T.C.
SÜLEYMAN DEMİREL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GSP ÇÖZÜMÜ BAŞARIMINI ARTIRMAK İÇİN HİBRİT SEZGİSEL ALGORİTMA TASARIMI

Raed Ashraf Kamil AL-BADRI

Danışman
Doç. Dr. Tuncay AYDOĞAN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ISPARTA - 2016

TEZ ONAYI

Raed Ashraf Kamil AL-BADRI tarafından hazırlanan **"GSP Çözümü Başarımını Artırmak İçin Hibrit Sezgisel Algoritma Tasarımı"** adlı tez çalışması aşağıdaki jüri üyeleri önünde Süleyman Demirel ÜniversitesiFen Bilimleri Enstitüsü **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**'nda **YÜKSEK LİSANS TEZİ** olarak başarı ile savunulmuştur.

Danışman	**Doç. Dr. Tuncay AYDOĞAN** Süleyman Demirel Üniversitesi	
Jüri Üyesi	**Doç. Dr. Ecir Uğur KÜÇÜKSİLLE** Süleyman Demirel Üniversitesi	
Jüri Üyesi	**Yrd. Doç. Dr. İsmail KIRBAŞ** Mehmet Akif Ersoy Üniversitesi	

Enstitü Müdürü **Doç. Dr. Yasin TUNCER** .....

TAAHHÜTNAME

Bu tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin referans gösterilerek tezde yer aldığını beyan ederim.

Raed Ashraf Kamil AL-BADRI

İÇİNDEKİLER

ÖZET

Yüksek Lisans Tezi

GSP ÇÖZÜMÜ BAŞARIMINI ARTIRMAK İÇİN HİBRİT SEZGİSEL ALGORİTMA TASARIMI

Raed Ashraf Kamil AL-BADRI

Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Tuncay AYDOĞAN

Karınca Koloni Optimizasyonu (Ant Colony Optimization-ACO) algoritmaları çoğu uygulamalı alanda ve belirli bir probleme ilişkin sonuçların optimize edilmesi amacıyla birçok gerçek yaşam probleminde kullanılır. Önemli gerçek yaşam problemlerinden birisi de Gezgin Satıcı Problemidir (GSP). Birçok algoritmanın özelliklede sezgisel algoritmaların kullanılarak çözüldüğü önemli bir problem olarak düşünülmektedir.

Bu araştırmada, GSP'de kullanılacak Genetik Algoritma (GA) ve MIN-MAX Karınca Sistemi (MIN-MAX Ant System-MMAS) Algoritmalarının güçlü yönleri melezleme işlemi ile birleştirilerek HGAMMAS adında yeni bir algoritma tasarlanmıştır.

HGAMMAS algoritması GA ve MMAS algoritmaları ile TSPLib (eil51, berlin52, eil76, rd100 ve kroA200) veri setleri kullanılarak denenmiştir. Yeni algoritmanın MMAS'e göre %3.2'ye, GA'ya göre %42.7'ye kadar daha düşük maliyette-daha iyi çözümü sunduğu görülmüştür. Ayrıca, HGAMMAS performansının eil51, berlin52, eil76 ve rd100 veri setlerinin literatürdeki "bilinen en iyi değerler" ile aynı sonuçları elde ettiği görülmüştür.

Anahtar Kelimeler: Üst sezgisel, ACO (Karınca Koloni Optimizasyonu), AS, ACS, MMAS, GA (Genetik Algoritma), GSP (Gezgin Satıcı Problemi).

2016, 74 sayfa

ABSTRACT

M.Sc. Thesis

HYBRID HEURISTIC ALGORITHM DESIGN TO IMPROVE THE PERFORMANCE OF THE TSP

Raed Ashraf Kamil AL-BADRI

Süleyman Demirel University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Assoc. Dr. Tuncay AYDOĞAN

Ant Colony Optimization (ACO) algorithms used in many real-life problems in order to optimize the results of most of the practical and specific problem areas. One of the important real-life problems is the Traveling Salesman Problem (TSP) .This is particularly so in the algorithm is considered as a major problem has been solved using heuristics.

In this study, to be used in TSP Genetic Algorithm (GA) and MIN-MAX Ant System (MIN-MAX Ant System-MMAS) algorithms combined with the strengths of the hybridization process is designed a new algorithm called HGAMMAS.

HGAMMAS algorithm tsplib G and MMAS algorithms (eil51, berlin52, eil76, rd100 and kroa200) were tested using data sets. The new algorithm is based on MMAS 3.2% ate gain by 42.7% up has been shown to lower the cost provide a better solution. Also, eil51 of HGAMMAS performance, berlin52, eil76 and the literature of rd100 data set "known best values" and it has been shown to achieve the same result.

Keywords: metaheuristic , ACO (Ant Colony Optimization), AS, ACS, MMAS, GA (Genetic Algorithm), TSP (Traveling Salesman Problem).

2016, 74 sayfa

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ-TABLE OF CONTENTS

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

GSP	Gezgin Satıcı Problemi (Travelling Salesman Problem)
GA	Genetik Algoritma (Genetic Algorithm)
ACO	Karınca Koloni Optimizasyonu (Ant Colony Optimization)
AS	Karınca Sistemi (Ant System)
ACS	Karınca Koloni Sistemi (Ant Colony System)
MMAS	MIN-MAX Karınca Sistemi (MIN-MAX Ant System)
HGAMMAS	Genetik Algoritma MIN-MAX Karınca Sistemi Melez Algoritma (Hybrid Genetic Algorithm MIN-MAX Ant System)

1. GİRİŞ

Optimizasyon yöntemleri geniş uygulama alanları ve hem teorik hem de pratik yaşam alanlarında birçok teknik kullanım ihtiyaçlarından dolayı oldukça dikkat çeken bir araştırma alanıdır.

Gezgin Satıcı Problemi (GSP), aralarında doğrudan yollar bulunan birden fazla düğüm (şehir) üzerinde, bir başlangıç düğümünden başlayarak başladığı düğüme dönerken bir düğümden bir daha geçmemek ve tüm düğümlere uğramak koşulu ile en kısa yolu, güzergahı hesaplayan bir önemli bir optimizasyon problemidir. Bu problem 1800'lerde matematikçi W. R. Hamilton ve Thomas Kirkman tafarından tanımlanmıştır (Saiyed, A., R., 2012).

Problemde düğümler arası mesafe maliyet olarak tanımlanarak en kısa yol en düşük maliyeti ortaya koymaktadır. GSP başta mühendislik ve istatistik olmak üzere hemen hemen tüm alanlardaki bazı problemlere uyarlanarak çözüm olmuştur. Geçmişten günümüze GSP için bir çok çözüm metodolojileri geliştirilmiştir. GSP, araştırmacıların geliştirdikleri yeni optimizasyon algoritmalarını sınayacakları bir araç olmuştur.

Son yirmi yıl içerisindeki sezgisel yaklaşım metodolojilerindeki artışlar bu problemin çözümüne de katkı sağlamıştır. Literatürde Genetik Algoritmalar, Yapay Sinir Ağları, Karınca Kolonisi, Arı Kolonisi, Swarm, Kanguru Algoritması yöntemleri ve bu yöntemlerden geliştirilen yöntemler en çok çalışılanlar olmuştur (Albayrak, M., Allahverdi, N., 2011) ve (Kuşcu, Ö., Küçüksille, E.U., 2011). Her yeni çalışma, maliyeti ve hesaplama süresini daha da azaltmayı hedeflemiştir.

Üzerinde çalışılan yöntemlerden birisi de mevcut algoritmaların birlikte kullanım olanaklarının araştırılarak melez(hybrid) algoritma tasarımları ile daha yüksek performanslar elde etmektir. Literatürde (Shi, X. H., Liang, Y. C., Lee, H. P., Lu, C., ve Wang, L. M., 2005), (Kao, Y. T., ve Zahara, E., 2008) ve (Merlot, L. T., Boland, N., Hughes, B. D., & Stuckey, P. J., 2002) gibi birçok melez

algoritma tasarımları ile odaklanılan problemlerin çözümlerinde performans artışları elde edilmiştir.

GSP için de (Zukhri, Z., ve Paputungan, I. V., 2013), (Takahashi, R., 2009) ve (Baraglia, R., Hidalgo, J. I., & Perego, R., 2001) gibi çalışmalar mevcuttur.

Bu çalışmada öncelikle, yaygın olarak kullanılan sezgisel algoritmalardan Genetik Algoritma (Genetic Algorithm-GA) ve Karınca Koloni Optimizasyonu Sistemi (Ant Colony Optimization-ACS)'nun çeşitleri olan Karınca Sistemi (Ant System-AS/ANT), Karınca Koloni Sistemi (Ant Colony System-ACS) ve MIN-MAX Karınca Sistemi (MIN-MAX Ant System-MMAS) algoritmalarının GSP çözümü için performansları araştırılmıştır. Daha sonra ise GA ile performansı en yüksek olan ACO algoritmasının melezlenmiş yeni bir tasarımı yapılarak GSP üzerindeki etkisi incelenmiştir.

Genetik Algoritma (Genetic Algorithm-GA), "evrimsel süreç" olarak adlandırılan evrendeki canlıların çoğalma ve yaşam süreçlerinin gözlenmiş veya kabul edilmiş evrelerinin matematiksel bir modelidir. John Holland tarafından 1975'de ortaya atılmıştır. Algoritma, problemin parametrelerini en iyi çözüme ulaştırmak için gen, kromozom, popülasyon, çaprazlama, mutasyon, seçim gibi biyolojik süreçleri ve operatörleri kullanır (Mastorakis, N, Bulucea, A, Tsekouras G., 2015).

Karınca Sistemi (Ant System-AS/ANT) Algoritması, Karınca Koloni Optimizasyonu (Ant Colony Optimization-ACO) algoritmalarının literatürdeki ilk olanı, temelidir (Dorigo, M., 1992) ve (Dorigo, M, Maniezzo, V. ve Colorni, A., 1991) Karıncaların yuvalarından ayrılarak, yuvalarına dönene kadar yiyecek bulmak için feromon adı verilen hormonu kullanarak gösterdikleri arama davranışlarının matematiksel bir modelidir. Algoritmada karıncaların yuva ile yiyecek arasındaki en kısa yolu bulma yöntemi taklit edilmektedir. Bu algoritmanın belirgin özelliği tur sonunda azalan feromon değerinin tüm karıncalar tarafından her tur sonunda güncellenmesidir.

Karınca Koloni Sistemi (Ant Colony System-ACS) Algoritması ile temel Karınca Sistemi algoritması üzerinde ilk önemli gelişme sağlanmış oldu (Dorigo, M., Maniezzo V. ve Colorni, A., 1996). Bu algoritmadaki önemli farklılık en iyi karıncanın feromon güncellemesine izin verilmesidir.

MIN-MAX Karınca Sistemi (MIN-MAX Ant System-MMAS) Algoritması da temel Karınca Sistemi algoritması üzerine kattığı özellikleri ile önemli gelişmeler sağlamıştır (Stützle, T. ve Hoos, H.H., 2000). Bu algoritmada da sadece en iyi karınca azalan feromonunu günceller ve feromonun en çok, en az değerleri sınırlıdır.

ACO'da kullanılan yapay karıncalar çözülen problemin durumu hakkındaki sezgisel bilgileri ve çalışma anında dinamik olarak değişen yapay feromon izleri dikkate alarak, araştırma deneyimi sırasında elde edilen yönleri yansıtmak için kısmi çözümlere çözüm öğelerinin iterasyonnarak eklenilmesi ile olasılığa dayalı bir şekilde çözüm oluşturan stokastik (olasılıksal) çözüm inşa süreçleridir.

ACO'da olasılıksal bir öğe karıncaların çeşitli farklı çözümler oluşturmasına ve böylece aceleci çözümlerdense daha çok sayıda çözümü araştırmasına olanak tanır. Aynı zamanda, birçok problem için hali hazırda mevcut bulunan sezgisel bilgi karıncaları en iyi çözüme doğru yönlendirmektedir. Daha da önemlisi, karıncaların araştırma deneyimleri algoritmanın gelecekteki iterasyonnmalarında çözüm oluşturmayı takviyeli öğrenme şeklinde (Sutton, S., ve Barto, A. G., 1998), etkilemek için kullanılabilir. Bunun yanında karınca kolonisinin kullanımı algoritmaya daha fazla sağlamlık verebilir. Birçok ACO uygulamasında bir problemin etkili bir şekilde çözülmesi için bir popülasyonun unsurlarının kolektif etkileşimine ihtiyaç duyulmaktadır. ACO algoritmasının uygulama alanı oldukça geniştir. Prensip olarak, ACO çözüm oluşturma mekanizmasının tasarlanabildiği herhangi bir ayrık optimizasyon problemi için uygulanabilir (Dorigo, M., Stützle, T,. 2009 ve 2003), (Solnon, C., ve Bridge, D., 2006), (Gambardella, L. M. al., 1998).

Buna ek olarak, yerel optimum çözümlerden kaçmak için; ya boş çözümden başlayarak tamamlanmış ve iyi bir çözüm inşa etmek için öğelerin eklendiği yapıcı sezgisel ya da tam çözümden başlayıp daha iyi bir çözümün elde edilmesi için bazı öğelerin iterasyonnan bir şekilde geliştirildiği yerel araştırma sezgisi kullanılır. Genellikle kontrol mekanizması, yerel komşu çözüm setlerinin rastgele hale getirilmesi ile tavlama benzetimi algoritması durumunda olduğu gibi (Kirkpatrick, S., Gelatt, C. D., ve Vecchi, M. P., 1983) veya tabu arama (F. Glover., 1989) veya farklı genetik (Holland, J. H., 1975) veya biyonomik (Maniezzo, V., Mingozzi, A., ve Baldacci, R.,1998) algoritma çözümlerinden alınan öğelerin birleştirilmesi ile daha iyi elde edilir (Z. Hlaing ve M. Khine., 2011) (Takahashi, R., 2009) ve (Al Salami, N. M. 2009).

Karınca çözümlerinin yerel değişiklikler bakımından optimal olması garanti değildir ve yerel arama yöntemleri kullanılarak geliştirilebilirler (Stützle, T. ve Hoos, H. H., 2000).

Bu araştırmada da arama döngüsünü mümkün olduğunca geliştirmek için MMAS ve GA arasında melezleştirme yoluyla en iyi global çözümleri veya global çözüme yakın olan çözümler bulunacaktır. Yöntem olarak, MMAS'de jenerasyona girdi olarak genetik algoritmada kromozom başlangıç fonksiyonunu kullanılarak melezleştirme yapılacak, bu da en düşük maliyet olan en iyi çözümleri bulmakta yardımcı olacaktır.

Bu araştırmada, GSP'ni GA ve MMAS algoritmalarından daha düşük maliyette çözebilecek melez yapıda, HGAMMAS adında yeni bir algoritma tasarlanması amaçlanmıştır. Bu amaç doğrultusunda;

- Gezgin Satıcı Optimizasyon Problemini tanımak,
- Sezgisel algoritmalardan Genetik Algoritma ve Karınca Koloni Optimizasyonunu tanımak,
- Karınca Sistemi, Karınca Koloni Sistemi ve MIN-MAX Karınca Sistemi algoritmalarını tanımak,
- Bu algoritmaları Java platformunda kodlayarak ortak bir arayüzde birleştirmek,

- Bu algoritmaları aynı veri setlerinde test ederek performanslarını sınamak,
- Genetik Algoritma ve MIN-MAX Karınca Sistemi Algoritmalarını kullanarak melez yenibir algoritma tasarlamak,
- Yeni algoritmanın performansını uluslararası literatür doğrultusunda değerlendirmek hedeflenmiştir.

Tezin ikinci bölümünde konu kapsamında amaç ve hedeflere katkı sağlayan literatür özeti verilmiştir. Üçüncü bölümde materyal ve yöntem bilgisi, dördüncü bölümde de çalışmada elde edilen bulgular literatür ışığında değerlendirilmiştir.

2. KAYNAK ÖZETLERİ

Gezgin satıcı probleminin (Jünger, M., Reinelt, G., ve Rinaldi, G., 1995) çözümünü araştırmak için karınca kolonisi optimizasyonunun (ACO) uygulandığı bir çok araştırma yapılmıştır. Yapılan bir araştırmada yazarlar GSP hesaplamalarında oldukça etkili olduğunu ispatladığı için ACS' yu seçmişlerdir (Hlaing, Z. C. S. S., ve Khine, M. A., 2011). Bu araştırmada, GSP'yi çözmek için bir algoritma sunulmuştur ve denemeler TSPLIB (Reinhelt, 2014)'den alınan ölçütlere göre yürütülmüştür. Sonuç olarak sunulan algoritma böyle problemlerde ACS uygulanarak daha iyi sonuçlar sağladığını göstermiştir, böylelikle elde edilen sonuçlar oldukça tatmin edicidir. Algoritma geliştirilmiş bir ACS versiyonunu kullanmakta ve lokal araştırma entropisine dayanmaktadır.

Bir makalede (Mishra, R., ve Jaiswal, A., 2012), Bulutta Yük Dengeleme problemini çözmek için ACO'yu etkili bir yöntem olarak sunmuştur. Bulutta Yük Dengeleme- olan ana problemin bir tanımını internet ağları arasında hesaplama stili olarak yükleri yönetmedeki faydaları ve arızaların üstesinden gelmesi ile tanımlamış ve bulut içerisinde güvenlik açıkları ile birleştirmiştir. ACO'nun bilgisayar mühendisleri tarafından karıncaların kendi feromonlarını kullanarak güzergah oluşturma davranışlarına benzer bir şekilde kurdukları ve feromon tablosu adını verdikleri yöntem üzerinden en kestirme yolu bulmada kullanıldığı ifade edilmiştir. Makalede bellek, gecikme zamanı, CPU yeteneği ve bulut yükleri gibi bulut içerisindeki birçok açıdan performansı arttırmak veya azaltmada etkili ve verimli gelişimler sağlamak için sezgisel ACO'dan faydalanılmıştır.

Karınca kolonisi optimizasyonu (Dorigo ,M., ve Stutzle, T., 2009) tarafından da ele alınmış ve ileri uygulamalar tartışılmıştır. Makalede ACO'nun uzun zamandır kullanılan bir yöntem olduğu, bu konuda yapılan ilk çalışmanın 1991 yılında yapıldığı ve o tarihten beri ACO'nun dikkat çekici bir konu haline geldiğinden ve sonuçların gerçekte gösterildiğinden bahsedilmiştir. ACO hakkında yayınlanmış olan birçok çalışma günümüzde zorlayıcı olarak kabul edilen hesaplama

problemlerini çözmeyi amaçlamıştır. Aslında bu çalışmaların büyük çoğunluğu teorik düzeyde bulunmaktadır fakat çok sayıda çalışma ise bunların gerçek yaşam problemlerine pratik olarak nasıl uygulanacağını ele almıştır. Örneğin, Ant Optima (www.antoptima.com) isimli bir şirket ACO'nun zaman değişkenli veri ve çoklu hareket parçacıkları gibi uygulamaları içeren gerçekte kullanımını desteklemektedir.

Bu çalışmaya göre, ACO uygulamaları gittikçe artan bir oranda dikkat çekmektedir böylelikle ''iyi yapılandırılmamış'' problemlerin ACO ile çözülmüş olması tahmin edilebilir bir durumdur, bu da ACO nun ileri avantajlar sağladığını daha kuvvetli bir şekilde kanıtlamaktadır. Makale ayrıca sadece lokal bilgilerin bulunduğu durumlarda ulaşılabilir veri eksikliği yüzünden ACO algoritmalarının oldukça dinamik doğası bulunan problemlerde etkili bir şekilde uygulanabilir olup olmadığını konusunda kesin bir sonucun bulunmadığından bahsetmektedir.

ACO için algoritma geliştirmeye yardımcı olan bir çeşit danışma rehberi tarafından ACO şematik bloklarına ek olarak optimize edilmiştir (Monteiro ,M.D ve Fontes. F.,2012).

Karınca Kolonisi Algoritması Sözde-Kodu

1: Parametreleri başlat

2: Feromon izlerini başlat

3: Karıncaları yarat

4: while durdurma kriteri karşılanmadı do

5: bütün karıncaların çözümlerini inşa etmelerine izin ver

6: feromon izlerini güncelle

7: Artalan sürecine izin ver

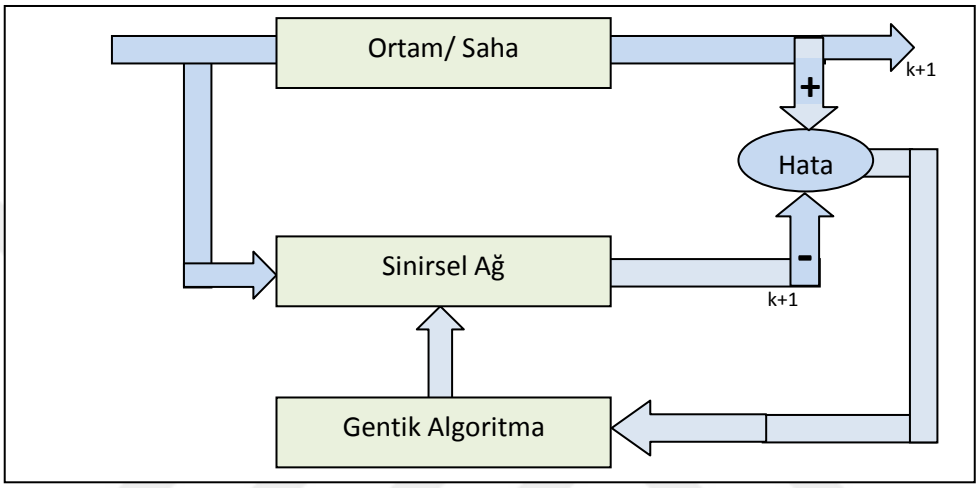
8: end while (sonlandır)

Çalışma aynı zamanda optimizasyon problemlerini NP-zor problemler olarak tanımlamıştır, fakat tam olarak çözülebilecek bir olasılık söz konusudur, hatta sezgisel yaklaşımlar bile kullanılır, ACO kombinasyonal problemler şeklindeki problemleri çözmek için tanımlanmıştır, bununla birlikte çalışmada oluşturulan algoritmalar çok çeşitli problemleri çözmekte kullanılmış fakat kombinasyonal problemlere odaklanmıştır. Bu çalışma ile elde hedeflenen amaç, böylesi problemlerin çözümünde en uygun metoda ulaşmak için ACO özelliklerine dayanan alternatif çözüm yöntemleri sunmaktır.

ACO ile karşılaştırıldığında çalışmamızda kullandığımız başka bir teknik ise Genetik Algoritmadır (GA). Bu konu (Joshi. G.,2014) tarafından da ele alınmıştır. GA bir optimizasyon kriteri olarak değerlendirilmiş ve GA'ya dayanan birçok teknik optimizasyon hedefleri için sunulmuştur. Makalede GA'nın biyolojik süreçlerden ilham aldığı ifade edilmiş, genel kural olan ''en uygun olanın hayatta kalmasına'' vurgu yapılmıştır. Yazar optimizasyon için GA'yı uygulamanın sekiz gerekli adımından bahsetmiştir. Bu adımlar şunlardır (Kodlama, Başlatma, Değerlendirme, Seçilim, Çaprazlama, Mutasyon, Tekrar ve Kod Çözme). Performansını genetik davranışlardan esinlenerek biyolojik süreçler ve diğer süreçler için uygulanan genel bir algoritma vardır, uygun ortamlarda GA kriterlerini uygulanır. Bununla birlikte, makale bir çeşit karşılaştırma olması için tepe tırmanma, karınca kolonisi optimizasyonu ve benzetilmiş tavlama gibi bilinen diğer optimizasyon çeşitlerini de ele almıştır. Makalede genetik algoritmaların gelecekte daha fazla geliştirileceği ve diğer başka yeni optimizasyon algoritmaları ile birlikte karşılaştırılacağı vurgulanmaktadır.

