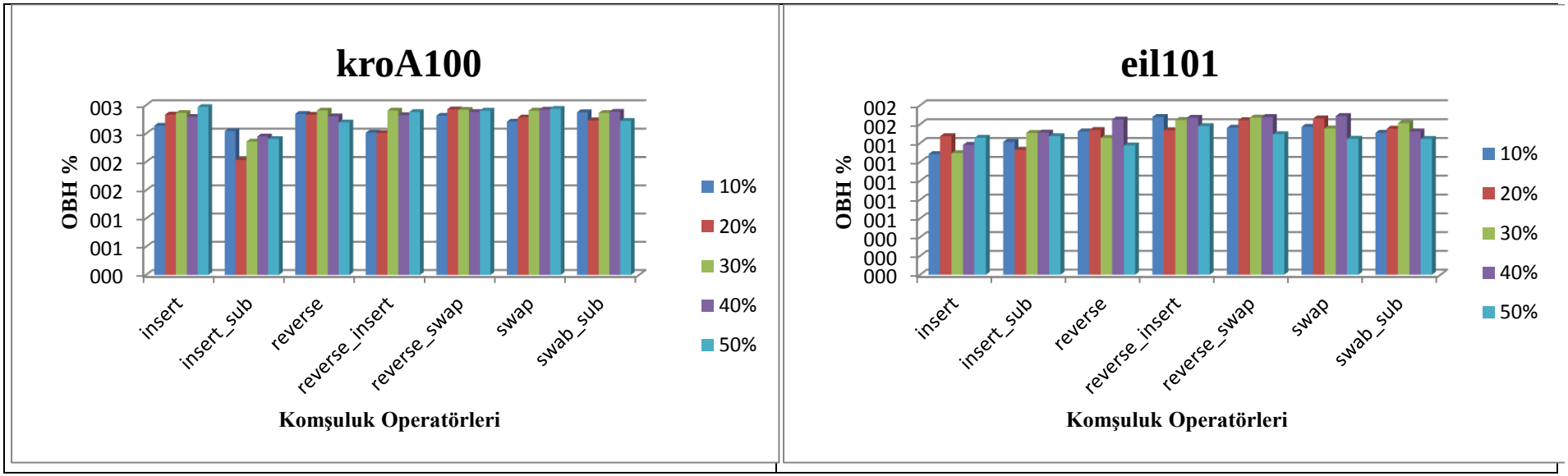
EK-11 TSP problemlerinin her bir komşuluk operatörünün belirtilen yüzdesindeki ortalama sonuçların bağıl hatalarının grafiksel gösterimi

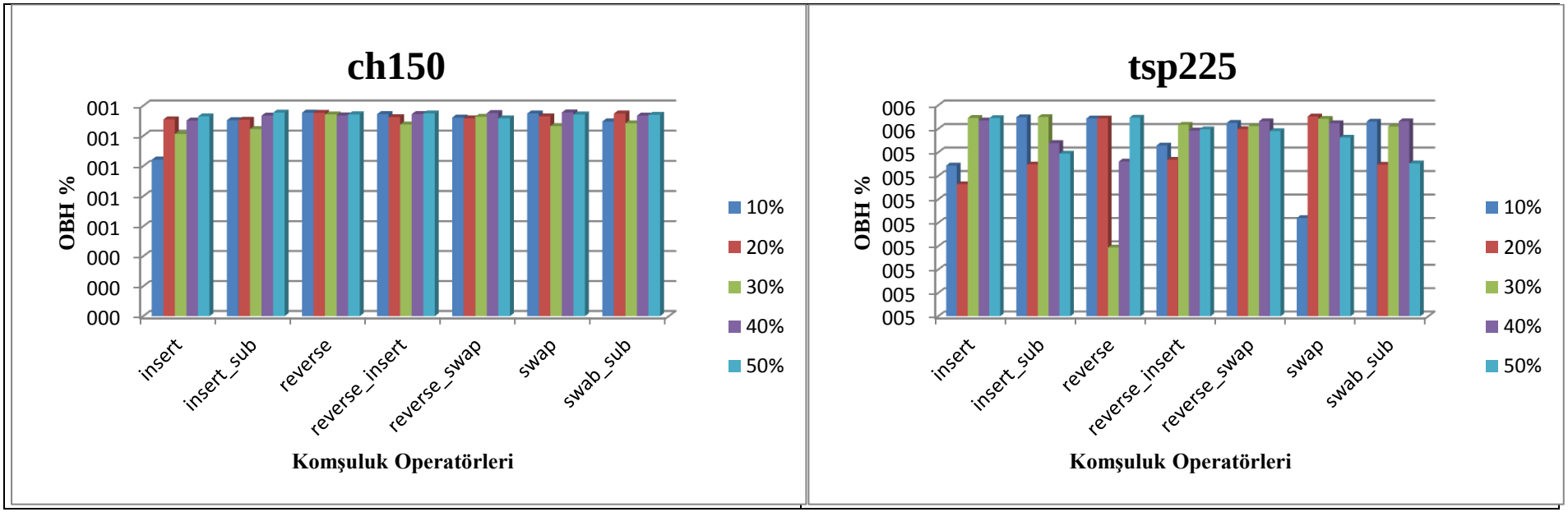



berlin52
OBH %
000
000
000
000
000
000
000
000
insert
insert_sub
reverse
reverse_insert
reverse_swap
swap
swab_sub
10%
20%
30%
40%
50%
Komşuluk Operatörleri