Daha önceden bahsedildiği gibi, bu makaledeki metodolojinin bir parçası ACO ile Yapay Sinir Ağı (YSA) ve Genetik Algoritma (GA) gibi diğer teknikler elde edilen sonuçların değerlendirilmesi ve bu sonuçlar arasında bir karşılaştırma yapmaktır; aslında çok sayıda çalışma YSA ve GA'yı ele almak için veya bunların uygulanma süreçlerini incelemek için yapılmıştır. Bu iki tekniğin temel olarak bilgisayar mühendisleri ve yapay bilimler alanında uzmanlaşmış kişiler tarafından kullanıldığını ve geliştirildiğini ifade etmek gereklidir (Vishwakarma .,H. D., 2012). Vishwakarma (2012) hem YSA hem de GA'yı ele almış, GA'yı

ağırlık (YSA ağı içerisinde bütün nöron bağlantıları için ağırlık) optimizasyonu için kullanılan bir teknik olarak sunmuştur. Aynı zamanda düzensiz oluşumlu ağ uygulamalarında ve MANETler (Mobil Düzensiz Oluşumlu Ağ Uygulamaları) içinde yönlendirme protokollerinde kullanılmaktadır. Aşağıda verilmiş olan diyagram GA ve YSA'yı birleştiren basitleştirilmiş algoritma şemasını göstermektedir.



Şekil 2.1. GA ile YSA ağının bağlandığını gösteren blok diagramı

Bu makalede, denemeler YSA'nın GA ile bağlanarak ve bağlanmadan yapılmıştır, denemelerden bir tanesi (saniyeler ile ölçülen) hello interval üzerinde yapılmıştır. Makale bu farklılıkların tatmin edici olduğunu göstermiş ve GA'nin ağ performansını düğüm ağırlıklarını optimize ederek arttırdığını kanıtlamıştır.

(Putha, R., Quadrifoglio, L., ve Zechman, E., 2012) tarafından yapılan bir diğer çalışma ise genetik algoritmaların karınca kolonisi optimizasyonu ile uygulanmasını araştırmıştır. Bu makale ''Aşırı Doygunluk Koşulları Altında Trafik Sinyal Koordinasyonu'' problemine optimize edilmiş bir çözüm sağlamayı amaçlamıştır. ACO burada ihtiyaç duyulmayan tıkanma ortadan kaldırarak ve kuyruk dağılımını dikkate alarak performansı arttıran akıllı zamanlama planı bulmak için kullanılmıştır. Zamanlama bu çalışmada optimize edilmiştir çünkü bu tarz problemler trafik ağları ile ilişkilendirilmiştir ve doygunluk periyodları arasında bütün kavşaklar için yeşil zamanları kullanan aralıklı zaman ağlarıdır.

(Mavrovouniotis, M., ve Yang, S., 2015) tarafından yapılan araştırmada örüntü sınıflandırma problemlerinde optimizasyon algoritmasının uygulanması amaçlanmıştır. Genellikle örüntü sınıflandırması Feed-forward (ileri besleme) sinirsel ağları ile yapılır. Doğruluk genellikle süreç konfigürasyonunun seçimine dayanmaktadır. Daha iyi bir performans elde etmek için, kullanılan algoritmaya karınca koloni optimizasyonu eklenmiştir. Bu makalede ayrıca örüntü sınıflandırma araştırması hibrit model eklenerek ve eklenmeden yapılmış, bir çok ölçütlere göre sonuçlar her iki durum içinde değerlendirilmiş ve birlikte karşılaştırılmıştır. ACO performansı ile karşılaştırma elde edilmesi için diğer sürü zeka türleri kullanıldığında böyle problemler için performans tanımına makalede yer verilmiştir.

Elde edilen sonuçların istendik analizi yapıldıktan sonra, yazarlar ACO tekniğinin böylesi uygulamalar için ve BP için tatmin edici değerler elde edilmesi için başarılı bir seçim olduğunu göstermiştir.

Sonuç olarak eğitim algoritması olarak kullanıldığı zaman bağımsız ACOR durumu bağımsız ACO eğitim durumundan daha iyi performans göstermiş, diğer yandan bu problemlerde büyük sayılardaki vakalara uyarlandıkları zaman hibrit ACO-BP daha iyi bir performans ortaya koymuştur. Dereceli azaltma (Gradient Descent) teknikleri kullanıldığında, problemin boyutu büyüdükçe düşüş göstermiştir, bu durum ACO-BP ile karşılaştırıldığında da geçerlidir.

Dereceli azaltma tekniklerinin bağımsız üst-sezgisel eğitim modellerinden (ACO eğitimi de dahil olmak üzere) daha iyi sonuç verdiği gösterilmiştir. Bunun nedeni kullanılan dereceli azaltma modellerinin doğruluğunun üst-sezgisel yöntemlerin doğruluğundan daha iyi olmasıdır.

Buna ek olarak, ACO diğer üst-sezgisel metotlar ile karşılaştırıldığında görece daha yüksek bir performans göstermiştir. Bununla birlikte, bu çalışmada çeşitli üst-sezgisel yöntemlerin birçok problem durumu üzerinde tatmin edici bir şekilde uygulanabildiği gösterilmiştir. Genel bir tanımlama olarak, yazarlar model sınıflandırma optimizasyonu için özellikle eğitimde melezleştirilmiş

dereceli azaltma tekniği ile ACO üst-sezgisel metotların sinirsel ağ eğitiminde kullanılabileceği sonucuna varmıştır. Gelecekteki çalışmalar için, yazarlar bu çalışmada elde ettikleri etkili ve verimli sonuçlarına dayanarak eğitimi içeren modellerde ACO'nın dinamik çevrelerde uyarlanabileceği beklentisindedir.

Son olarak, (Mahajan, R., ve Kaur, G., 2013) tarafından yapılan bir çalışmada sinirsel ağ ve genetik algoritmaların uygulamasını göstermiştir. Bu yöntemler bu araştırmada temel olarak kullanılan (ACO) teknik ile karşılaştırma ve araştırılmıştır. Makale ilk olarak her iki algoritmanın da genel kavramlarını açıklamıştır. Adlarında da anlaşılabileceği gibi birinci yöntem olan sinirsel ağ yaşayan canlılardaki sinir sisteminden ilham almıştır. Sinir bağlantıları ve aralarındaki iyon geçişleri simüle edilmiş ve birçok algoritma buna dayandırılmıştır.

Diğer tür ise genetik olan biyolojik kaynaktan esinlenmiştir (Bar-Cohen, Y., 2005), DNA ve RNA değişim olasılıklarına dayanmaktadır. Makalede hem genetik hem de sinirsel algoritmalar kullanılarak Gezgin satıcı problemi (GSP) çözümünde esnek bir yöntem geliştirilmiştir. Bu çözüm algoritma uygulamalarını maksimal performans yaklaşımını verecek ve maliyette dokunulabilir bir azaltma sağlayacak şekilde verilmiştir. Son olarak, makale bu teknikleri birçok girdi ve tahmini bir çıktı ile kara kutu olarak tanımlamış ve birçok uygulama alanı olduğundan bahsetmiştir. Yazarlar bazı kısıtlamalar olmasına rağmen, gelecekte bu kısıtlamaların çözülebileceğini ve üstesinden gelinebileceği beklentisi içerisindedir.

Temel özellik en iyi çözümü elde edebilme yeteneğidir. Daha geniş ölçekli problemler sezgisel algoritmalar kullanılarak çözümlenebilmektedir. Elde edilen çözümler genellikle (sub-optimal) yaklaşık optimal çözümler veya global çözümlere yakın optimal çözümlerdir. Üst-sezgisel (Moscato, P., and Cotta, C., 2003) yöntemler rastgele olmayan (deterministik) ve rastgele (stokastik) yapıyı birleştiren sezgisel algoritmaların en önemli parçası olarak görülmektedir. Yaygın olarak kullanılan sezgisel algoritmalar şunlardır; ''Parçacık Sürü Optimizasyonu (PSO)'', ''Diferansiyel Evrim (DE)'', ''Karınca Kolonisi

Optimizasyonu (ACO)'', ''Yapay Sinirsel Ağlar (YSA)'' ve ''Genetik Algoritmalardır (GA)''.

Melezleştirme sürecinde ACO algoritması ve Genetik Algoritmanın melezleştirilmesine dayanan bir takım işlemler söz konusudur. Hibrit algoritması yeni bir algoritma olarak GSP probleminin çözüm süreci içerisinde kullanılmıştır.

GA'nın bu tür problemlerde kullanılmasının temel amacı ACO'daki feromon miktarını içeren matrisi başlatmak ve ACO'dan gelen rotayı yeniden yapılandırmak ve yeniden şekillendirmektir. Önceki çalışmada, yazarlar sağlamış oldukları Hibrit algoritmanın bir yanda ACO versiyonlarından biri diğer yandan GA ile karşılaştırıldığında daha stabil ve etkili olduğunu iddia etmişlerdir (Glover, et., 2006).

İki ana yakın algoritma olarak geliştirilen bir başka algoritma ise AS ve GA 'ya dayanmaktadır. Yazarlar melezlemenin her iki algoritmanın da zayıf yönlerinin üstesinden geldiğini ileri sürmüşlerdir.Bu çalışmada ACO kullanımı geçersiz güzergahların kaçınılması açısından yardımcı olmuştur. Diğer yandan GA-AS feromon matrisi bağımlılığı probleminin üstesinden gelinmesi amacıyla kullanılmıştır (Shang, G., Xinzi, J., ve Kezong, T., 2007) ve (MITRAS, B., & ABOO, A. K., 2014).

Melezleme sağlayan, Shang ve arkadaşları tarafından yapılan bir başka çalışmada ise, Memetik Algoritma (MA) kullanan ACS içerisindeki parametrelerin bir kombinasyonunun bulunduğu ileri sürülmüştür.

Önerilen melezleme çözüm üreten alt çözüm oluşturucu olarak her algoritmadaki temel metotları içermektedir. Bundan sonra, her alt çözüm en uygun ve stabil olanın seçilmesi süreci ile devam etmedir böylelikle bu alt çözüm süreç içerisindeki bir sonraki tekrariçin yeni popülasyonu oluşturmaktadır. bu süreç durma koşuluna ulaşılıncaya kadar geliştirilmektedir (Duan, H., ve Yu, X., 2007).

3. MATERYAL VE YÖNTEM

Bu bölümde, tez çalışmasının ana araştırma konusu olan GSP'nin en düşük maliyette çözülmesi için tasarlanan bir melez (hibrit) algoritmanın tasarım aşamaları anlatılmaktadır.

Tasarlanan algoritmada literatür taraması analizi sonunda Genetik Algoritma ve Karınca Koloni Optimizasyonu yöntemlerinin kullanılması uygun bulunmuştur. Karınca Koloni Optimizasyonu yöntemi temelinde Karınca Sistemi (Ant System-AS veya ANT) algoritması ve Karınca Koloni Sistemi (Ant Colony System-ACS) ile Min-Max Karınca Sistemi (Min-Max Ant System-MMAS) algoritmalarından meydana gelmektedir. AS temel teori üzerine yapılan ilaveler ile zaman içerisinde güçlendirilmiştir.

Çalışmada önce GA ile hangi karınca koloni optimizasyon algoritması kullanılacağı bu algoritmaların performanslarına bakılarak MMAS algoritmasına karar verilmiştir. Daha sonra da GA ile MMAS algoritmaları melez bir kullanım ile HGAMMAS adında yeni bir algoritma tasarlanmıştır. Son olarak da, tasarlanan yeni algoritmanın GSP çözümü üzerindeki performansı incelenmiştir.

Bölümün ilerleyen kısımlarında GSP, GA, AS, ACS, MMAS algoritmaları ve melezleştirme yöntemi anlatılmıştır.

3.1.Gezgin Satıcı Problemi(GSP)

Gezgin Satıcı Problemi'nde (GSP) amaç, bir satıcının, bulunduğu şehirden başlayıp, her şehre sadece bir kez uğradıktan sonra başladığı şehre geri dönen en kısa turu bulmaktır. Herhangi iki şehir arasında bir yol olduğunu ve o yolun uzunluğunu bilindiği varsayılır. Görüldüğü gibi, GSP, anlaşılması için matematiksel herhangi bir temel gerektirmeyen bir problemdir. Anlaşılması kolaydır ama çözümü zordur! GSP, grafik kuramı dilinde, şehirlerin noktalarla, şehirlerarası yolların kenarlarla temsil edildiği (yalın) bir grafik üzerinde, en

kısa Hamilton turunun bulunmasıdır. Hamilton turu, bir grafik üzerindeki her noktadan sadece bir kez geçen (dolayısıyla aynı yoldan da sadece bir kez geçen) ve başlladığı noktada biten, 19. yüzyılda yaşamış matematikçi William Hamilton'ın adıyla anılan turdur.

n noktadan oluşan bir grafikte, yani Kn grafiğinde (n−1)!/2 Hamilton turu içerir. Bu çözüm yöntemiyle, 10 şehir içeren bir GSP için bulunması gereken tur sayısı 9!/2 = 181.440'tır. Şehir sayısı 20'ye çıktığında ise bulunması gereken tur sayısı 19!/2 ≈ 6,08 × 1016'yı bulur (Haldun Sural,2003) ve (Hingrajiya, K. H., Gupta, R. K., ve Chandel, G. S., 2012).

GSP için aşağıdaki üç adımlık bir çözüm yolu geliştirilebilir:

1. Grafiğin tüm Hamilton turlarını bul
2. Her turun uzunluğunu hesapla.
3. Turlar arasından en kısasını seç.

Literatürde klasik GSP'nin amaç fonksiyonunun değiştirilmesi ve/veya kısıtlarının farklılaştırılması ile Simetrik GSP, Asimetrik GSP, Kârlı GSP, Zaman Pencereli GSP (ZPGSP), Belirsiz GSP, İki Depolu Heterojen GSP (2-HTSP), Çoklu GSP'de (ÇGSP) ve Açık Döngülü GSP isimlerinde farklı GSP çeşitleri bulunmaktadır (Sultan Kuzu, Onur Önay, Uğur Şen, Mustafa Tunçer, Bahadır Fatih Yıldırım, Timur Keskintürk, 2014).

3.2. Genetik Algoritma (GA)

GA doğal genetik ve seçilim için Darwin'in teorisine (Hedge, C. 1874) dayandırılarak geliştirilmiştir. GA temel olarak evrim için gerekli olan doğal sistemler içerisindeki süreçler için simülasyon uygulayabilmek için tasarlanmıştır. GA'nın ele alınan problem için çözüm elde etmek için belirli arama uzayında rastgele araştırma konuları için iyi bir uygulama olduğu düşünülmektedir (Moscato, P., ve Cotta, C., 2003).

Bireyler olarak GA araştırma uzayında popülasyon korunmaktadır, belirlenen problem için bunlardan birinin çözüm olduğu düşünülmektedir. Bireyler daha sonra sınırlı uzunluk öğe vektörü formunda tamsayı n (genler ve şehir sayıları) temsili kullanılarak kodlanırlar. Bu bireyler genetik analojinin devamı için kromozomlar ile bağlanırlar. Bu yüzden sonuç olarak, bir çok gen (değişkenler) kromozom (çözüm) içerisine dahil edilirler (Zukhri, Z., ve Paputungan, I. V., 2013). Sonuç olarak, kromozom aşağıdaki Şekil 3.1'de gösterildiği gibi verilir.

Tam sayı n = {2,5,3,9…….. n};

2	**5**	**3**	**9**	**……..**	**n**

Şekil 3.1. Tamsayı gösteriminde kromozom örneği

Bu dizi birçok tam sayı geni (şehir sayısını) içerir, her biri daha sonra çözüm karakteristiğinin bir parçasını oluşturur.

Evrimsel algoritma konusunun detaylı analizinden önce temel bazı kavramların ve terminolojinin açıklanması gerekmektedir. Aşağıda bazı terimler verilmiştir (Joshi, G., 2014):

1. Popülasyon: Popülasyon bireyler topluluğudur bu nedenle uzayın bireysel elemanlarıdır. Başlangıçta popülasyon rastgele oluşturulur.

2. Jenerasyon: Jenerasyon mevcut popülasyondur. Her seçilim döngüsünden sonra, çaprazlama ve mutasyon yeni birey jenerasyonu oluşturur.

3. Genotip: Bir bireyin genetik bilgi dizesi.

4. Fenotip: Bir bireyin gözlemlenen özellikler kümesidir.

5. Operatörler: Seçim, Çaprazlama, Mutasyon, vb. bu operatörler bireyleri değiştirmek ve sonrasında yeni özellikler edinen bir popülasyon oluşturmak için kullanılır.

6. Uygunluk: Amaç fonksiyonu (uygunluk fonksiyonu) bireyin kalitesini belirlemek için kullanılır. Bu çalışmada Gezgin Satıcı Problemi için belirlenen

amaç fonksiyon, şehirleri temsil eden noktalar arasındaki uzaklıkların hesaplanmasının ardından her bir alternatif turun toplam yol uzunluğunun hesabıdır. i şehrinin koordinatı $i = (x_1,\ y_1)$ ise iki şehir arasındaki mesafe;

$d_{1,2} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ olarak hesaplanır. Problemimizdeki maliyeti tanımlayan turun toplam uzunluğu ise $Maliyet = d_1 + d_2 + \cdots + d_n$ biçiminde hesaplanır. Amac fonksiyonu ifade eden bu toplam uzunluk;
$Maliyet(i) = \sum_{n=1}^{i} d_n$ olarak da ifade edilebilir.

Genetik Algoritma (GA) Evrimsel Algoritmalar (EA) sınıfına ait olan doğal seleksiyon süreçlerinden ilham alan sezgi üstü bir algoritmadır.Evrimsel algoritmalar genellikle seçilim, çaprazlama ve mutasyon işlemlerinden oluşan üç adım ile tanımlanır (Joshi, G., 2014), (Tang K.S, Man K.F, Kwong S, He Q, 1996), (Dwivedi, V., Chauhan, T., Saxena, S., ve Agrawal, P., 2012).

1. Seçim: Evrimsel strateji ile (optimizasyon tekniği) gelecek nesilleri üretmek için kullanılan genomların seçimidir. Amaç, daha iyi bireylerin tercih edilmesi ve böylece gelecek nesillere kendi genlerini aktarmalarıdır. Her bir bireyin iyiliği bireyi uygunluğuna dayanır. Uygunluk problemin çözümünü sağlayacak bir amaç fonksiyon veye özel bir değer ile tanımlanır.

2. Çaprazlama: Mevcut popülasyondan yeni generasyon oluşturmanın yöntemidir. Popülasyonu oluşturan herhangi iki bireyi genotip dizesinin rastgele belirleme herhangi bir noktasıdan karşılıklı değiştirilmesi işlemidir.

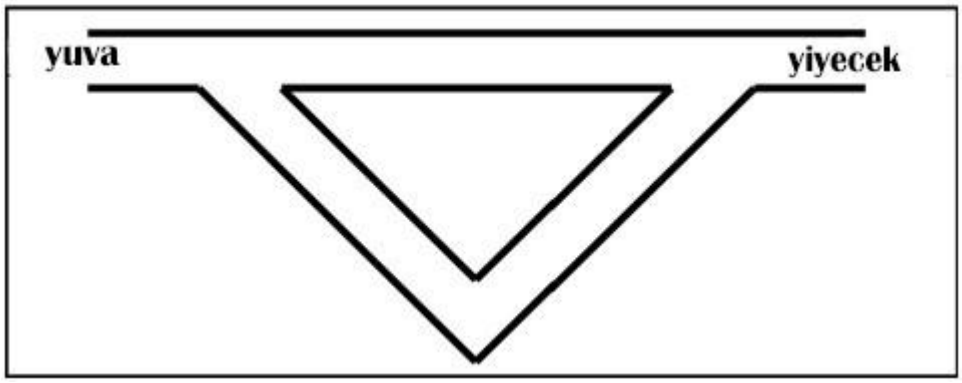
3. Mutasyon: Mutasyon genetik algoritmalar için sadece bazı bireylerin rasgele bir yeni üzerinde uygulanır. Evrimsel algoritmalar bunun aksine daima kullanılmaktadır. Gerçek mutasyon rastgele bir numaranın eklenmesi ile uygulanır. Rastgele numaranın uygun standart sapması genellikle sıfıra yakın bir değer olacak şekilde küçüktür. Bu sayede bir birey üzerinde sadece küçük sapmalar elde edilir. Böylelikle, genetik algoritmalar gibi büyük değişiklikler meydana gelmez (Streichert, F.2002).

3.3. Karınca Kolonisi Optimizasyonu Kavramı

Karınca Kolonisi Optimizasyonu GSP'de dahil olmak üzere çeşitli uygulamalar için optimizasyon problemlerini çözmede kullanılan en yaygın sezgisel algoritmalardan bir tanesidir. Karınca Kolonisi Optimizasyonu için ilham verici olan kaynak gerçek karınca kolonileridir. Özellikle, karıncanın yiyecek arama davranışı bu algoritma için ilham verici olmuştur. Karıncalar arasındaki kimyasal feromon izleri sayesinde dolaylı iletişim bu davranışın temelini oluşturmaktadır. Bu sayede karıncalar yiyecek kaynağına en kestirmeden giden yolu belirlemekte ve keşfetmektedir.

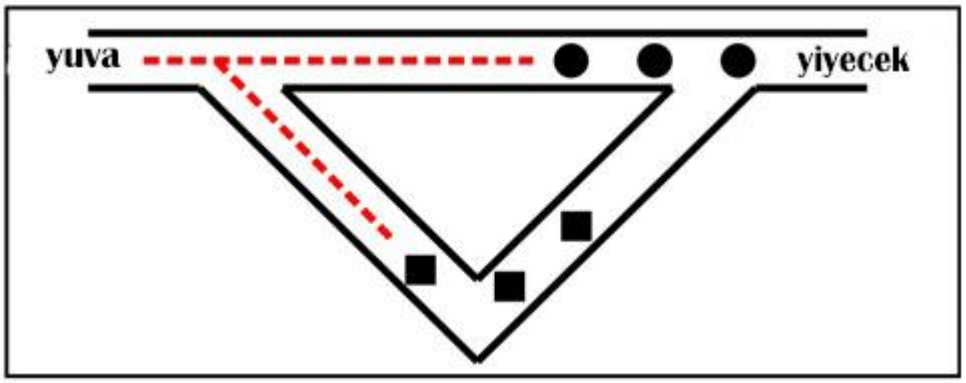
Optimizasyon problemleri karınca kolonisi özelliklerine dayanarak geliştirilen algoritmalar ile kullanılarak çözülebilmektedir. Karıncalar koloniler halinde yaşayan sosyal böcekler olarak sınıflandırılmaktadır. Karınca davranışı bireylerin yaşamlarına odaklanmaktansa doğrudan koloninin yaşamına dayanarak sürdürülür ve yönlendirilir. Karınca yiyecek bulmaya teşebbüs ettiğinde, etrafı çevreleyen alan karıncalar tarafından rastgele bir şekilde araştırılmaktadır. Karınca hareketleri sırasında yerde kimyasal feromon izleri bırakılır. Bu feromon karıncalar tarafından koklanabilir. En yoğun feromon kokusunun alındığı yolun karıncalar tarafından tercih edilmesi daha olasıdır.

Yiyecek kaynağı karınca tarafından bulunduğunda, bulunan yiyeceğin miktarı ve kalitesi için tahminde bulunulur. Bu yiyeceğin bir parçası karıncalar tarafından yuvaya taşınır. Karıncalar daha önce gittikleri yoldan geri dönerlerken bıraktıkları feromon miktarı ve bunun sonucu ortaya çıkan feromon yoğunluğu artmaktadır. Yiyecek kaynağı ve karıncaların yuvası arasındaki en kestirme yol bırakılan feromon izleri sayesinde oluşturulan dolaylı iletişim üzerinden bulunmaktadır. Yiyecek bulmadaki karınca davranış adımları Şekil 3.2, 3.3, 3.4, ve 3.5'te gösterilmektedir (Blum, C., 2005).



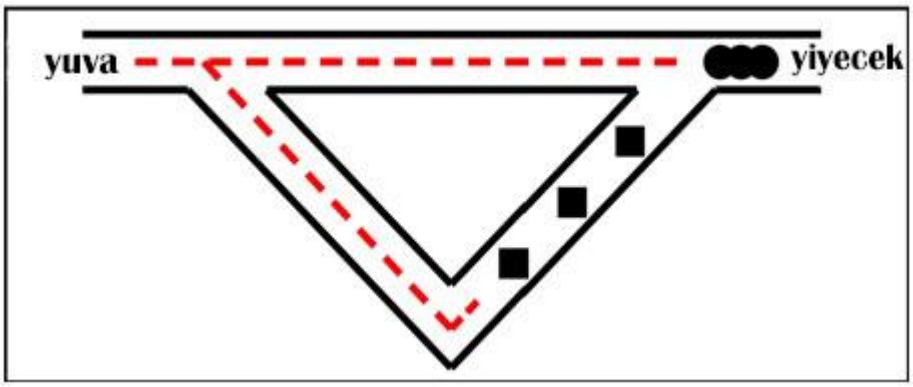
Şekil 3.2. Karınca kolonileri için en kestirme yolu bulma, Adım1

Şekil 3.2'de gösterildiği gibi, bütün karıncalar yuvanın içindedir ve yiyecek kaynağına doğru iki muhtemel yol için feromon miktarı bulunmamaktadır.



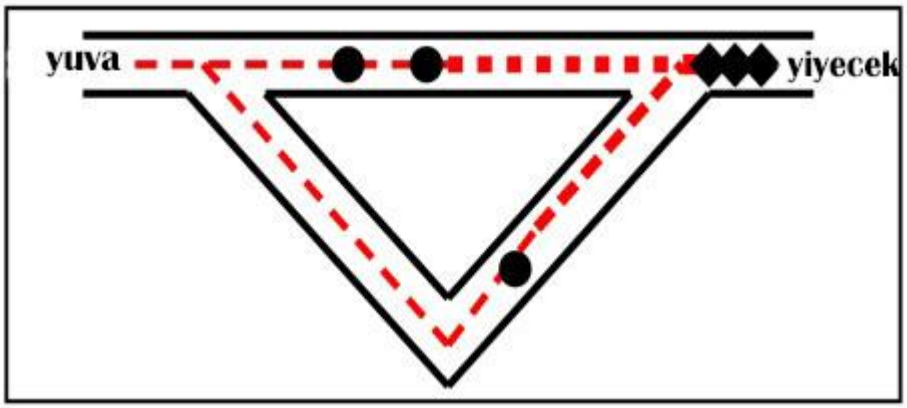
Şekil 3.3. Karınca kolonileri için en kestirme yolu bulma, Adım 2

Yiyecek kaynağına doğru giden en kestirme yolu keşfetme sürecinde ikinci adım yiyecek arama sürecini başlatmaktır. Şekil 3.3'te gösterildiği gibi, karınca tarafından seçilmesi muhtemel iki yol söz konusudur. Başlangıçta yol üzerinde hiçbir feromon miktarının bulunmaması nedeniyle, her iki yolunda karıncalar tarafından tercih edilme olasılığı 0.5'tir. Burada ilk yola doğru giden karıncalar daire şekli ile, ikinci yola doğru giden karıncalar ise kare şekli ile gösterilmiştir. Kırmızı renk ise karıncaların bu yolları kullanırken bırakmış oldukları feromonu göstermektedir.



Şekil 3.4. Karınca kolonileri İçin en kestirme yolu bulma, Adım 3

Şekil 3.4 'te, en kestirme yolu takip eden karıncalar tarafından daha az maliyete ihtiyaç duyulmuştur. Bir önceki şekilde olduğu gibi bu karıncalar burada da daire şekli ile gösterilmiştir. Sonuç olarak, bu yolun yiyeceklerini yuvaya geri taşıyan karıncalar tarafından tercih edilmesi daha olasıdır. Bu da yol üzerinde bırakılan feromon miktarını arttıracaktır.



Şekil 3.5. Karınca kolonileri için en kestirme yolu bulma, Adım 4.

Yüksek miktarda feromon yüzünden en kestirme yolun seçilmesi olasılığı bakımından güçlendirme yapılmış olur. Sonuç olarak, bu yolun karıncalar tarafından seçilmesi olasılığı artmış olacaktır. Son olarak, uzun yoldaki feromon buharlaşacağı için bütün koloni örnek Şekil 3.5'te gösterildiği gibi en kısa yol olan ve en yoğun feromon konsantresini taşıyan yolu kullanacaklardır (Dorigo, M., Di Caro, G., ve Gambardella, L. M., 1999) ve (Blum, C., 2005).

3.3.1. GSP için Karınca Kolonisi optimizasyonu matematiksel modeli

İlk olarak aşağıdaki parametreler tanımlanır;

$V = \{a, b, c, \ldots \ldots \ldots, z\}$ şehir dizisini gösterir.

$\boldsymbol{A} = \{(\boldsymbol{i}, \boldsymbol{j}): \boldsymbol{i}, \boldsymbol{j} \in \boldsymbol{V}\}$ kenar dizisini gösterir

$\delta(i, j)$ =kenar ile ilgili maliyet için ölçüm parametresidir $(i, j) \in A$.

GSP muhtemel en az maliyet ile bir kapalı turu bulmaktır. Bu yolculuk boyunca şehirler yalnızca bir kez ziyaret edilir. Öklid GSP şehirlerin konumları bakımından tanınması ile elde edilir ve $\delta(i, j)$ i ve j arasındaki ''Öklid Uzaklığını'' tanımlar. Asimetrik GSP (AGSP) maliyet parametresinin özel (i,j) için $\delta(i, j) \neq \delta(j, i)$olduğu durumlarda elde edilirken, simetrik GSP $\delta(i, j) = \delta(i, j)$'dir (Mavrovouniotis, M., Müller, F. M., & Yang, S. 2016) (M. Dorigo and L.M. Gambardella,1997; 1996). Kapsamlı ve toplam tur her bir karıncanın ''Olasılıksal Durum Geçiş Kuralına'' dayanarak şehirleri seçmesi vasıtasıyla oluşturulur. Tipik olarak kısa kenarlar üzerinden bağlantılı olan şehirler yüksek feromon miktarı ile karakterize edilir. Kısa yol için feromon küresel güncelleme kuralı bütün koloninin turlarını bitirmeleri ile uygulanır. Bu yüzden, ölçüm parametreleri de çalışma boyunca kullanılacaktır. Bu parametreler feromondur ve $\tau(i, j)$ile gösterilir. Bu parametre yapay karınca vasıtası ile güncellenir.

Çözüm oluşturmada karıncalar stokastik (rastgele) mekanizma vasıtası ile ziyaret edilecek şehir olarak sıradaki şehri seçmektedir. k karıncası i şehrindeyken s^p kısmi çözümünü oluşturmuştur, j şehrine gitme olasılığı Denklem.3.1 ((Mavrovouniotis, M., Müller, F. M., & Yang, S., 2016) ve (Bajaj, R., ve Malik, V., 2016)).

Aşağdaki bilgiler GSP parametrelerinin seçilen değerlerini göstermektedir. Her iki deney içinde çalışma boyunca aynı paramereler kullanılmıştır. şehir sayısı: Harita yapısına dayanmaktadır.

$$p^k ij = \begin{Bmatrix} \frac{\tau ij^{\alpha} . \eta ij^{\beta}}{\sum_{cij \in N(s^p)} \tau ij^{\alpha} . \eta ij^{\beta}}, & cij \in N(s^p) \\ 0 , & aski\ durumda \end{Bmatrix} \quad (3.1)$$

Denklem (3.1) i şehrinde bulunan k karıncasının j şehrine seyahat etme olasılığını göstermektedir. Burada;

τ: feromonu göster mektedir.

$\eta = \frac{1}{\delta} . \delta(i, j)$ uzaklığı için ters olarak tanımlanır.

$N(s^p)$: makul öğelerin dizisidir; yani kenarlar (i, l) burada l karıncası tarafından henüz ziyaret edilmemiş olan şehirdir.

β,α>0:(β=3,α=1)(Botee, H. M., ve Bonabeau, E., 1998). Bu parametre uzaklık fonksiyonu olarak feromon göreli önemliliğini belirlemek ile ilgilidir.

(i,j) kenarındaki feromon eş değeri η (i,j) sezgisel bilgi ile çarpılmıştır. M sonucu olarak, en iyi yol en kestirme yolu olmasının yanında, en yüksek feromon miktarına bağlı olarak seçilecektir. Bütün koloninin turlarının oluşturulması durumunda, Denklem 3.2. bütün kenarlar için feromon değerini güncellemek için uygulanır;

$$\tau(i, j) \leftarrow (1 - p) . \tau ij + \sum_{k=1}^{m} \Delta\tau^k ij \quad (3.2)$$

burada;

$$\Delta\tau^k i = \begin{Bmatrix} \frac{1}{L_k} , & \text{Eğer tur yaparken k karıncası(i, j)kenarını kullandı ise} \\ 0, & aksi\ durum\ da \end{Bmatrix} \quad (3.3)$$

p: feromonun buharlaşma oran parametresidir, (0,1)(ρ=0.1) aralığında değer alır (Botee,H. M., & Bonabeau, E. 1998).

m: koloni içerisindeki toplam karınca sayısını göstermektedir.

L_k : k karıncası üzerinden yapılan tur uzunluğunu göstermektedir.

Bu matematiksel modelde verilen Denklem (3.1) ve varsayılan parametreler Karınca Kolonileri Optimizasyonu algoritmalarının tamamında kullanılmaktadır (Onwubolu, G. C., ve Babu, B. V., 2013).

3.3.2. Karınca Sistemi (AS) matematiksel modeli

Karınca sistemi (Dorigo, M., Maniezzo, V., ve Colorni, A., 1991; 1996) ve (Cordon, O., Herrera, F., ve Stützle, T., 2002) de öne sürülen ilk Karınca Kolonileri Optimizasyonu algoritmasıdır. Temel özelliği şudur: her iterasyonda, feromon değerleri tekrar içerisinde çözüm oluşturan m karıncaları tarafından güncellenir. Feromon $\tau_{i\,j}$ kenar bağlantı şehirleri i ve j ile ilişkilendirilip Denklem 3.2 ve Denklem 3.3'te güncellenir.

3.3.3. Karınca Kolonisi Sistemi (ACS) matematiksel modeli

ACS oluşturma sürecinin sonunda uygulanan (offline feromon güncelleme olarak adlandırılan) feromon güncellemesine ek olarak lokal feromon güncellemesinin sunulmasıdır. Yerel feromon güncelleme bütün karıncalar tarafından her bir oluşturma adımından sonra uygulanır. Her karınca bunu Denklem 3.4'deki gibi yalnızca geçtiği son kenara uygular (M. Dorigo and L.M. Gambardella,1997; 1996) ve (Cordon, O., Herrera, F., & Stützle, T., 2002):

$$\tau(i,j) \leftarrow (1-\varphi).\tau ij + \varphi.\tau_0 \qquad (3.4)$$

Denklem 3.4'de $\varphi\epsilon$ (0, 1] (φ=0.1)(Botee,H.M.,& Bonabeau,E.1998), feromon yok olma katsayısıdır, τ=0 feromonun başlangıç değeridir. Lokal güncellemenin temel amacı tekrar sırasında sonradan gelen karıncalar tarafından yapılan araştırmayı çeşitlendirmektir.

Buna karşın offline feromon güncellemesi için güncelleme formülü Denklem 3.5'deki gibi nispeten farklıdır.

$$\tau i,j \leftarrow \begin{cases} (1-p).\tau ij + p.\Delta\tau\, ij \\ \tau\, i,j \qquad\qquad \text{aksi durum da} \end{cases} \tag{3.5}$$

3.3.4. MIN-MAX Karınca Kolonisi (MMAS) matematiksel modeli

Bu algoritma (Stützle, T., and Hoos, H. H) ve (Cordon, O., Herrera, F., & Stützle, T.2002) orijinal Karınca Sistemi üzerinde bir geliştirilmiştir. Belirleyici özelliği sadece en iyi karıncanın feromon izlerini güncellemesidir ve feromon değerine bağlıdır. Feromon güncellemesi Denklem 3.6'de gösterildiği şekilde uygulanır:

$$\tau(i,j) \leftarrow \left[(1-p).\tau ij + \Delta\tau^{best}\, ij\right]_{\tau_{min}}^{\tau_{max}} \tag{3.6}$$

burada $\boldsymbol{\tau_{max}}$ ve $\boldsymbol{\tau_{min}}$ feromon üzerinde uygulanan göreli en yüksek ve en düşük sınırlardır.

$$\Delta\tau^{best}\, ij = \begin{cases} \dfrac{1}{L_{best}}\text{, Eğer } (i,j) \text{en iyi tura aitise} \\ 0, \qquad\qquad \text{aksi durum da} \end{cases} \tag{3.7}$$

Burada L_{best} en iyi karıncanın tur uzunluğudur. Bu tekrariçerisinde bulunan mevcut en iyi tur, L_{ib} veya en iyi karınca için güncel feromona dayanan en iyi çözüm, L_{bs} veya her ikisininde bir kombinasyonudur. Feromon değerleri üzerindeki en düşük ve en yüksek sınırlar ele alındığında(τ_{min} ve τ_{max}) tipik olarak denegsel biçimde elde edilir ve GSP gibi belirli problem üzerinde uygulanır.

Aslında feromon güncelleme kısa yol üzerindeki feromonu arttırmak için uygulanır. Böylelikle bütün koloni tarafından takip edilir. Karınca Kolonileri Optimizasyonu süreci matematiksel modeli aşağıdaki gibidir;

- Düğüm (örneğin şehirler) sayısının belirlemesi: Bu çalışmada n şehir veya Düğüm ile kullanıcı ara yüzünde rastgele ve TSPLİB tarafından tanımlanan veri setlerindeki düğümler kullanılmıştır.

- Karıncaların sayısının belirlemesi: Bu çalışmada karınca sayısı = 5 olarak gösterilmiştir.

- Koloni içerisindeki her bir karınca tarafından daha sonra durum Geçiş Kuralı uygulanarak bir tur oluşturulması sağlanmıştır. Bu tur sırayla Denklem.3.1'de muhtemel GSP çözüm dikkate alınmıştır.

- Karıncalar tarafından ziyaret edilen kenarlar (i, j) içerisindeki feromon miktarını güncellemek için karıncalar tarafından lokal güncelleme yöntemi uygulanmıştır.

- Karıncanın turunun tamamlanması durumunda, kenarlardaki feromon miktarının global güncelleme yöntemi kullanılarak bir kez daha güncellenmiştir.

- Daha önce de belirtildiği gibi daha yüksek feromon miktarı bulunduran kenarlar için daha fazla tercih edilen turları oluşturmada hem feromon hem de sezgisel bilgi kullanılmıştır.

- Feromon güncelleme kurallarının tasarımında karıncaların ziyaret etmeleri için daha fazla feromonun konulacağı şekilde uygulanır.

MMAS'ının farkı çalışma performansını arttırmak için iki feromon miktarı kullanmasıdır.

3.4. GA ile MMAS'nin Melezleme Algoritması

Bu çalışmanın temel amacı GA ve MMAS'deki belirli adımları birleştirerek bir kavram ortaya koymak ve böylelikle GSP çözümü bulmak için GA ve MMAS'yi melezleştirmek için uygun bir yaklaşım tanımlamaktır. Melezleştirme aynı zamanda hesaplamada birbirine uyumlu özellikleri taşıyan GA veya MMAS'nin bazı parametre ve değişkenlerine de uygulanmıştır

Örneğin GA'daki popülasyon büyüklüğü ile MMAS'deki karınca sayısı, GA'da kromozom ile MMAS'de şehir sayısı ve GA'da yeni jenerasyon oluşturma ile MMAS'de şehir turlamadaki tekrar sayıları parametreleri benzerlik ve uygunluk göstermektedir.

Kromozom (gen grubu), şehir sayısı n, şehir sayısına eşit olan gen sayıları ve her k karıncası tarafından ziyaret edilen şehir (m toplam karınca) kromozom yapısı (Penev, M. K. 2005), (Zukhri, Z., & Paputungan, I. V.2013), (Dalip, N. K. 2011) Şekil 3.6'de gösterilmiştir.

Gen1(i,j)	Gen2(i,j)	……..	Gen n-1
Şehir 1(i,j),k_m (sadece en iyi olan k)	Şehir 2(i,j),k_m (sadece en iyi olan k)	……..	Şehir n-1 (sadece en iyi olan k)

Şekil 3.6. Kromozom temsili (gösterimi)

Örneğin, 5 şehirli GSP rotası (2-1-7-3-6) yol temsili biçiminde olabilir. Bu rota kromozom dizisi olarak Şekil 3.7'daki gibi gösterilir.

2	**1**	**7**	**3**	**6**
[0]	[1]	[2]	[3]	[4]

Şekil 3.7. Kromozomların dizi temsili (gösterimi)

Örnek 2: İki kromozom altı şehir GSP için mutasyon ve çaprazlamayı (tek düğm) temsil etmektedir.Şekil 3.8'da 6 şehirli kromozom yapısı için çaprazlama ve mutasyon örnekleri görülmektedir.

Baba1:	***1***	***3***	***5***	***7***	***9***	***0***
Baba2:	2	4	6	8	10	11

Çaprazlama :

Oğul1	***1***	***3***	***5***	8	10	11
Oğul2	2	4	6	***7***	***9***	***0***

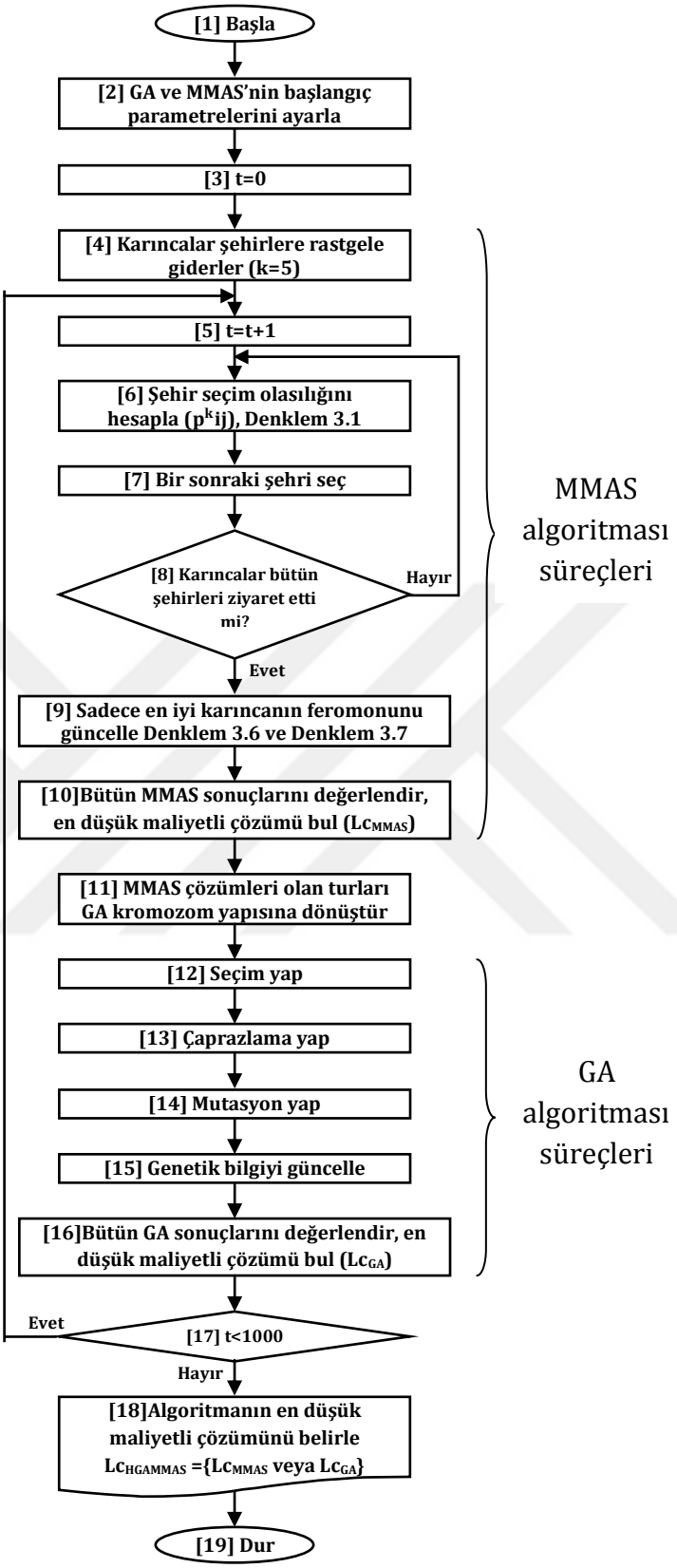
Mutasyon :

Oğul1	***1***	***3***	***5***	8	***9***	11
Oğul2	2	4	6	***7***	10	***0***

Şekil 3.8. Mutasyon ve çaprazlamayı temsil eden kromozomlar

Ara adımları Şekil 3.9'deki akış şemasında verilen HGAMMAS algoritması çözüme başlarken önce MMAS algoritması ile bir çözüm seti oluşturmakta, daha sonra oluşturulan MMAS çözümleri GA süreçlerindeki parametre ve operatörlere uygun biçime dönüştürülerek birde GA çözüm seti oluşturulmaktadır.

MMAS ve GA çözümlerini oluşturan karınca-tur ve gen-kromozom yapıları her tekrarda birbirlerinde kullanılmaya devam edilerek amaç çözüm olan maliyet azaltılmıştır.



Şekil 3.9. HGAMMAS algoritması akış şeması

[1]: HGAMMAS algoritmasını başlat.