st70
OBH %
001
001
001
000
000
000
insert
insert_sub
reverse
reverse_insert
reverse_swap
swap
swab_sub
10%
20%
30%
40%
50%
Komşuluk Operatörleri


eil76
OBH %
001
001
001
001
000
000
000
000
insert
insert_sub
reverse
reverse_insert
reverse_swap
swap
swab_sub
10%
20%
30%
40%
50%
Komşuluk Operatörleri
pr76
OBH %
004
003
003
002
002
001
001
000
insert
insert_sub
reverse
reverse_insert
reverse_swap
swap
swab_sub
10%
20%
30%
40%
50%
Komşuluk Operatörleri


kroA100
OBH %
003
003
002
002
001
001
000
insert
insert_sub
reverse
reverse_insert
reverse_swap
swap
swab_sub
10%
20%
30%
40%
50%
Komşuluk Operatörleri
eil101
OBH %
002
002
001
001
001
001
000
000
000
insert
insert_sub
reverse
reverse_insert
reverse_swap
swap
swab_sub
10%
20%
30%
40%
50%
Komşuluk Operatörleri


ch150
OBH %
001
001
001
001
001
000
000
000
insert
insert_sub
reverse
reverse_insert
reverse_swap
swap
swab_sub
Komşuluk Operatörleri
10%
20%
30%
40%
50%
tsp225
OBH %
006
006
005
005
005
005
005
005
005
insert
insert_sub
reverse
reverse_insert
reverse_swap
swap
swab_sub
Komşuluk Operatörleri
10%
20%
30%
40%
50%

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Hüseyin ELDEM
Uyruğu : T.C.
Doğum Yeri ve Tarihi : Konya / 16.03.1981
Telefon : 0 554 572 27 81
e-mail : heldem@kmu.edu.tr

EĞİTİM

Derece		Adı, İlçe, İl	Bitirme Yılı
Lise	:	Konya İmam Hatip Lisesi, Selçuklu, KONYA	1999
Üniversite	:	Selçuk Üniv. Bilgisayar Mühendisliği Selçuklu KONYA	2005
Yüksek Lisans	:	Selçuk Ünv. Bilgisayar Mühendisliği A.B.D., Selçuklu, KONYA	2011-2014
Doktora	:		

İŞ DENEYİMLERİ

Yıl	Kurum	Görevi
2009-?	Karamanoğlu Mehmetbey Üniv. Bilgisayar Teknolojileri Bölümü, KARAMAN	Öğr. Gör.
2008-2009	Misal Ltd. Şti., KONYA	Bilgisayar Mühendisi
2006-2008	Bankasya Genel Müd., Yazılım Geliştirme Müdürlüğü, İSTANBUL	Yazılım Mühendisi / Sistem Analisti

UZMANLIK ALANI: Nesne Tabanlı Programlama (OOP), .NET Teknolojileri, Optimizasyon, Sürü Zekâsı.

YABANCI DİLLER: İngilizce

YAYINLAR

1- ELDEM, A., BASÇİFTÇİ, F., ELDEM, H., 2012, Application of Fuzzy Expert System for Cucumber Growing Irrigation, Advanced Science Letters, 15: 172–175
2- ELDEM A., BAŞÇİFTÇİ F., ELDEM H., 2013, Tomato Growing Model Using Fuzzy Expert System, Advanced Science Letters, ISSN: 1936-6612.
3- BALCI M., ÇELİK Y., ELDEM H., ELDEM A., 2013, Vocational Training in the European Union, Yükseköğretim ve Bilim Dergisi/Journal of Higher Education and Science, Vol. 3, Num. 1, pp: 38-47.

4- ÇELİK, Y., BALCI, M., PALALI, A., SOY, H., ELDEM, H., ELDEM, A., DİLAY, Y. (2012), "An Investigation of the Vocational Training in European Council", International Higher Education Symposium, Aksaray-Turkey, October 17-19 2012
5- ELDEM H., ÜLKER E., 2014, Application of Ant Colony Optimization for the Solution of 3 Dimensional Cuboid Structures, Journal of Computer and Communications, Vol. 2 No: 4, 99-107.
6- ELDEM H., ÜLKER E., 2014, Küre Üzerinde 3 Boyutlu Gezgin Satıcı Problemi Çözümünde Parçacık Sürü Optimizasyonu Uygulaması, 2014 Akademik Bilişim, Mersin.