[2]: Feromon izlerinin ve bazı parametrelerin başlatılmasından sonra, yolda i ve j şehirleri arası biriken feromon miktarı (i,j) ve r(i,j) başlangıç olarak =0'a ayarlanır.

[3]: Başlangıçtekrar sayısını t = 0 olarak belirle.

[4]: Tekrar sayısını artır t =t+1.

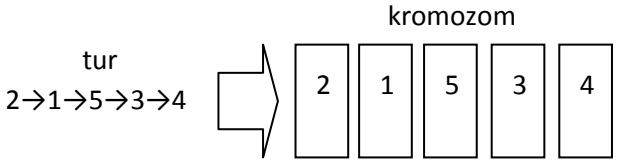
[5]: Karıncalar feromon ismi verilen bazı kimyasalları izledikleri yol boyunca biriktirirler ve böylece diğer karıncaların aynı yolu takip etmelerini sağlamak için diğer karıncaları bu maddelere çekerler. Başlangıçta karıncalar etrafta rastgele yürümektedir, fakat bazı karıncalar kazara belli bir yiyecek kaynağına rastlar ve yuvaya dönerler. Bu çalışmada karınca sayısı k=5 olarak belirlenmiştir.

[6, 7]: Her bir karıncanın seçeceği şehir $p^k ij$ olasılığı ile seçilir. i şehrinden j şehrine hareket eden karıncanın seçim olasılığı Denklem 3.1'de karar verilir.

[8, 9]: Tüm karıncaların bütün şehirleri ziyaret etmesinden ve turlarını tamamlamalarından sonra, feromon izlerini güncellerler. Başlangıçta, bütün izler eşit miktarda feromon içermektedir. MMAS'de her tekrardan sonra, sadece en iyi karınca feromon izlerinigünceller. Feromon güncellemesi Denklem 3.6 ve 3.7'de olduğu gibi yapılmaktadır.

[10]: MMAS için tüm muhtemel çözümler en kısa yol (en düşük maliyet) açısından değerlendirilir. MMAS'ınen iyi çözümü Lc_{MMAS} olarak tespit edilir.

[11]: MMAS sonuçlarını GA sürecine sokmak için Şekil 3.10'de görüldüğü gibi elde edilen karınca-tur çözümleri gen-kromozom biçimine dönüştürülerek GA için popülasyon oluşturulur



Şekil 3.10. Her bir karıncanın elde ettiği turun kromozom olarak gösterilmesi

[12]: Seçim havuzundaki tüm kromozomlar üzerinde seçim işlemini uygula.

[13, 14]: Kromozomların rastgele belirlenen yarısını çaprazlama ve mutasyon GA süreçlerini uygulayarak, kalan yarısını da GA süreçleri uygulanmadan seçim havuzuna gönder.

[15]: Genetik operatörlerden sonra, en iyi geziyi (en düşük maliyeti) bulmak için genetik bilgiyi güncelle. Kazanılan genetik bilgiler, karıncaya en iyi farklı çözümleri bulmak için yardımcı olacaktır. Çözüm geliştirmek için bütün en iyi karıncaların genetik bilgisini (şehir sayısı ve şehirler arasındaki uzaklıklar) taşı. Yeni jenerasyonlar için çözüm çeşitliliğini geliştirmek için genetik bilgiyi en iyi çözüme indirge (daha az maliyet).

[16]: GA amaç fonksiyonu kullanlarak tüm muhtemel çözümler içerisinden en kısa yol (en düşük maliyet) açısından değerlendirilir. GA'nınen iyi çözümü Lc_{GA} olarak tespit edilir. GA sonunda elde edilen en iyi çözümler yeni tekrarda karıncalara başlangıç değeri olur.

[17]: Tanımlanan en yüksek tekrar sayısına ulaşılıncaya kadar GA sonunda elde edilen en iyi gen-kromozom çözümleri MMAS karınca-tur biçimine aktarılarak algoritma çalıştırılmaya devam edilir.

[18]: HGAMMAS algoritmasının en düşük maliyetli çözümü $Lc_{HGAMMAS}$ olarak Lc_{MMAS} veya Lc_{GA}'dan en küçüğü seçilerek belirlenir.

[19]: HGAMMAS algoritmasını bitir.

4. BULGULAR

Bu bölümde, önce Genetik Algoritma (GA), Karınca sistemi Algoritması (Ant System-AS), Karınca Koloni Algoritması (Ant Colony System-ACS), Min-Max Karınca Sistemi Algoritması (Min-Max Ant System -MMAS) ve bu araştırmanın temelini oluşturan, tasarlanan Melezleştirilmiş Genetik Algoritma Min-Max Karınca Sistemi Algoritması (Genetic Algorithm ve Min-Max Ant System Agorithm-HGAMMAS) algoritmasının performansını görmek için hazırlanan arayüz tanıtılmıştır

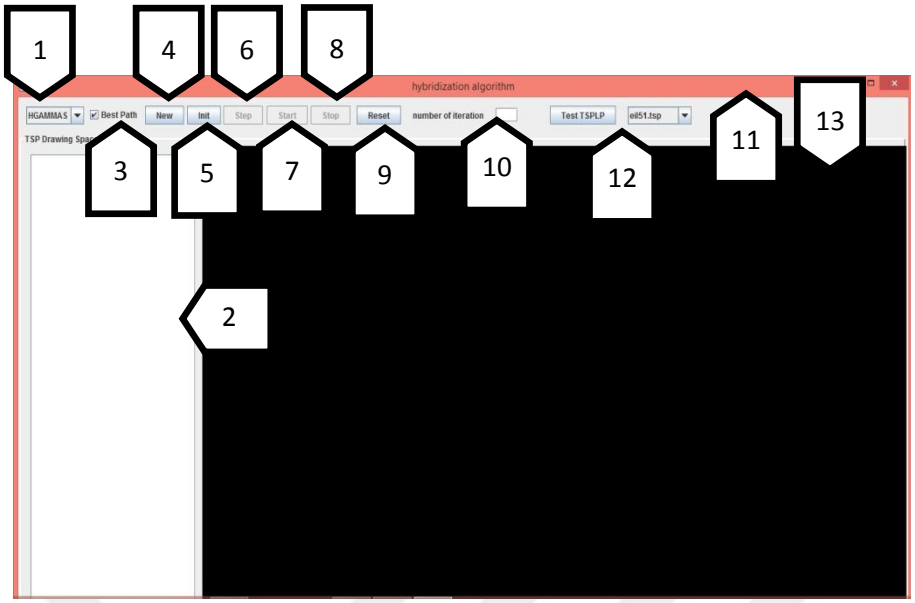
Daha sonra da algoritmalara ait kodların geçerliğini görmek için karmaşık sayılmayacak, az sayıda düğümden oluşan bir harita üzerinde algoritmalar çalıştırılmış ve tamamının aynı maliyette çözümler ürettiği görülmeye çalışılmıştır

Algoritma performansları önce rastgele oluşturulan 36, 56, 76, 101 ve 150 düğüm (şehir)'den oluşan farklı 5 harita üzerinde denenerek analiz edilmiştir. Sonra, literatürde kullanımına sıklıkla rasatlanan GSP için hazırlanmış TSPLIB kütüphanesindeki eli51, berlin52, eli76, ra100, kroa200 veri setleri ile performans denenerek literatür ile kıyaslama yapılmasına olanak sağlanmıştır.

Denemeler her bir veri seti için 1000 iterasyonluk 10 çalıştırmadan elde edilen sonuçların ortalamaları alınarak yapılmıştır. Arayüz ve algoritmalar Java platform JDK 1.7, Intel core i5 ve Windows 8'e sahip kişisel bir bilgisayar (3.00 GHz CPU speed and 4GB RAM) kullanılarak yapılmıştır.

4.1. Uygulamanın Kullanıcı Arayüzü

Şekil 4.1'de görülen kullanıcı arayüzü çalışmada kodlanan beş algoritmanın rastgele oluşturulacak ve hazır veri setlerinden elde edilecek şehirler üzerinde GSP çözümüne ait ara değerleri de gösterebilecek biçimde tasarlanmıştır.



Şekil 4.1. Araştırmada tasarlanan kullanıcı grafik arayüzü

Arayüz özellikleri;

1. Denenmek, uygulanmak istenilen GA, AS, ACS, MMAS ve HGAMMAS algoritmaları için seçim listesi.
2. Algoritmaların adım, maliyet ve düğüm sayısı bilgilerinin listelendiği alan.
3. Deneme, çözüm sonunda hesaplanan en kısa yolu gösterilmesini sağlayan seçim.
4. Yeni bir harita oluşturmak için ekranın temizlenmesini sağlayan seçim.
5. Mevcut harita üzerinde yeni bir deneme, çözüm hesaplaması için iterasyonun 0 değerine alınmasını sağlar.
6. Algoritmanın adım adım çalıştırılmasını sağlar.
7. Algortimayı çalıştırır, denemeyi başlatır.
8. Denemeyi durdurur.
9. Mevcut harita üzerinde diğer algoritmalar için deneme yapılmasını sağlar.
10. Tekrar sayısını tanımlamak için kullanılır.

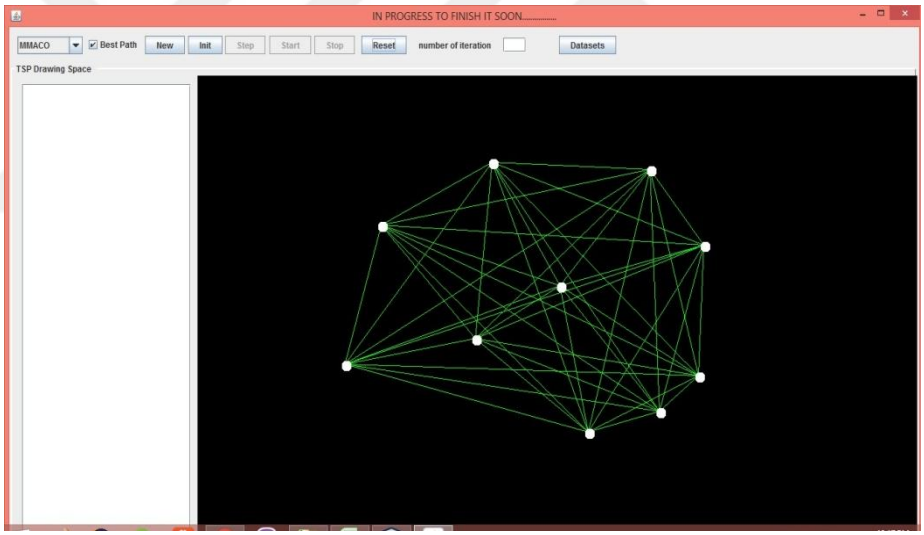
11. Proje başlığı çubuğu.

12. TSPLIB gibi önceden tanımlı kütüphane veri setleri dosyaların seçiminin yapıldığı liste.

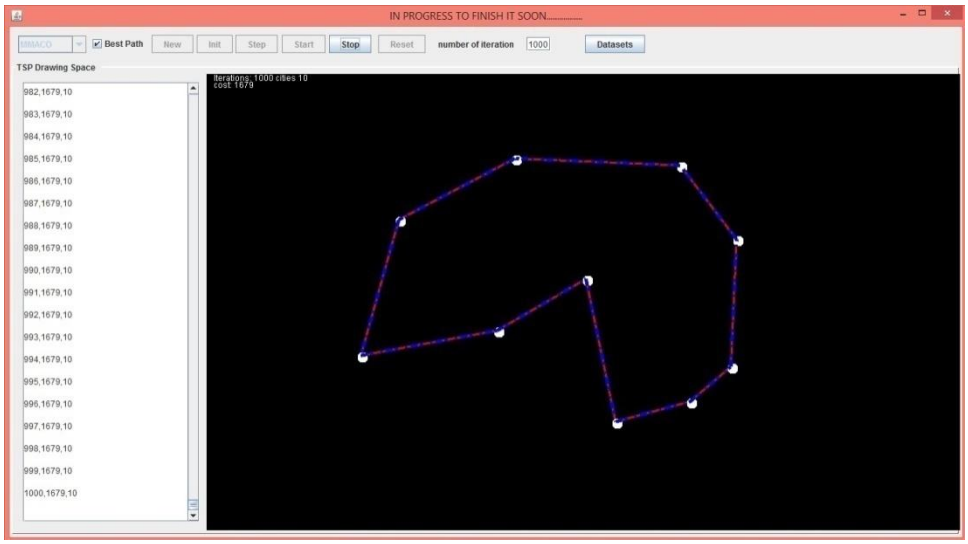
13.Düğümlerin tanımlandığı ve denemenin çözümün izlendiği çalışma alanı.

4.2. Geçerlilik/Doğrulama Denemesi

GA, AS, ACS ve MMAS algoritmaları kodlarının geçerliğini (validation) görebilmek için Şekil 4.2'deki az sayıda düğümden oluşan (10 şehirli) bir haritada algoritmaların kodları çalıştırılmıştır. Algoritmaların tamamı Şekil 4.3'de görülen aynı yollardan oluşan çözümü, 1679 değerindeki aynı maliyetle hesaplayabilmişlerdir.



Şekil 4.2. Geçerlik haritasının deneme öncesi görüntüsü



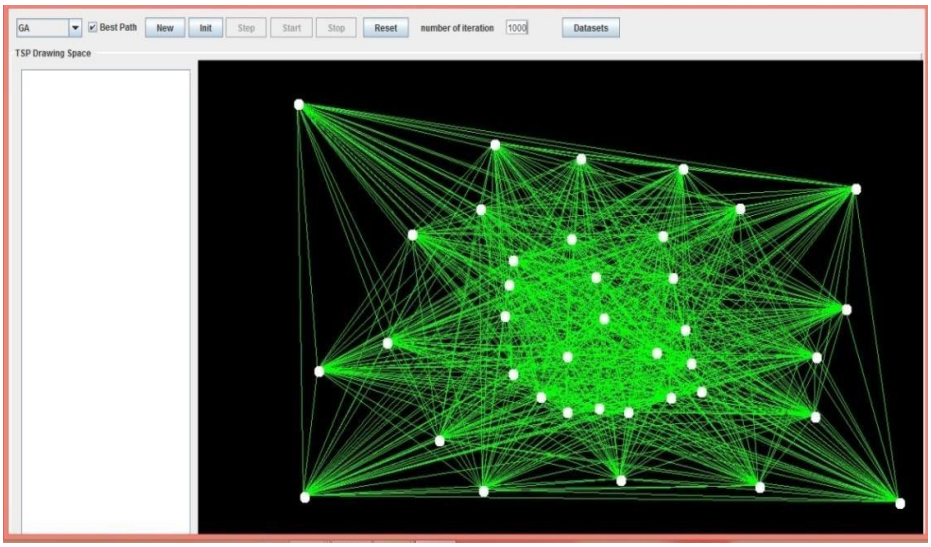
Şekil 4.3. Geçerlik haritasının deneme sonrası görüntüsü

4.3. Rastgele Düğümler ile Performans Denemeleri

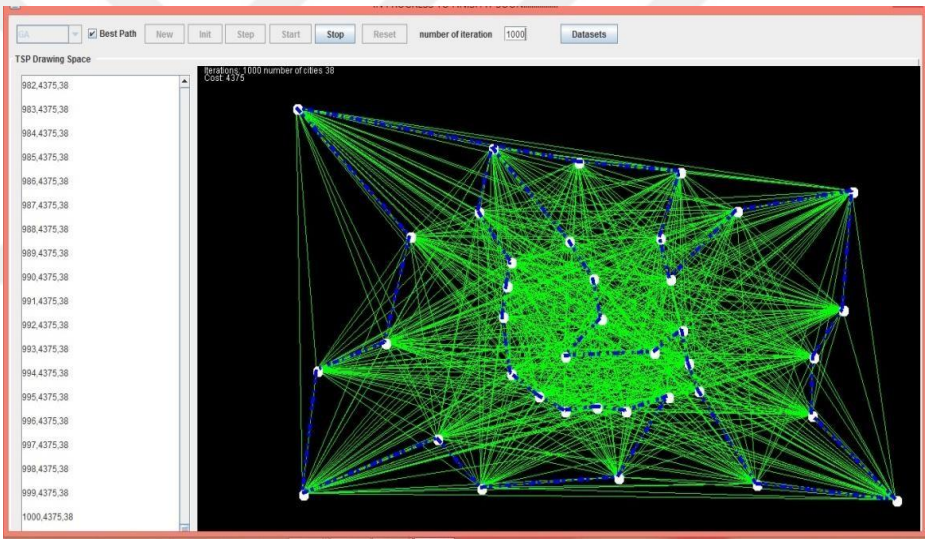
Kodlanan GA, AS, ACS ve MMAS algoritmaları, koordinatları rastgele oluşturulan 36, 56, 76, 101 ve 150 düğüm (şehir)'den oluşan 5 harita üzerinde denenerek performansları incelenmiştir.

4.3.1. 36 Düğüm (Şehir)'li Harita-1 denemesi:

İlk deneme 36 düğümden oluşan Harita-1 ile yapılmıştır. Şekil 4.4'de rastgele 36 şehirin oluşturulduğu ve aralarındaki tüm yol olasılıklarının çizildiği deneme öncesi görüntü görülmektedir.

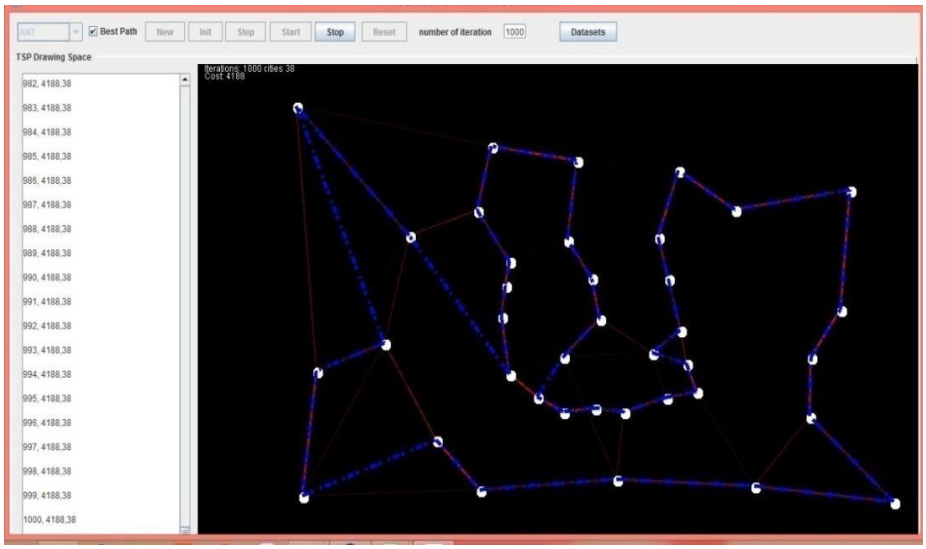


Şekil 4.4. Harita-1'in deneme öncesi görüntüsü



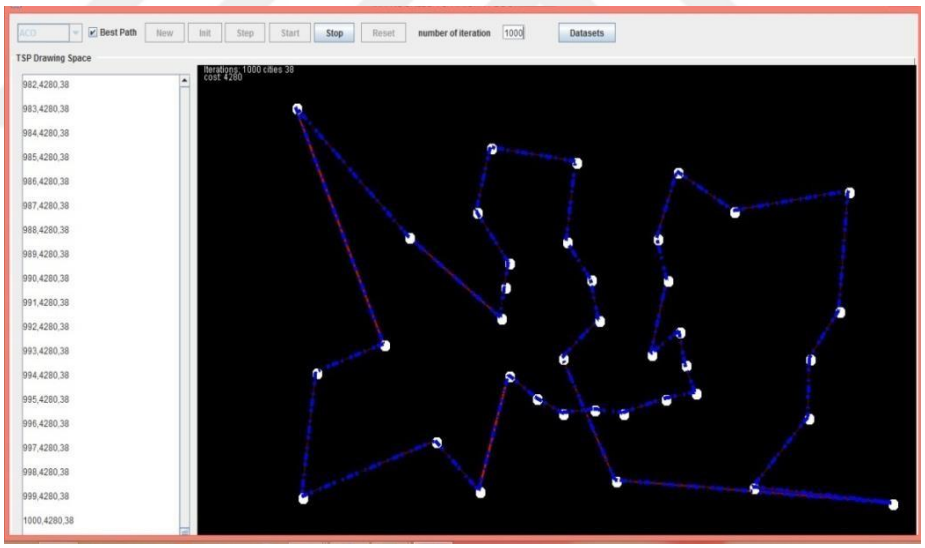
Şekil 4.5. GA Algoritmasının Harita-1 çözümü

Uygulama arayüzünden GA algoritması seçilerek uygulama çalıştırıldıktan sonra Şekil 4.5'deki GA çözümü elde edilmiştir. Mavi çizgiler en kısa yol çözümünü, yeşil çizgiler olası diğer yolları ifade etmektedir.



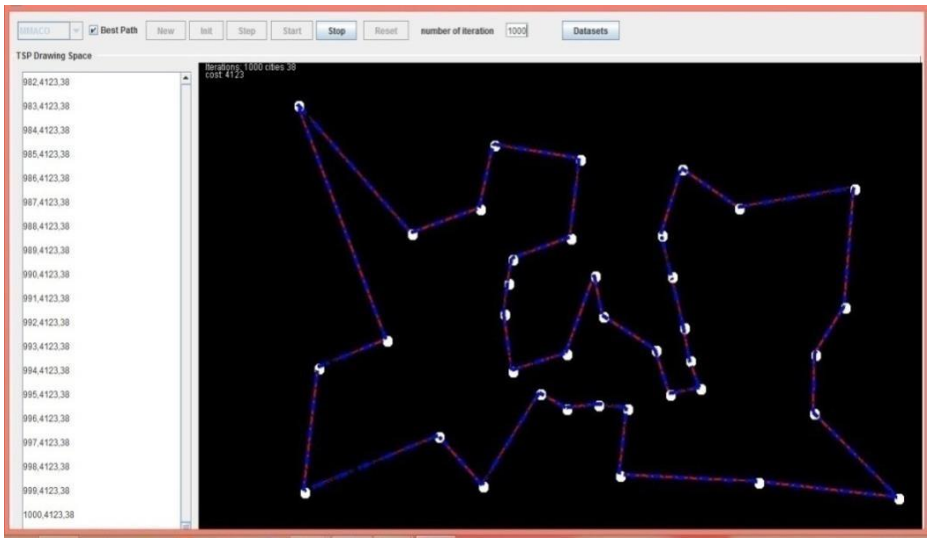
Şekil 4.6. AS Algoritmasının Harita-1 çözümü

Şekil 4.6'da, uygulama arayüzünden AS seçilerek Şekil 4.4'deki harita için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



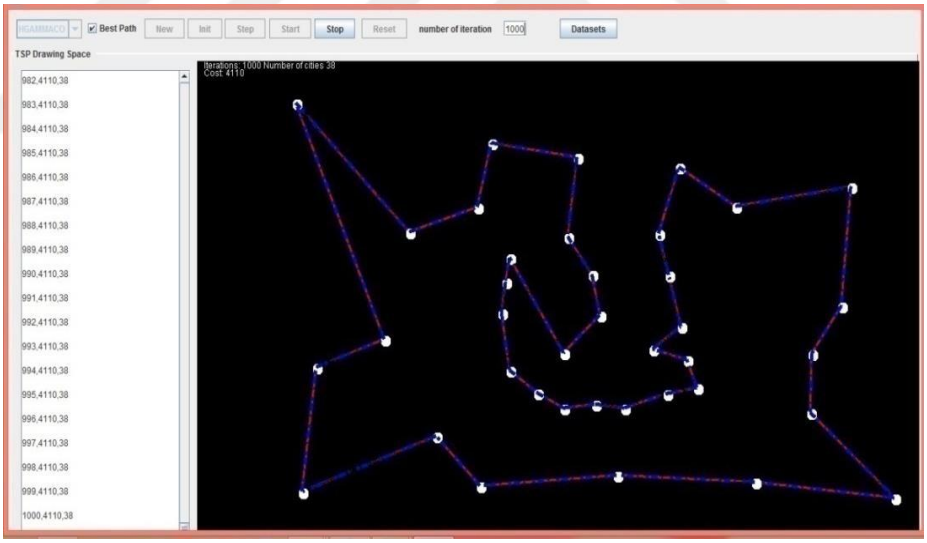
Şekil 4.7. ACS Algoritmasının Harita-1 çözümü

Şekil 4.7'de, uygulama arayüzünden ACS seçilerek Şekil 4.4'deki harita için hesplan en kısa yol çözümü mavi çizgiler ile görülmektedir.



Şekil 4.8 MMAS Algoritmasının Harita-1 çözümü

Şekil 4.8'da, uygulama arayüzünden MMAS seçilerek Şekil 4.4'deki harita için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



Şekil 4.9. HGAMMAS Algoritmasının Harita-1 çözümü

Şekil 4.9'de ise, bu çalışmada tasarlanan HGAMMAS'nın Şekil 4.4'deki harita için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.

Çizelge 4.1'de GA, AS, ACS, MMAS ve HGAMMAS algoritmalarının Harita-1 için iterasyon sayısı ve maliyet açısından performans ayrıntıları görülmektedir.

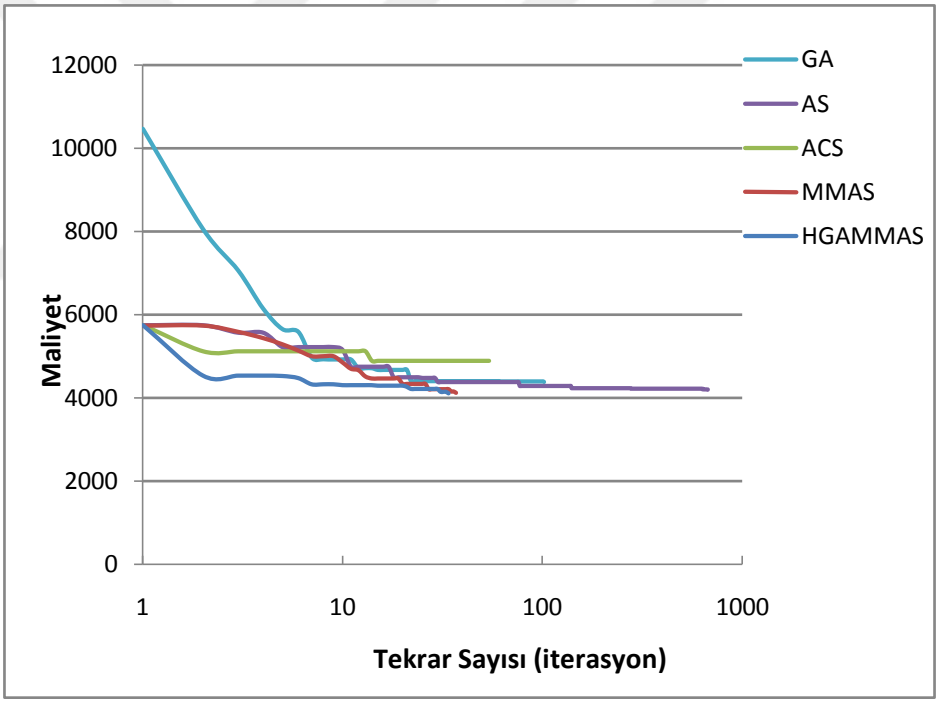
Çizelge 4.1 GA, AS, ACS, MMAS ve HGAMMAS Algoritmalarının Harita-1 için performans ayrıntıları

Harita-1 performansı ayrıntılı verileri					
#iterasyon	**GA maliyet**	**AS maliyet**	**ACS maliyet**	**MMAS maliyet**	**HGAMMAS maliyet**
0	10460	5739	5739	5739	5739
1	8051	5739	5114	5739	4533
-	4395	4470	4886	4338	4216
32	4395	4371	4886	4211	4147
33	4395	4371	4886	4211	4110
-	4395	4371	4886	4156	4110
36	4395	4371	4886	4123	4110
-	4395	4371	4886	4123	4110
53	4395	4371	4886	4123	4110
-	4388	4277	4531	4123	4110
101	4375	4277	4531	4123	4110
-	4375	4223	4531	4123	4110
219	4375	4223	4280	4123	4110
-	4375	4223	4280	4123	4110
678	4375	4188	4280	4123	4110
-	4375	4188	4280	4123	4110
999	4375	4188	4280	4123	4110
1000	4375	4188	4280	4123	4110
ortalama maliyet	**4398.754**	**4234.718**	**4360.243**	**4140.415**	**4117.628372**
en yüksek maliyet	**10460**	**5739**	**5739**	**5739**	**5739**
en az maliyet	**4375**	**4188**	**4280**	**4123**	**4110**

36 düğüm (şehir)'li Harita-1 için GA 10460 maliyet ile çözüme başlarken, AS, ACS ve MMAS algoritmaları 5739 maliyet ile çözüme başlamıştır. GA 101. tekrarda 4375 maliyet ile en iyi çözümünü tamamlarken, AS algoritması 678. tekrarda 4188 maliyet ile, ACS algoritması 219. tekrarda 4280 maliyet ile ve

MMAS algoritması 36. tekrarda 4123 maliyet ile en iyi çözümlerini tamamlamışlardır. Harita-1 için maliyet bazında klasik algoritmalar arasından en iyi çözümü 4123 ile MMAS hesaplamıştır. Ancak, HGAMMAS algoritması 5739 maliyet ile başladığı çözüme 33. tekrarda 4110 maliyet ile tamamlamış böylece daha düşük maliyette ve daha kısa sürede hesaplama yaparak daha yüksek bir performans göstermiştir. Çizelge sonunda da ortalama maliyet, en yüksek maliyet ve en az maliyet bilgileri verilmiştir.

Ortalama maliyet, en yüksek maliyet ve en az maliyet değerleri algoritmaların her birisi aynı harita üzerinde 10'ar defa çalıştırılarak elde edilen değerlerin aritmetik ortalamaları alınarak hesaplanmıştır.

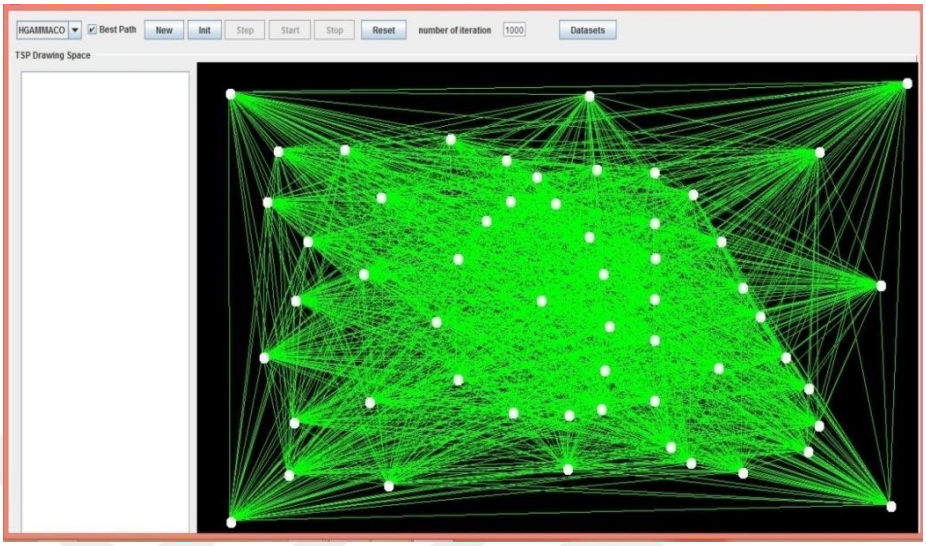


Şekil 4.10. GA, AS, ACS, MMAS ve HGAMMAS Algoritmalarının Harita-1 için iterasyon sayısı ve maliyet performansları

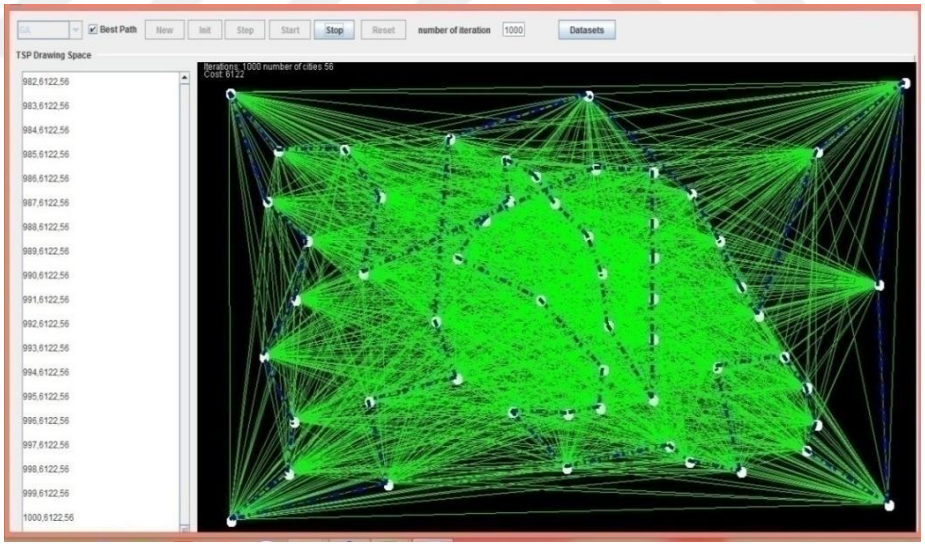
Şekil 4.10'de GA, AS, ACS, MMAS ve HGAMMAS algoritmalarının Çizelge 4.1'de verilen Harita-1 için iterasyon sayısı ve maliyet performansları grafiği görülmektedir.

4.3.2. 56 Düğüm (Şehir)'li Harita-2 denemesi

İkinci deneme de kullanılan Harita-2 rastgele 56 düğüm ile yapılandırılmıştır (Şekil 4.11).

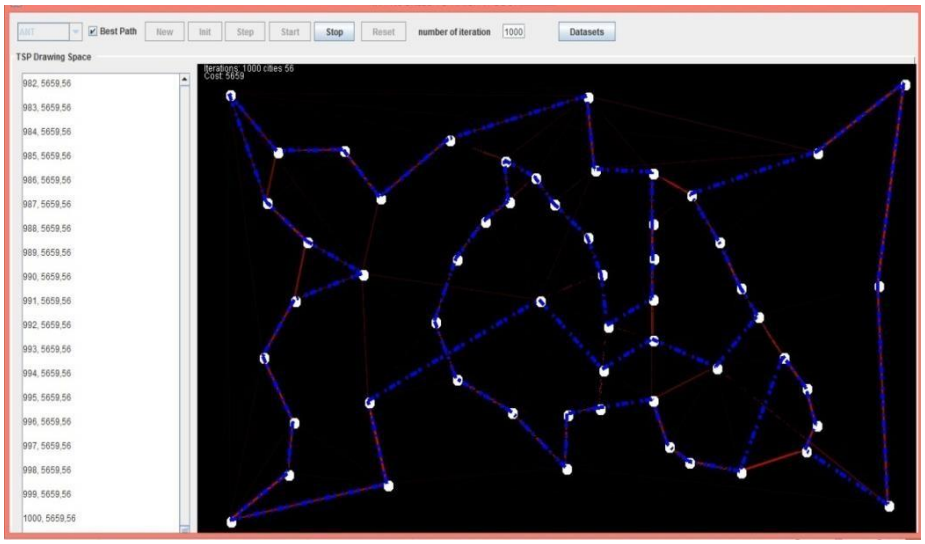


Şekil 4.11. Harita-2'nin deneme öncesi görüntüsü



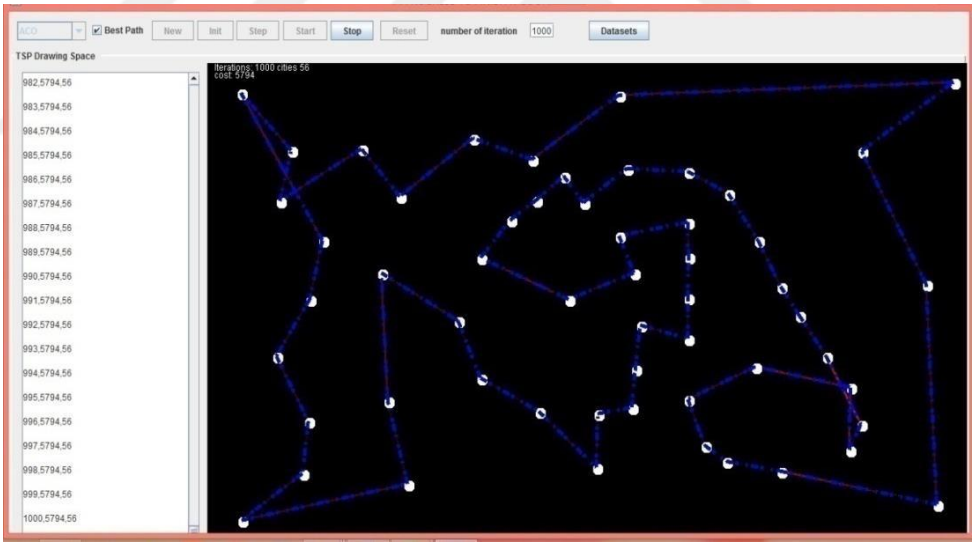
Şekil 4.12. GA Algoritmasının Harita-2 çözümü

Uygulama arayüzünden GA algoritması seçilerek uygulama çalıştırıldıktan sonra Şekil 4.12'daki GA çözümü elde edilmiştir. Mavi çizgiler en kısa yol çözümünü, yeşil çizgiler olası diğer yolları ifade etmektedir.



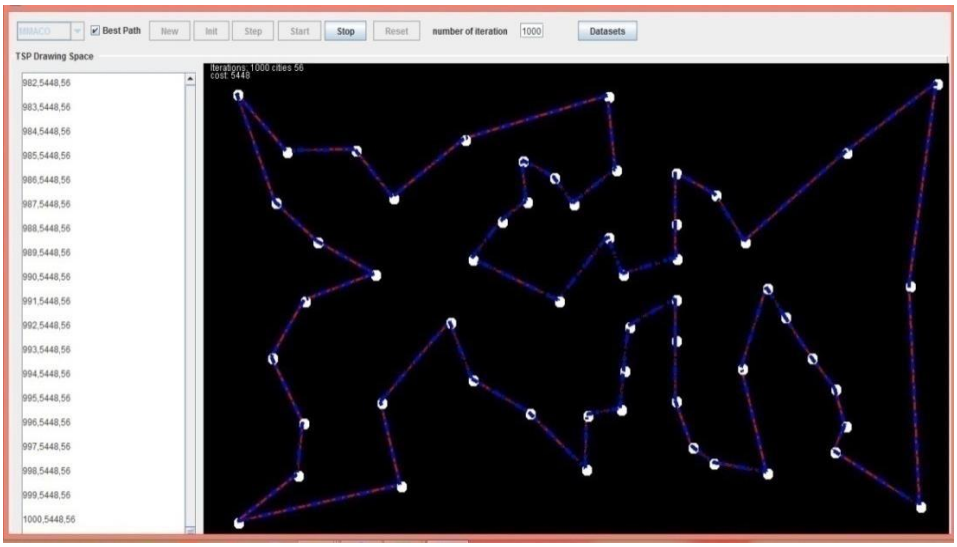
Şekil 4.13. AS Algoritmasının Harita-2 çözümü

Şekil 4.13'de, uygulama arayüzünden AS seçilerek Harita-2 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



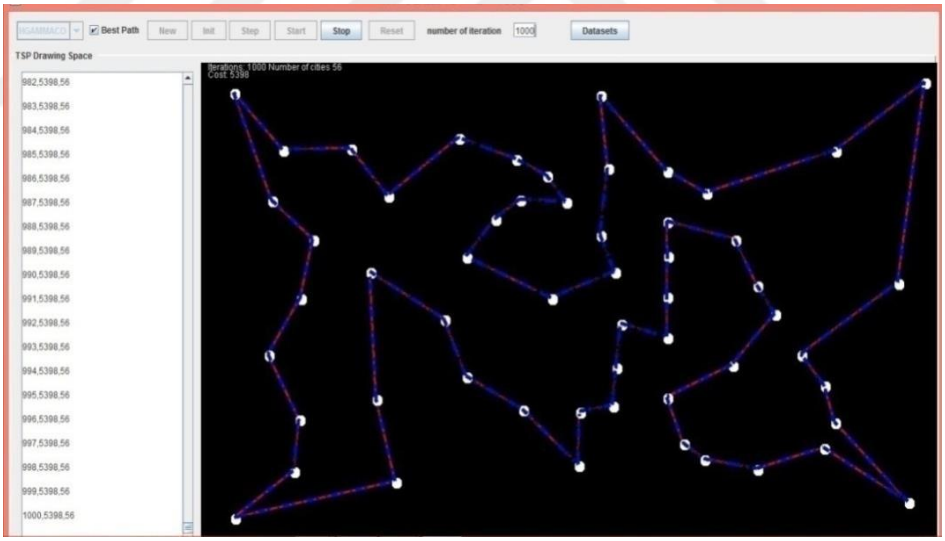
Şekil 4.14 ACS Algoritmasının Harita-2 çözümü

Şekil 4.14'de uygulama arayüzünden ACS seçilerek Harita-2 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



Şekil 4.15. MMAS Algoritmasının Harita-2 çözümü

Şekil 4.15'de, uygulama arayüzünden MMAS seçilerek Harita-2 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



Şekil 4.16. HGAMMAS Algoritmasının Harita-2 çözümü

Şekil 4.16'de ise, bu çalışmada tasarlanan HGAMMAS'nın Harita-2 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir

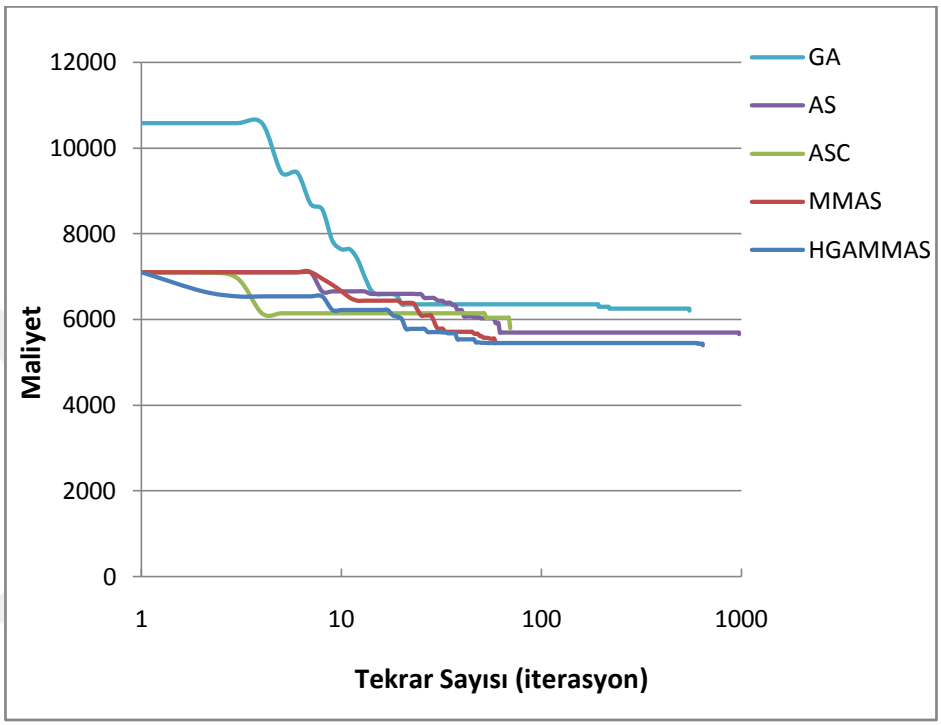
Çizelge 4.2'de GA, AS, ACS, MMAS ve HGAMMAS algoritmalarının Harita-2 için iterasyon sayısı ve maliyet performans ayrıntıları görülmektedir.

Çizelge 4.2. GA, AS, ACS, MMAS ve HGAMMAS Algoritmalarının Harita-2 için ayrıntıları

Harita-2 performansı ayrıntılı verileri					
#iterasyon	**GA maliyet**	**AS maliyet**	**ACS maliyet**	**MMAS maliyet**	**HGAMMAS maliyet**
0	10583	7101	7101	7101	7101
1	10583	7101	7101	7101	6662
-	6358	6028	6041	5555	5452
58	6358	5918	6041	5448	5452
-	6358	5696	6041	5448	5452
69	6358	5696	5794	5448	5452
-	6358	5696	5794	5448	5452
160	6358	5696	5794	5448	5452
-	6182	5696	5794	5448	5437
640	6182	5696	5794	5448	5398
-	6182	5696	5794	5448	5398
900	6122	5696	5794	5448	5398
-	6122	5696	5794	5448	5398
974	6122	5659	5794	5448	5398
-	6122	5659	5794	5448	5398
999	6122	5659	5794	5448	5398
1000	6122	5659	5794	5448	5398
ortalama maliyet	**6270.455**	**5738.708**	**5819.165**	**5486.606**	**5455.582418**
en yüksek maliyet	**10583**	**7101**	**7101**	**7101**	**7101**
en az maliyet	**6122**	**5659**	**5794**	**5448**	**5398**

56 düğüm (şehir)'li Harita-2 için GA 10583 maliyet ile çözüme başlarken, AS, ACS ve MMAS algoritmaları 7101 maliyet ile çözüme başlamıştır. GA 900. tekrarda 6122 maliyet ile en iyi çözümünü tamamlarken, AS algoritması 974. tekrarda 5659 maliyet ile, ACS algoritması 69. tekrarda 5794 maliyet ile ve MMAS algoritması 58. tekrarda 5448 maliyet ile en iyi çözümlerini tamamlamışlardır. Harita-2 için maliyet bazında klasik algoritmalar arasından en iyi çözümü 5448

ile MMAS hesaplamıştır. Ancak, HGAMMAS algoritması 7101 maliyet ile başladığı çözüme 640. tekrarda 5398 maliyet ile tamamlamış, daha düşük maliyette hesaplama yaparak daha yüksek bir performans göstermiştir. Çizelge sonunda da ortalama maliyet, en yüksek maliyet ve en az maliyet bilgileri verilmiştir.

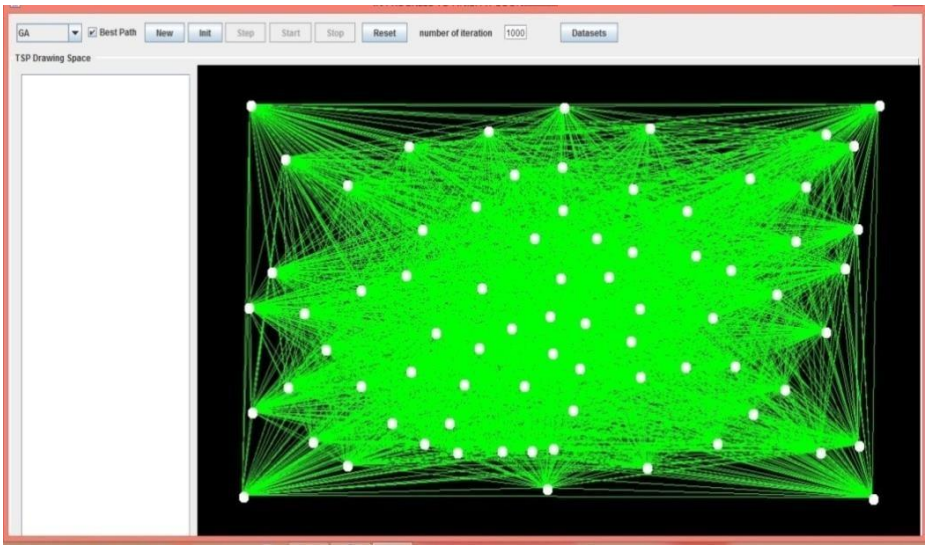


Şekil 4.17. GA, AS, ACS, MMAS ve HGAMMAS Algoritmalarının Harita-2 için iterasyon sayısı ve maliyet performansları

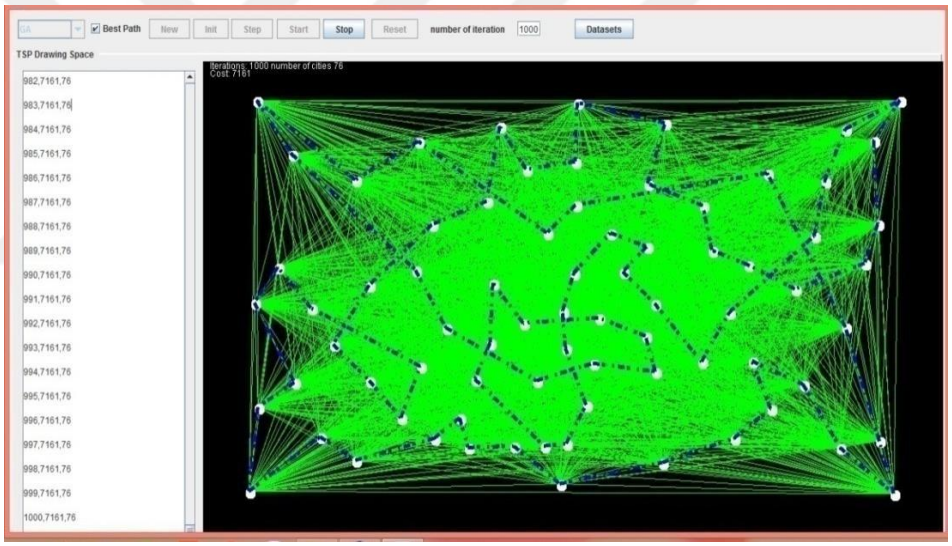
Şekil 4.17'de GA, AS, ACS, MMAS ve HGAMMAS algoritmalarının Çizelge 4.2'de verilen Harita-2 için iterasyon sayısı ve maliyet performansları grafiği görülmektedir.

4.3.3. 76 Düğüm (Şehir)'li Harita-3 denemesi

Üçüncü denemede kullanılan Harita-3 rastgele 76 düğüm ile yapılandırılmıştır (Şekil 4.18).

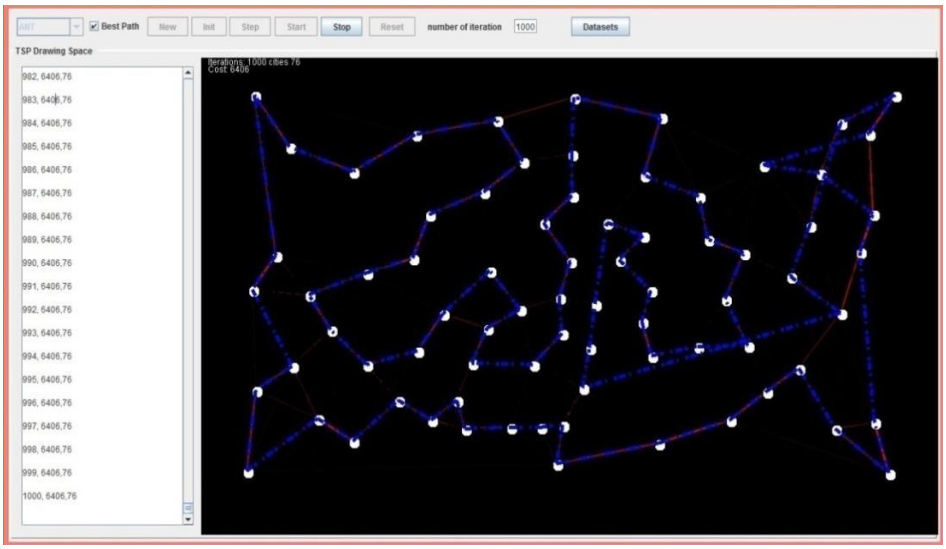


Şekil 4.18. Harita-3'ün deneme öncesi görüntüsü



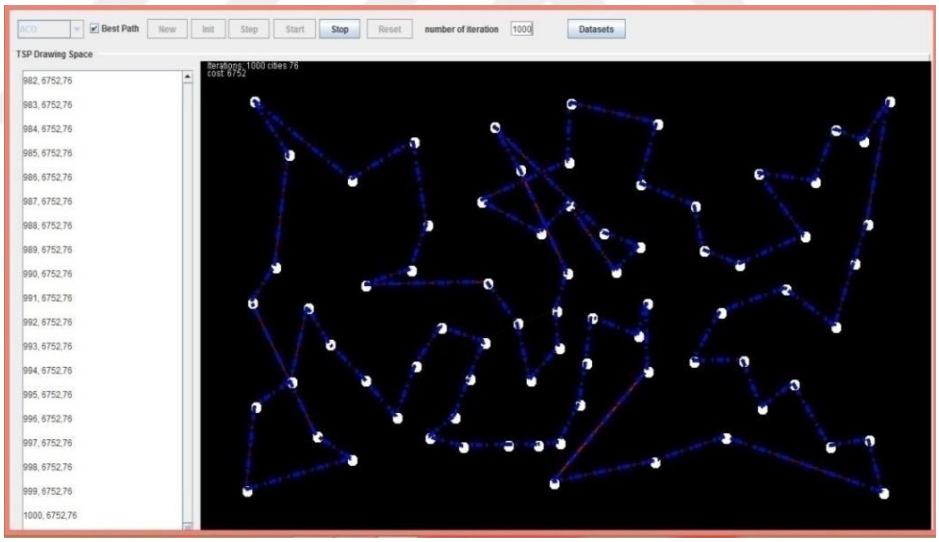
Şekil 4.19. GA Algoritmasının Harita-3 çözümü

Uygulama arayüzünden GA algoritması seçilerek uygulama çalıştırıldıktan sonra Şekil 4.19'daki GA çözümü elde edilmiştir. Mavi çizgiler en kısa yol çözümünü, yeşil çizgiler olası diğer yolları ifade etmektedir.



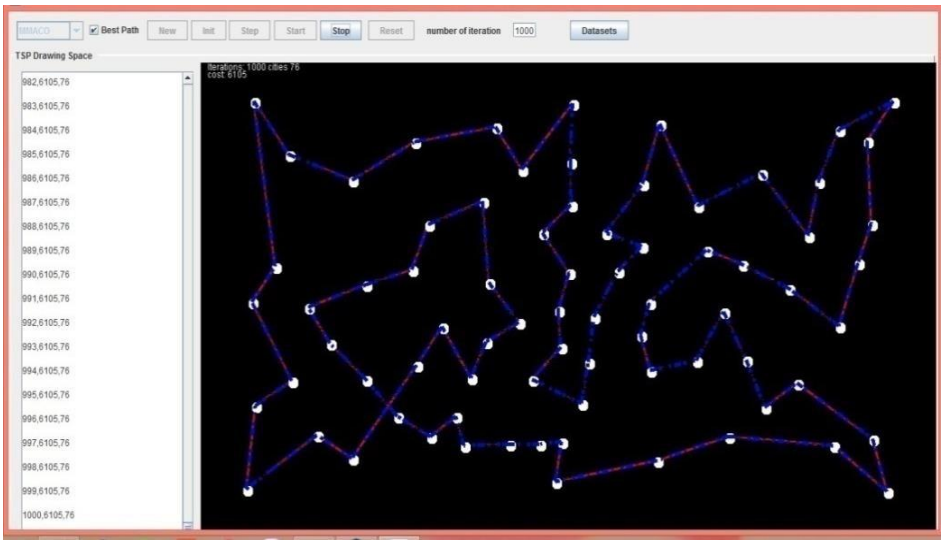
Şekil 4.20. AS Algoritmasının Harita-3 çözümü

Şekil 4.20'de uygulama arayüzünden AS seçilerek Harita-3 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



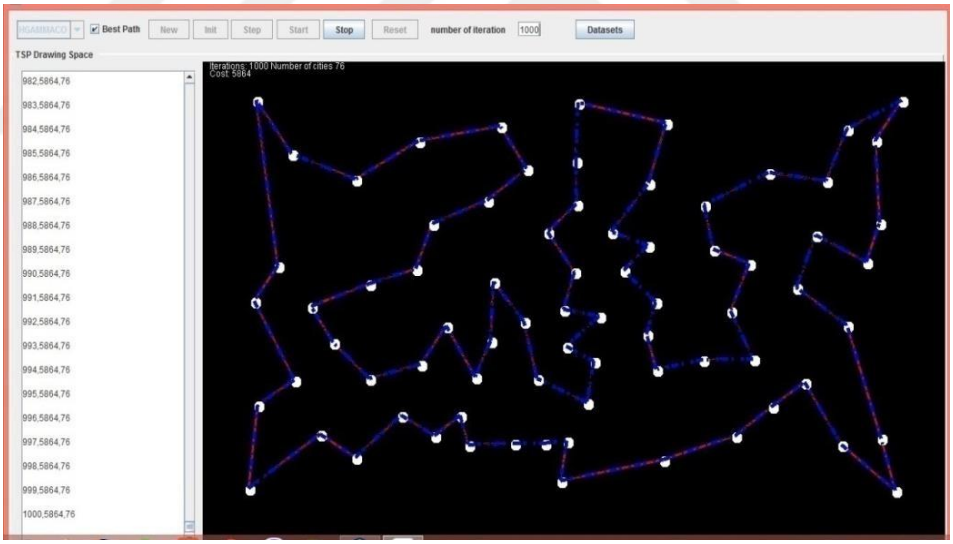
Şekil 4.21. ACS Algoritmasının Harita-3 çözümü

Şekil 4.21'da uygulama arayüzünden ACS seçilerek Harita-3 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



Şekil 4.22. MMAS Algoritmasının Harita-3 çözümü

Şekil 4.22'de uygulama arayüzünden MMAS seçilerek Harita-3 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



Şekil 4.23. HGAMMAS Algoritmasının Harita-3 çözümü

Şekil 4. 23'de ise, bu çalışmada tasarlanan HGAMMAS'nın Harita-3 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir

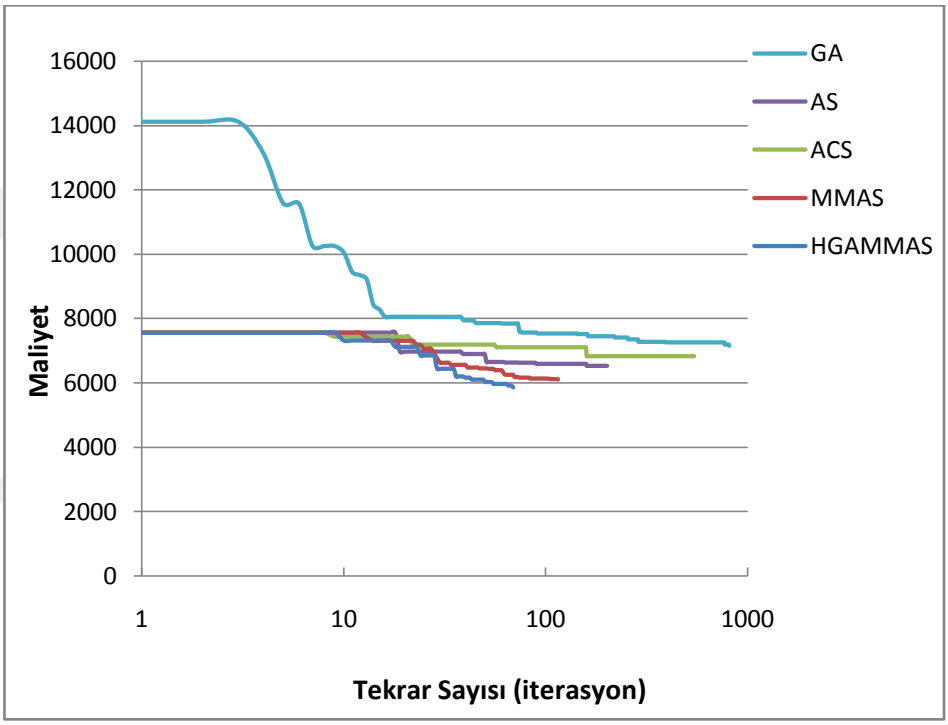
Çizelge 4.3'de GA, AS, ACS, MMAS ve HGAMMAS algoritmalarının Harita-3 için iterasyon sayısı ve maliyet performans ayrıntıları görülmektedir.

Çizelge 4.3 GA, AS, ACS, MMAS ve HGAMMAS Algoritmalarının Harita-3 için performans ayrıntıları

Harita-3 performansı ayrıntılı verileri					
#iterasyon	**GA maliyet**	**AS maliyet**	**ACS maliyet**	**MMAS maliyet**	**HGAMMAS maliyet**
0	14110	7557	7557	7557	7557
1	14110	7557	7557	7557	7557
-	7851	6627	7103	6251	5925
68	7851	6627	7103	6251	5864
-	7851	6627	7103	6179	5864
114	7543	6586	7103	6105	5864
-	7458	6524	6825	6105	5864
201	7458	6524	6825	6105	5864
-	7268	6485	6825	6105	5864
544	7268	6485	6825	6105	5864
-	7268	6485	6825	6105	5864
684	7268	6485	6752	6105	5864
-	7268	6485	6752	6105	5864
752	7268	6406	6752	6105	5864
-	7199	6406	6752	6105	5864
811	7161	6406	6752	6105	5864
-	7161	6406	6752	6105	5864
999	7161	6406	6752	6105	5864
1000	7161	6406	6752	6105	5864
ortalama maliyet	**7384.965**	**6526.584**	**6856.843**	**6155.898**	**5913.447552**
en yüksek maliyet	**14110**	**7557**	**7557**	**7557**	**7557**
en az maliyet	**7161**	**6406**	**6752**	**6105**	**5864**

76 düğüm (şehir)'li Harita-3 için GA 14110 maliyet ile çözüme başlarken, AS, ACS ve MMAS algoritmaları 7557 maliyet ile çözüme başlamıştır. GA 811. tekrarda 7161 maliyet ile en iyi çözümünü tamamlarken, AS algoritması 752. tekrarda 6406 maliyet ile, ACS algoritması 684. tekrarda 6752 maliyet ile ve

MMAS algoritması 114. tekrarda 6105 maliyet ile en iyi çözümlerini tamamlamışlardır. Harita-3 için maliyet bazında klasik algoritmalar arasından en iyi çözümü 6105 ile MMAS hesaplamıştır. Ancak, HGAMMAS algoritması 7557 maliyet ile başladığı çözüme 68. tekrarda 5864 maliyet ile tamamlamış, daha düşük maliyette ve daha kısa sürede hesaplama yaparak daha yüksek bir performans göstermiştir. Çizelge sonunda da ortalama maliyet, en yüksek maliyet ve en az maliyet bilgileri verilmiştir.

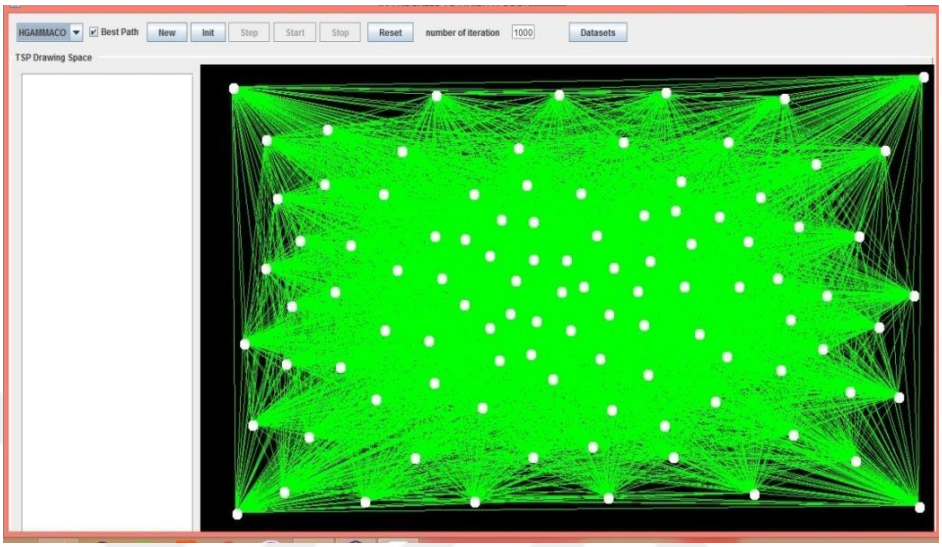


Şekil 4.24. GA, AS, ACS, MMAS ve HGAMMAS Algoritmalarının Harita-3 için iterasyon sayısı ve maliyet performansları

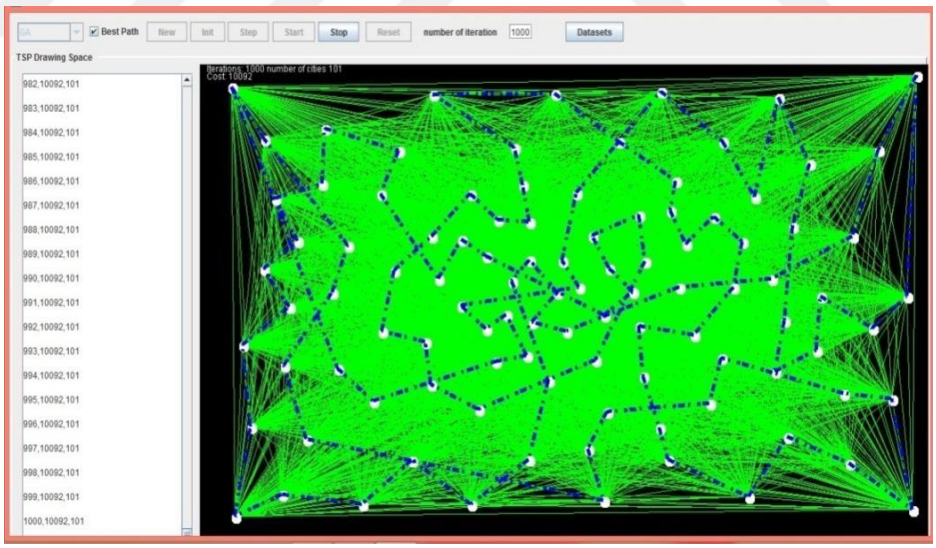
Şekil 4.24'de GA, AS, ACS, MMAS ve HGAMMAS algoritmalarının Çizelge 4.3'de verilen Harita-3 için iterasyon sayısı ve maliyet performansları grafiği görülmektedir.

4.3.4. 101 Düğüm (Şehir)'li Harita-4 denemesi

Şekil 4.25'da görülen Harita-4 dördüncü deneme için 101 düğüm ile yapılandırılmıştır.

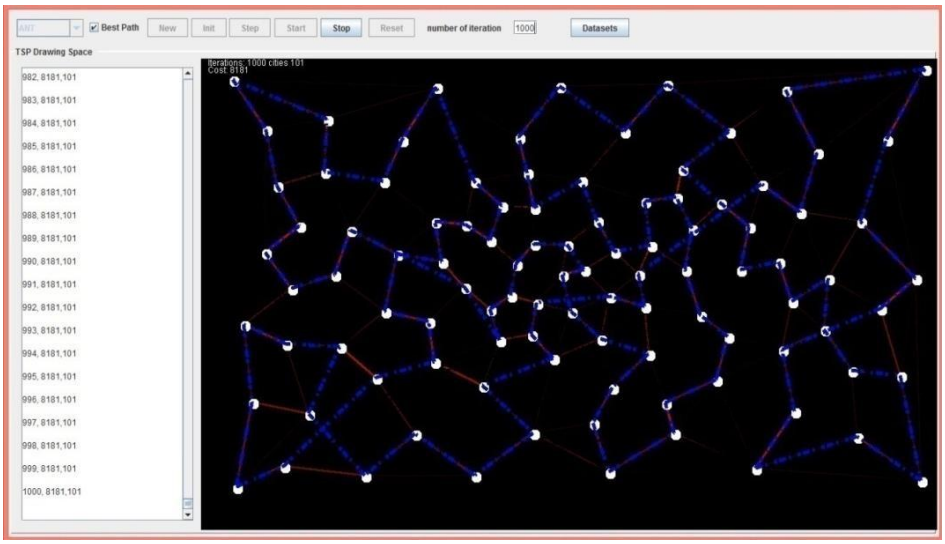


Şekil 4.25. Harita-4'ün deneme öncesi görüntüsü



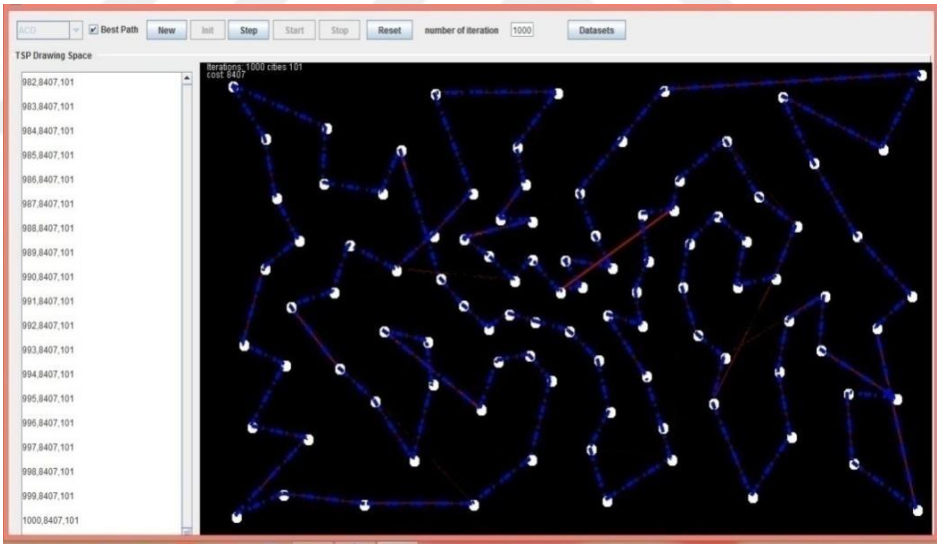
Şekil 4.26. GA Algoritmasının Harita-4 çözümü

Uygulama arayüzünden GA algoritması seçilerek uygulama çalıştırıldıktan sonra Şekil 4.26'deki GA çözümü elde edilmiştir. Mavi çizgiler en kısa yol çözümünü, yeşil çizgiler olası diğer yolları ifade etmektedir.



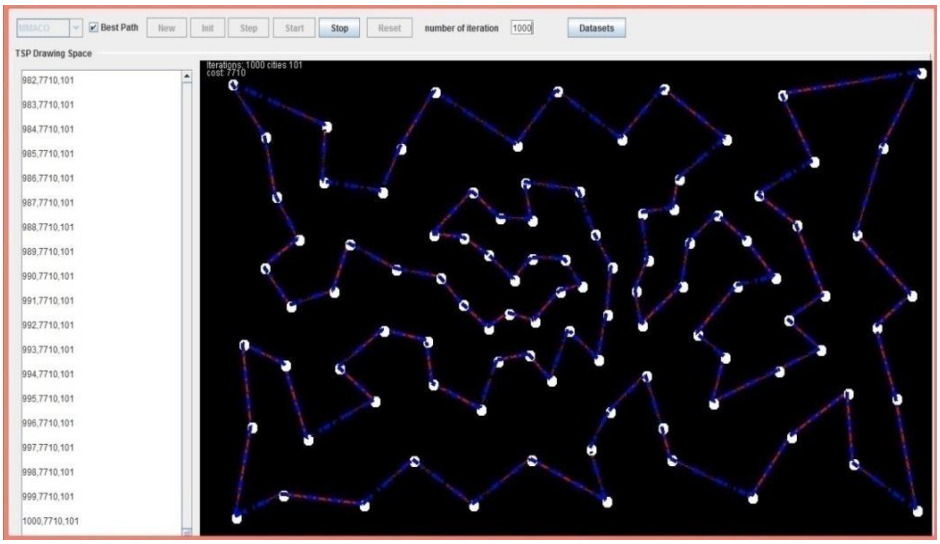
Şekil 4.27. AS Algoritmasının Harita-4 çözümü

Şekil 4.27'de uygulama arayüzünden AS seçilerek Harita-4 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



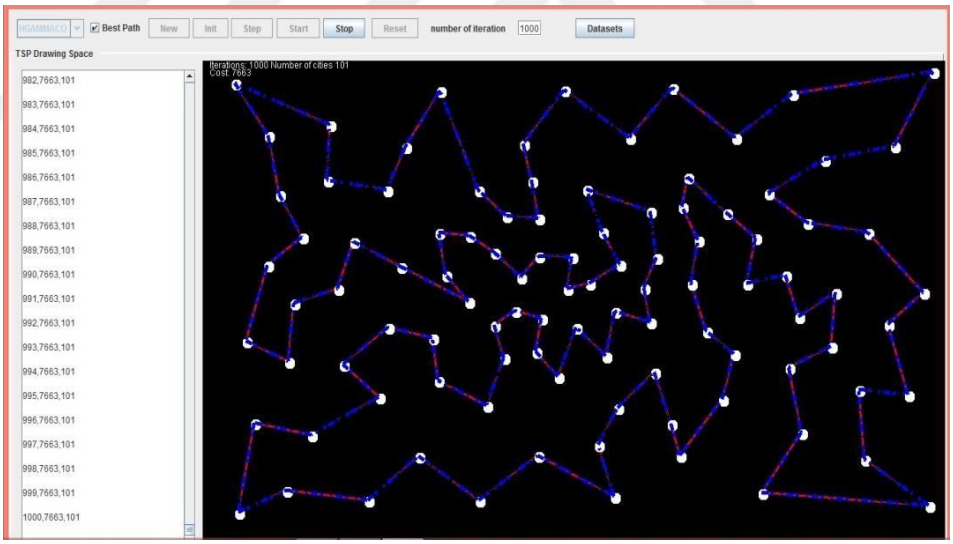
Şekil 4.28. ACS Algoritmasının Harita-4 çözümü

Şekil 4.28'da uygulama arayüzünden ACS seçilerek Harita-4 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



Şekil 4.29. MMAS Algoritmasının Harita-4 çözümü

Şekil 4.29'de uygulama arayüzünden MMAS seçilerek Harita-4 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



Şekil 4.30. HGAMMAS Algoritmasının Harita-4 çözümü

Şekil 4.30'de ise, bu çalışmada tasarlanan HGAMMAS'nın Harita-4 için hesplan en kısa yol çözümü mavi çizgiler ile görülmektedir

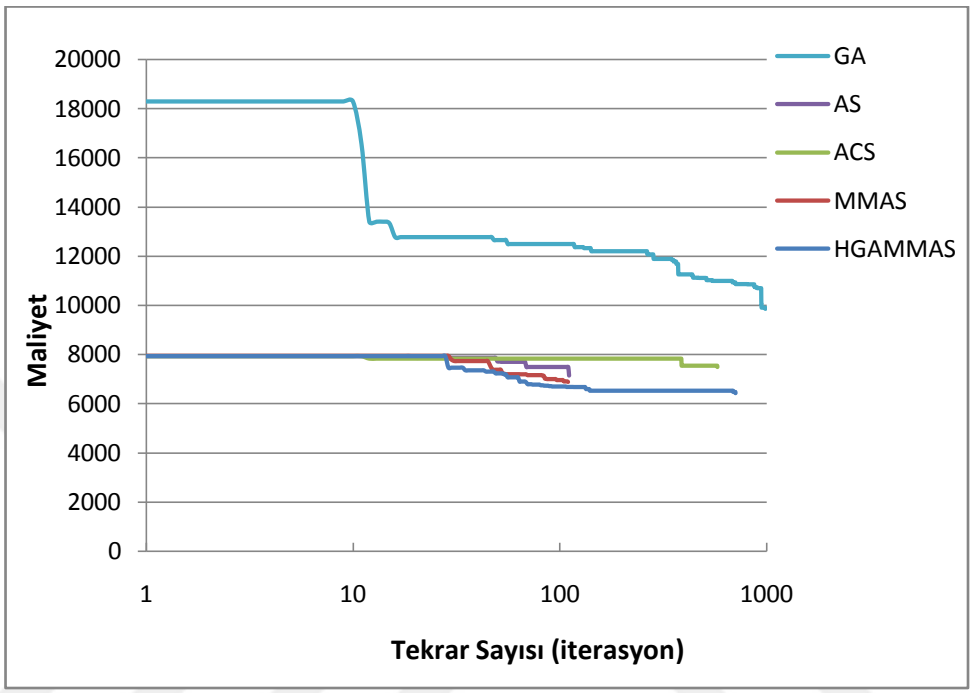
Çizelge 4.4'de GA, AS, ACS, MMAS ve HGAMMAS algoritmalarının Harita-4 için iterasyon sayısı ve maliyet performans ayrıntıları görülmektedir.

Çizelge 4.4 GA, AS, ACS, MMAS ve HGAMMAS Algoritmalarının Harita-4 için performans ayrıntıları

Harita-4 performansı ayrıntılı verileri					
#iterasyon	**GA maliyet**	**AS maliyet**	**ACS maliyet**	**MMAS maliyet**	**HGAMMAS maliyet**
0	18300	7934	7934	7934	7934
1	18300	7934	7934	7934	7934
2	18300	7934	7934	7934	7934
-	12493	7485	7833	6895	6669
108	12493	7485	7833	6895	6669
109	12493	7485	7833	6873	6669
110	12493	7131	7833	6873	6669
-	10991	7131	7538	6873	6515
577	10991	7131	7484	6873	6515
-	10861	7131	7484	6873	6465
708	10861	7131	7484	6873	6415
-	9901	7131	7484	6873	6415
985	9852	7131	7484	6873	6415
-	9852	7131	7484	6873	6415
999	9852	7131	7484	6873	6415
1000	9852	7131	7484	6873	6415
ortalama maliyet	**11440.38**	**7193.81**	**7630.288**	**6932.738262**	**6565.252747**
en yüksek maliyet	**18300**	**7934**	**7934**	**7934**	**7934**
en az maliyet	**9852**	**7131**	**7484**	**6873**	**6415**

101 düğüm (şehir)'li Harita-4 için GA 18300 maliyet ile çözüme başlarken, AS, ACS ve MMAS algoritmaları 7434 maliyet ile çözüme başlamıştır. GA 985. tekrarda 9852 maliyet ile en iyi çözümünü tamamlarken, AS algoritması 110. tekrarda 7131 maliyet ile, ACS algoritması 577. tekrarda 7484 maliyet ile ve MMAS algoritması 109. tekrarda 6873 maliyet ile en iyi çözümlerini tamamlamışlardır. Harita-4 için maliyet bazında klasik algoritmalar arasından en iyi çözümü 6873 ile MMAS hesaplamıştır. Ancak, HGAMMAS algoritması

7934 maliyet ile başladığı çözüme 708. tekrarda 6415 maliyet ile tamamlamış, daha düşük maliyette hesaplama yaparak daha yüksek bir performans göstermiştir. Çizelge sonunda da ortalama maliyet, en yüksek maliyet ve en az maliyet bilgileri verilmiştir.

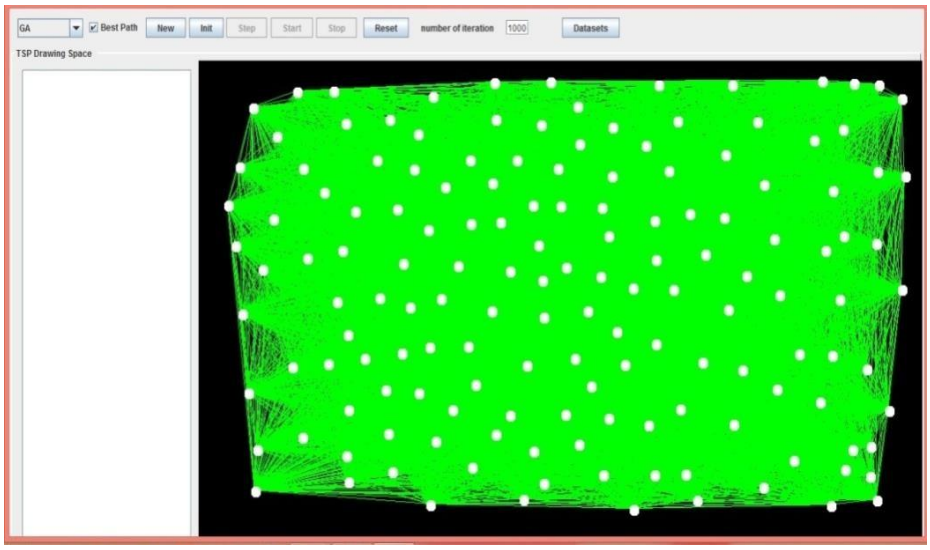


Şekil 4.31. GA, AS, ACS, MMAS ve HGAMMAS Algoritmalarının Harita-4 için iterasyon sayısı ve maliyet performansları

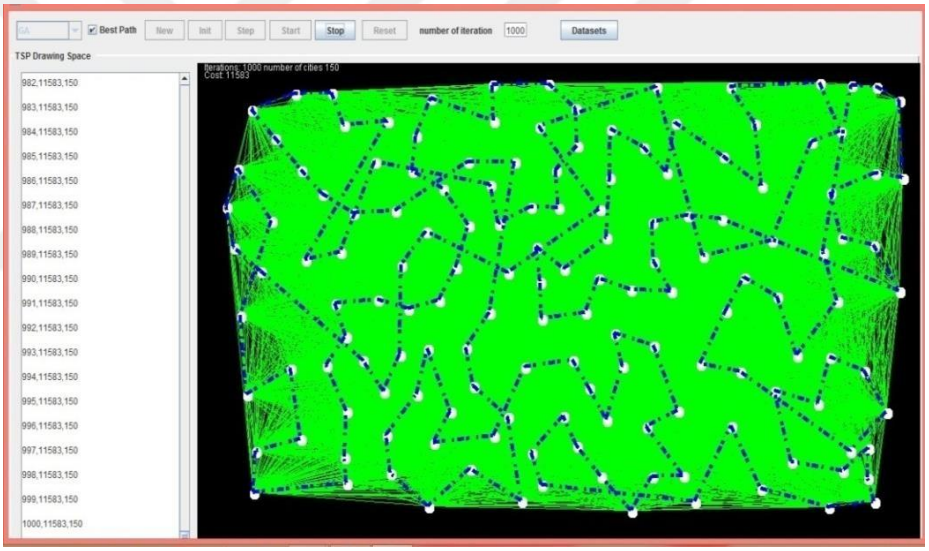
Şekil 4.31'de GA, AS, ACS, MMAS ve HGAMMAS algoritmalarının Çizelge 4.4'de verilen Harita-4 için iterasyon sayısı ve maliyet performansları grafiği görülmektedir.

4.3.5. 150 Düğüm (Şehir)'li Harita-5 denemesi

Şekil 4.32'da görülen Harita-5 beşinci deneme için 101 düğüm ile yapılandırılmıştır.

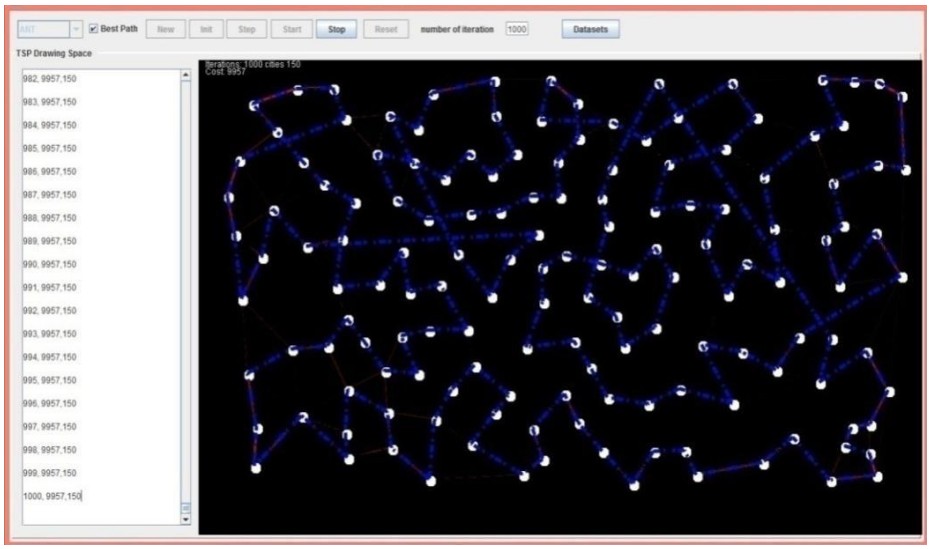


Şekil 4.32. Harita-5'in deneme öncesi görüntüsü



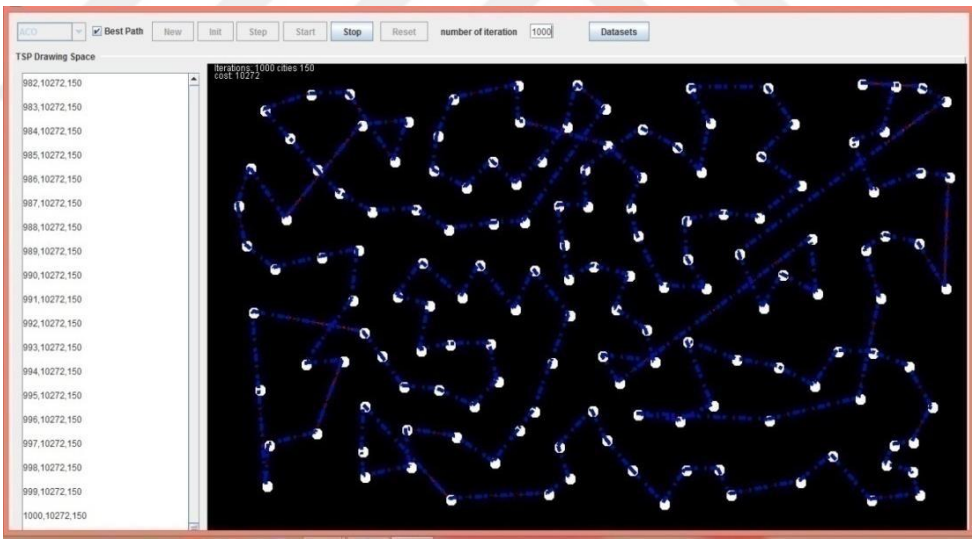
Şekil 4.33. GA Algoritmasının Harita-5 çözümü

Uygulama arayüzünden GA algoritması seçilerek uygulama çalıştırıldıktan sonra Şekil 4.33'deki GA çözümü elde edilmiştir. Mavi çizgiler en kısa yol çözümünü, yeşil çizgiler olası diğer yolları ifade etmektedir.



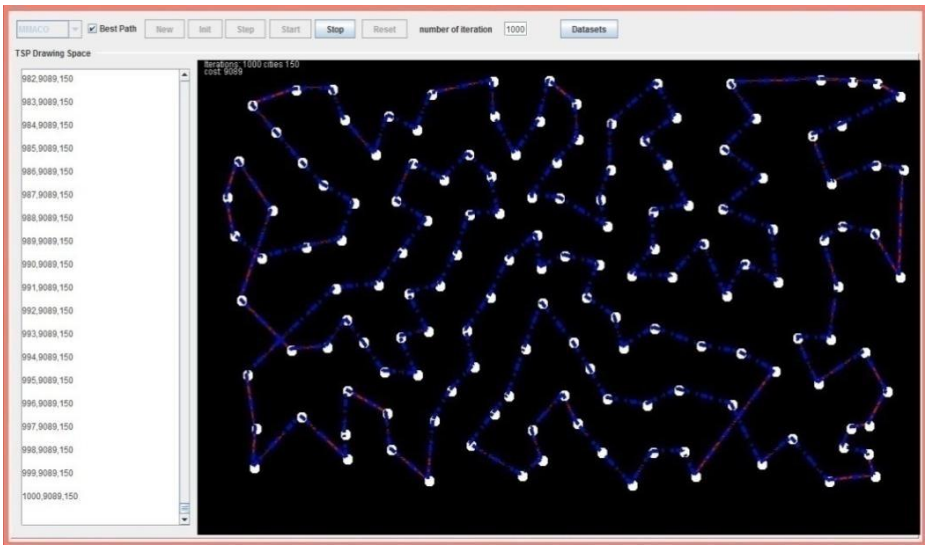
Şekil 4.34. AS Algoritmasının Harita-5 çözümü

Şekil 4.34'deki uygulama arayüzünden AS seçilerek Harita-5 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



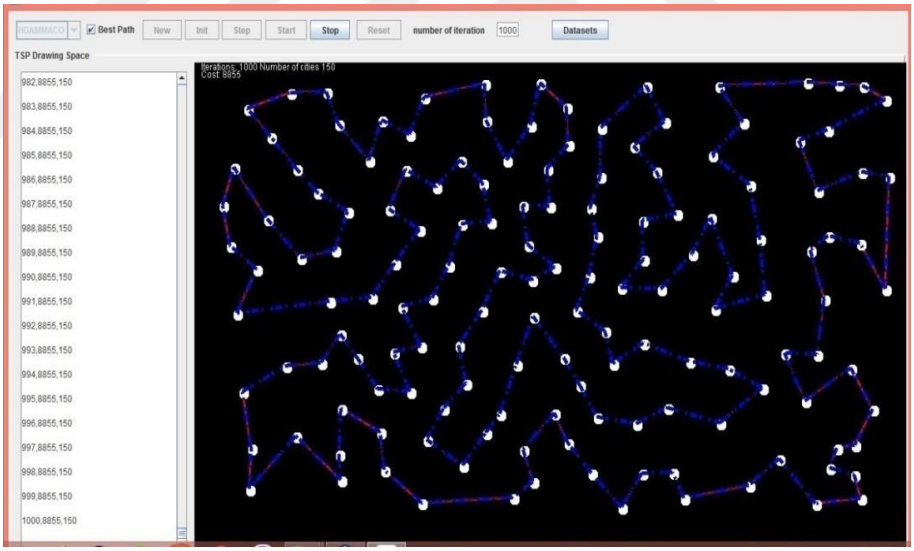
Şekil 4.35. ACS Algoritmasının Harita-5 çözümü

Şekil 4.35'de uygulama arayüzünden ACS seçilerek Harita-5 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.



Şekil 4.36. MMAS Algoritmasının Harita-5 çözümü

Şekil 4.36'de uygulama arayüzünden MMAS seçilerek Harita-5 için hesplan en kısa yol çözümü mavi çizgiler ile görülmektedir.



Şekil 4.37. HGAMMAS Algoritmasının Harita-5 çözümü

Şekil 4.37'de ise, bu çalışmada tasarlanan HGAMMAS'nın Harita-5 için hesaplanan en kısa yol çözümü mavi çizgiler ile görülmektedir.

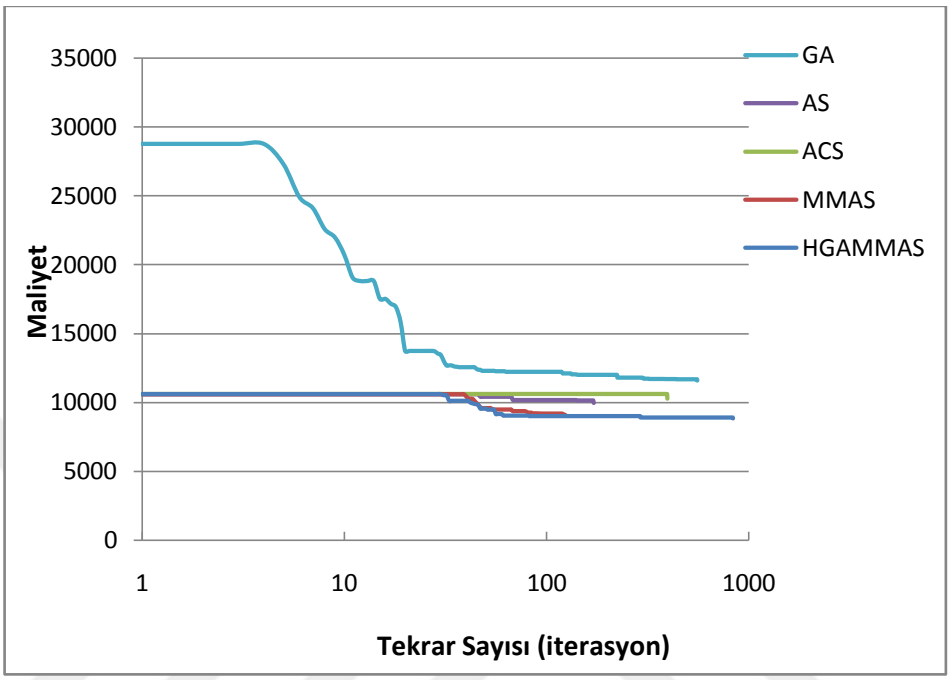
Çizelge 4.5'de GA, AS, ACS, MMAS ve HGAMMAS algoritmalarının Harita-5 için performans ayrıntıları görülmektedir.

Çizelge 4.5 GA, AS, ACS, MMAS ve HGAMMAS Algoritmalarının Harita-5 için iterasyon sayısı ve maliyet performans ayrıntıları

Harita-5 performansı detaylı verileri					
#iterasyon	**GA maliyet**	**AS maliyet**	**ACS maliyet**	**MMAS maliyet**	**HGAMMAS maliyet**
0	28776	10615	10615	10615	10615
1	28776	10615	10615	10615	10615
-	12101	10179	10615	9123	9021
123	12101	10179	10615	9089	9021
-	11997	10147	10615	9089	9021
170	11997	9957	10615	9089	9021
-	11689	9957	10615	9089	8919
396	11689	9957	10272	9089	8919
-	11689	9957	10272	9089	8919
555	11583	9957	10272	9089	8919
-	11583	9957	10272	9089	8919
832	11583	9957	10272	9089	8855
-	11583	9957	10272	9089	8855
999	11583	9957	10272	9089	8855
1000	11583	9957	10272	9089	8855
ortalama maliyet	**11953.68**	**10018.98**	**10407.69**	**9174.56444**	**9008.925075**
en yüksek maliyet	**28776**	**10615**	**10615**	**10615**	**10615**
en az maliyet	**11583**	**9957**	**10272**	**9089**	**8855**

150 düğüm (şehir)'li Harita-5 için GA 28776 maliyet ile çözüme başlarken, AS, ACS ve MMAS algoritmaları 10615 maliyet ile çözüme başlamıştır. GA 555. tekrarda 11583 maliyet ile en iyi çözümünü tamamlarken, AS algoritması 170. tekrarda 9957 maliyet ile, ACS algoritması 396. tekrarda 10272 maliyet ile ve MMAS algoritması 123. tekrarda 9089 maliyet ile en iyi çözümlerini tamamlamışlardır. Harita-5 için maliyet bazında klasik algoritmalar arasından en iyi çözümü 9089 ile MMAS hesaplamıştır. Ancak, HGAMMAS algoritması 10615 maliyet ile başladığı çözüme 832. tekrarda 8855 maliyet ile tamamlamış,

daha düşük maliyette hesaplama yaparak daha yüksek bir performans göstermiştir. Çizelge sonunda da ortalama maliyet, en yüksek maliyet ve en az maliyet bilgileri verilmiştir.



Şekil 4.38. GA, AS, ACS, MMAS ve HGAMMAS Algoritmalarının Harita-5 için iterasyon sayısı ve maliyet performansları

Şekil 4.38'de GA, AS, ACS, MMAS ve HGAMMAS algoritmalarının Çizelge 4.5'de verilen Harita-5 için iterasyon sayısı ve maliyet performansları grafiği görülmektedir.

4.4. Literatür İle Kıyaslama Denemeleri

4.4.1. GA, AS, ACS ve MMAS algoritmalarının literatur kıyaslamaları

Çalışmada incelenen GA, AS, ACS ve MMAS algoritma kodlamalarının, aynı algoritmaların literatürdeki diğer kodlamalar ile performans kıyaslaması da yapılmıştır. Kıyaslama için DataSet_ch150'yi çalışmalarında kullanan (Wang, Y.

,2014), (Puris, A., Bello, R. ve Herrea, F., 2010) ve literatürde bilinen en iyi maliyet verileri kullanılmıştır .

Çizelge 4.6 “Data Set-ch150” için yapılan literatür kıyaslaması performans sonuçları

DATASET ch150 (bilinen en iyi maliyet=6528)	GA	AS	ACS	MMAS
Wang, Y. (2014)	11908.5	-	-	-
Puris, A., Bello, R. & Herrea, F. (2010)	-	7219.8	6908.8	6867.6
Bu çalışmanın sonuçları	10225.5	7118.4	7250.6	6780.7

Çizelge 4.6’da görülen kıyaslama verilerinde en iyi çözümün 6528 olduğu, Wang’ın çalşmasında GA, Puris, A., Bello, R. & Herrea, F.’in çalışmasında AS, ACS ve MMAS algoritmalarının çalışıldığı görülmektedir. Bu çalışmadaki GA, AS ve MMAS algoritma maliyet performanslarının incelenen literature göre daha iyi olduğu, ancak en iyi çözüme ulaşamadığı görülmektedir.

4.4.2. GA, MMAS ve HGAMMAS algoritmalarının literatür kıyaslamaları

Literatürde GSP için geliştirilen algoritmaların performanslarının karşılaştırılabilir olmalarını sağlamak için TSPLIB veri setlerinin kullanıldığı görülmektedir (Reinhelt, G. TSPLIB, 2014).

Bu bölümünde, çalışmanın temelini oluşturan, melezlenen GA ve MMAS algoritmalarını çalışmalarında kullanan ve performanslarını TSPLIB kütüphanesindeki eil51, berlin52, eil76, rd100 ve kroA200 veri setleri ile değerlendiren (Somhom et al.algorithm, 1997), (Pasti ve De castro algorithim, 2006) ve (Masutti ve De Castro algorithm, 2009)'nin sonuçları ile bu çalışmada tasarlanan HGAMMAS algoritmasının performansını kıyaslamak için yapılan denemeler ve elde edilen bulgular sunulmuştur. Kıyaslamalar en az maliyet, ortalama maliyet ve standart sapma değerleri ile yapılmıştır. Aşağıda ayrıntıları verilen deneme sonuçlarının literatür ile kıyaslamalı özeti Çizelge 4.12’de verilmiştir.

eil51 veri seti ile denemesi:

Çizelge 4.7.'de GA, MMAS ve HGAMMAS Algoritmalarının eil51 veri seti için 10 denemede elde edilen en iyi çözüm maliyetleri görülmektedir. Her denemede 1000 iterasyon yapılmıştır.

Çizelge 4.7. GA, MMAS ve HGAMMAS algoritmalarının eil51 veri seti performansları

#deneme	GA maliyet	MMAS maliyet	HGAMMAS maliyet
1.	457	434	427
2.	476	430	427
3.	484	432	434
4.	480	430	426
5.	496	431	428
6.	484	428	427
7.	462	432	431
8.	474	441	432
9.	473	432	429
10.	454	428	427
en az maliyet	**454**	**428**	**426**
ortalama maliyet	**474**	**431.8**	**428.8**
standart sapma	**13.15716957**	**3.735713527**	**2.658320272**

Bu çalışmada melezleme yöntemi ile tasarlanan HGAMMAS algoritmasının eil51 veri setinde kendisini oluşturan GA ve MMAS algoritmalarına göre daha iyi çözüm maliyetinde performans gösterdiği görülmektedir.

HGAMMAS, eil51 veri seti ile denemesi sonunda MMAS'e göre %0.4, GA'ya göre %6.1 daha düşük maliyette bir sonuç hesaplamıştır.

berlin52 veri seti ile denemesi:

Çizelge 4.8'de GA, MMAS ve HGAMMAS algoritmalarının berlin52 veri seti için 10 denemede elde edilen en iyi çözüm maliyetleri görülmektedir. Her denemede 1000 iterasyon yapılmıştır.

Çizelge 4.8 GA, MMAS ve HGAMMAS Algoritmalarının berlin52 veri seti performansları

#deneme	GA maliyet	MMAS maliyet	HGAMMAS maliyet
1.	8962	7803	7542
2.	8712	7542	7542
3.	8926	7542	7542
4.	8314	7542	7542
5.	8760	7734	7542
6.	8467	7999	7542
7.	8769	7715	7542
8.	8407	7681	7542
9.	7865	7542	7542
10.	9060	7542	7542
en az maliyet	**7865**	**7542**	**7542**
ortalama maliyet	**8624.2**	**7664.2**	**7542**
standart sapma	**363.123426**	**154.1123688**	**0**

Bu çalışmada melezleme yöntemi ile tasarlanan HGAMMAS algoritmasının berlin52 veri setinde kendisini oluşturan GA ve MMAS algoritmalarına göre daha iyi çözüm maliyetinde performans gösterdiği görülmektedir.

HGAMMAS, berlin52 veri seti ile denemesi sonunda MMAS ile aynı, GA'ya göre %4.1 daha düşük maliyette bir sonuç hesaplamıştır.

eil76 veri seti ile denemesi:

Çizelge 4.9'da GA, MMAS ve HGAMMAS algoritmalarının eil76 veri seti için 10 denemede elde edilen en iyi çözüm maliyetleri görülmektedir. Her denemede 1000 iterasyon yapılmıştır.

Çizelge 4.9 GA, MMAS ve HGAMMAS Algoritmalarının eil76 veri seti için elde edilen en iyi çözüm maliyet performansları

#deneme	GA maliyet	MMAS maliyet	HGAMMAS maliyet
1.	649	559	545
2.	645	561	538
3.	658	553	543
4.	635	560	543
5.	588	552	555
6.	640	549	545
7.	655	553	551
8.	653	550	542
9.	624	556	548
10.	635	566	543
en az maliyet	**588**	**549**	**538**
ortalama maliyet	**638.2**	**555.9**	**545.3**
standart sapma	**20.552372**	**5.466056877**	**4.877385456**

Bu çalışmada melezleme yöntemi ile tasarlanan HGAMMAS algoritmasının eil76 veri setinde kendisini oluşturan GA ve MMAS algoritmalarına göre daha iyi çözüm maliyetinde performans gösterdiği görülmektedir.

HGAMMAS, eil76 veri seti ile denemesi sonunda MMAS'e göre %2, GA'ya göre %8.5 daha düşük maliyette bir sonuç hesaplamıştır.

rd100 veri seti ile denemesi:

Çizelge 4.10.'da GA, MMAS ve HGAMMAS algoritmalarının rd100 veri seti için 10 denemede elde edilen en iyi çözüm maliyetleri görülmektedir. Her denemede 1000 iterasyon yapılmıştır.

Çizelge 4.10. GA, MMAS ve HGAMMAS Algoritmalarının rd100 veri seti performanslar

#deneme	GA maliyet	MMAS maliyet	HGAMMAS maliyet
1.	11291	8471	7932
2.	10634	8768	8081
3.	10521	8191	7933
4.	10710	8331	8109
5.	11003	8435	7910
6.	10516	8546	8163
7.	10680	8204	7933
8.	12055	8473	8195
9.	10654	8182	8057
10.	10181	8428	8121
en az maliyet	**10181**	**8182**	**7910**
ortalama maliyet	**10824.5**	**8402.9**	**8043.4**
standart sapma	**523.274147**	**183.6109716**	**107.3894263**

Bu çalışmada melezleme yöntemi ile tasarlanan HGAMMAS algoritmasının rd100 veri setinde kendisini oluşturan GA ve MMAS algoritmalarına göre daha iyi çözüm maliyetinde performans gösterdiği görülmektedir.

HGAMMAS, rd100 veri seti ile denemesi sonunda MMAS'e göre %3.2, GA'ya göre %22.3 daha düşük maliyette bir sonuç hesaplamıştır.

kroA200 veri seti ile denemesi:

Çizelge 4.11.'da GA, MMAS ve HGAMMAS algoritmalarının kroA200 veri seti için 10 denemede elde edilen en iyi çözüm maliyetleri görülmektedir. Her denemede 1000 iterasyon yapılmıştır.

Çizelge 4.11. GA, MMAS ve HGAMMAS Algoritmalarının kroA200 veri seti performansları

#deneme	GA maliyet	MMAS maliyet	HGAMMAS maliyet
1.	60107	30815	29794
2.	51389	30474	29920
3.	55992	30704	29703
4.	62195	31127	30005
5.	55745	30985	29427
6.	60583	31853	30222
7.	52363	30748	29876
8.	53109	30802	30087
9.	61309	30318	30766
10.	58453	31902	30040
en az maliyet	**51389**	**30318**	**29427**
ortalama maliyet	**57124.5**	**30972.8**	**29984**
standart sapma	**3954.169536**	**528.7845182**	**353.3056593**

Bu çalışmada melezleme yöntemi ile tasarlanan HGAMMAS algoritmasının kroA200 veri setinde kendisini oluşturan GA ve MMAS algoritmalarına göre daha iyi çözüm maliyetinde performans gösterdiği görülmektedir.

HGAMMAS, kroA200 veri seti ile denemesi sonunda MMAS'e göre %2.9, GA'ya göre %42.7 daha düşük maliyette bir sonuç hesaplamıştır.

Sonuçların literatür ile karşılaştırılması:

Çizelge 4.12'de GPS çözümü performansını artırmak için bu araştırmada tasarlanan HGAMMAS algritmasını eil51, berlen52, eil76, rd100 ve kro200 veri setleri için hesapladığı en ksa yol maliyet sonuçları bilinen en iyi çözüm (BKS) ve aynı veri setleri ile çalışmış beş ayrı yayın sonuçları ile kıyaslanmıştır.

Çizelge 4. 12. Denenen HGAMMAS sonuçlarının diğer algoritma sonuçları ile karşılaştırılması

#Deneme		1	2	3	4	5
TSPLIB kütüphane data setleri		eil51	berlin52	eil76	rd100	kroA200
bilinen en iyi çözüm maliyeti (BKS-best known solution)		426	7542	538	7910	29368
Somhom et al.algorithm (1997)	ortalama maliyet	440.6	8025	562.3	8239	30416
	standart sapma	3.44	249	5.23	104	133
	en iyi maliyet	433	7715	552	8028	30144
Pasti ve De castro algorithim (2006)	Ortalama maliyet	438.7	8074	556.1	8253.9	30258
	standart sapma	3.52	270	8.03	149	343
	en iyi maliyet	429	7716	542	7947	29594
Masutti ve De Castro algorithm (2009)	ortalama maliyet	437.47	7932.5	556.33	8199.77	30190
	standart sapma	4.2	277.3	5.3	80.77	273.4
	en iyi maliyet	427	7542	541	7982	29600
Chen.S ve C. Chien(2011)	ortalama maliyet	427.27	7542	540.2	7987.6	29738.73
	standart sapma	0.45	0	2.94	62.06	356.07
	en iyi maliyet	427	7542	538	7910	29383
Ayman .S, Zulaiha .A , Abdul .H(2014)	ortalama maliyet	426.40	7542	538	7942.6	29438.20
	standart sapma	0.52	0	0	15.33	114.84
	en iyi maliyet	426	7542	538	7911	29368
HGAMMAS Algoritması	ortalama maliyet	**428.8**	**7542**	**545.3**	**8043.4**	**29984**
	standart sapma	**2.6**	**0**	**4.887**	**107.38**	**353.3**
	en iyi maliyet	**426**	**7542**	**538**	**7910**	**29427**

Çizelge 4.12'incelendiğinde HGAMMAS'ının eil51, berlen52, eil76, rd100 ve rd100 veri setlerinde bilinen en iyi çözüm miliyetinde hesplama yaptığı, aynı veri setlerini kullanan çalımalardan daha iyi sonuçlar verdiği görülmektedir. Ancak kroA200 veri setinde bilinen en iyi çözüme literatür çalışmalarına göre çok yaklaşan bir hesaplama yapabilmiştir.

5. SONUÇLAR

Bu araştırmada, sezgisel optimizasyon algoritmalarından GA ve MMAS'in özellikleri kullanılarak HGAMMAS adı verilen yeni bir melez algoritma tasarlanmış ve GSP çözüm performansı üzerindeki etkisi incelenmiştir.

Araştırma sürecinde Gezgin Satıcı Optimizasyon Problemi ve matematiksel modeli incelenmiştir. Sezgisel algoritmalardan Genetik Algoritma ve Karınca Koloni Optimizasyonu algoritmaları olan AS, ACS ve MMAS Matematiksel modelleri ve aralarındaki farklılıklar incelenmiştir. Bu konularla ilgili literatür taraması yapılmıştır. Bu algoritmalar Java platformunda kodlanarak ortak bir arayüzde birleştirilmiştir. Tasarlanan arayüz ve algoritmalara ait program kodları "https://github.com/raedalbadri" web adresinde açık kaynak kod olarak araştırmacıların erişimine sunulmuştur.

Önce, üretilen GA, AS, ACS ve MMAS algoritma kodları 10 düğümlük basit bir harita üzerinde geçerlilik (validation) denemesine tabi tutulmuştur. Dört algoritma da aynı (1679) maliyette en kısa yolu hesaplayarak birbirlerini doğrulamışlardır. Daha sonra, bu dört algoritma rastgele üretilen 36, 56, 76, 101 ve 150 düğüm (şehir)'den oluşan farklı 5 harita üzerinde denenerek Karınca Koloni Optimizasyonu algoritmalarının maliyeti en düşükten fazlaya doğru MMAS, AS, ACS biçiminde performans göstediği tespit edilmiştir. Performansı en yüksek olan MMAS ile GA melezlenerek HGAMMAS algoritması tasarlanmıştır. Son olarak, HGAMMAS'nin kendisini oluşturan GA ve MMAS'ye göre GSP üzerindeki performans farkını görmek için aynı 5 harita ve TSPLib kütüphanesindeki eil51, berlin52, eil76, rd100 ve kroA200 veri setleriile denenmiştir. HGAMMAS'in MMAS'e göre %3.2'ye, GA'ya göre %42.7'ye kadar daha düşük maliyette iyi sonuçlar hesapladığı görülmüştür.

HGAMMAS performansının literatürdeki diğer alternatif algoritma çözümleri ile karşılaştırmak için ise TSPLib kütüphanesindeki eil51, berlin52, eil76, rd100 ve kroA200 veri setleri ile tekrar denenmiştir. Çizelge 5.1'deki değerlendirme sonuçlarından HGAMMAS algoritma performansının eil51, berlin52, eil76 ve

rd100 veri setlerinin literatürdeki "bilinen en iyi değerler" ile aynı olduğunu ortaya çıkarmaktadır. kroA200 veri setinde ise (Somhom et al.algorithm, 1997), (Pasti ve De castro algorithim.,2006) ve, (Masutti ve De Castro algorithm.,2009) çalışmalarından daha iyi sonuç verdiği görülmektedir. kroA200 veri setinde (Chen.S ve C. Chienç.,2011) ve (Ayman .S, Zulaiha .A , Abdul .H., 2014) sonuçlarının HGAMMAS'den daha iyi sonuçlar verdiği görülmektedir.

Çizelge 5.1. HGAMMAS performansının literatür ile karşılaştırılma özeti

TSPLIB kütüphane data setleri	bilinen en iyi çözüm maliyet	Somhom et al. algorithm (1997)	Pasti ve castro algorithm (2006)	Masutti ve Castro algorithm (2009)	Chen ve Chienç., algorithm (2011)	Ayman, ve Abdul , algorithm (2014)	HGAMMAS
eil51	426	433	429	427	427	426	426
berlin52	7542	7715	7716	7542	7542	7542	7542
eil76	538	552	542	541	538	538	538
rd100	7910	8028	7947	7982	7910	7911	7910
kroA200	29368	30144	29594	29600	29383	29368	29427

Araştırma sonunda GSP üzerinde yüksek performans gösteren algoritmaların melezlenerek daha iyi sonuçlar elde edilebileceğini göstermiştir. Ancak melezlenecek algoritmaların uygulanacak probleme ve operatörlerinin birbirlerinin matematiksel modellerine uygun olması veya uygunlaştırılması gerekmektedir.

İleriki çalışmalarda, GSP çözümünde diğer sezgisel algoritmaların melezleştirilmesine yönelik araştırmalar ve başka problemlerin çözümlerine uygun melez sezgisel algoritma tasarımları yapılabilir.

KAYNAKLAR

Al Salami, N. M., 2009. Ant colony optimization algorithm. UbiCC Journal,4(3), 823-826.

Bajaj, R., & Malik, V., 2016. A Review on Optimization with Ant Colony Algorithm. Journal of Network Communications and Emerging Technologies (JNCET), 6(7).

Baraglia, R., Hidalgo, J. I., & Perego, R., 2001. A hybrid heuristic for the traveling salesman problem. IEEE Transactions on Evolutionary Computation,5(6), 613-622.

Bar-Cohen, Y., 2005. Biomimetics: biologically inspired technologies. CRC Press.

Blum, C., 2005. Ant colony optimization: Introduction and recent trends.Physics of Life reviews, 2(4), 353-373.

Botee, H. M., & Bonabeau, E., 1998. Evolving ant colony optimization.Advances in complex systems, 1(02n03), 149-159.

Chen, Shyi-Ming, and Chih-Yao Chien., 2011. Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. Expert Systems with Applications 38.12 .14439-14450.

Cordon, O., Herrera, F., & Stützle, T.,2002. A review on the ant colony optimization metaheuristic: Basis, models and new trends. In Mathware & Soft Computing.

Dalip, N. K., 2011. An Efficient Hybrid Genetic Algorithm for Performance Enhancement in solving Travelling Salesman Problem. International Journal on Computer Science and Engineering, 3(11), 3502.

Dorigo, M., & Gambardella, L. M.,1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on evolutionary computation, 1(1), 53-66.

Dorigo, M., & Stützle, T., 2003. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In Handbook of metaheuristics (pp. 250-285). Springer US.

Dorigo, M., 1992,. Optimization, Learning and Natural Algorithms (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

Dorigo, M., ana Stützle, T., 2009. Ant colony optimization: overview and recent advances. Techreport, IRIDIA, Universite Libre de Bruxelles.

Dorigo, M., Di Caro, G., & Gambardella, L. M., 1999. Ant algorithms for discrete optimization. Artificial life, 5(2), 137-172.

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

Dorigo,M., Maniezzo, V. and Colorni, A., 1991, Positive feedback as a search strategy, Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

Dorigo. M., and L.M. Gambardella., 1997. Ant colonies for the traveling salesman problem, BioSystems, vol. 43, no. 2.

Dorigo.M and Stutzle.T., 2009. Ant Colony Optimization: Overview and Recent Advances, Artificial Intelligence, pp.1-34.

Duan, H., & Yu, X., 2007 . Hybrid ant colony optimization using memetic algorithm for traveling salesman problem. In International Symposium on Approximate Dynamic Programming and Reinforcement Learning (pp. 92-95). IEEE.

Dwivedi, V., Chauhan, T., Saxena, S., & Agrawal, P., 2012. Travelling Salesman Problem using Genetic Algorithm. IJCA Proceedings on Development of Reliable Information Systems, Techniques and Related Issues, (1), 25.

Gambardella, L. M., De Luigi, F., & Maniezzo, V., 1998. Ant colony optimization. Istituto Dalle Molle di Studi sull'Intelligenza Artificiale Galleria.

Glover, Fred W, and Gary A. Kochenberger, eds., 2006. Handbook of metaheuristics. Vol. 57. Springer Science & Business Media.

Glover. F., 1989. Tabu search, ORSA Journal on Computing.

Haldun Sural, 2003. Gezgin Satıcı Problemi, Matematik Dünyası.

Hingrajiya, K. H., Gupta, R. K., & Chandel, G. S., 2012. An ant colony optimization algorithm for solving travelling salesman problem. International Journal of Scientific and Research Publications, 2(8), 1-6.

Hlaing, Z. C. S. S., & Khine, M. A., 2011. An ant colony optimization algorithm for solving traveling salesman problem. In International Conference on Information Communication and Management, Singapore, IPCSIT (Vol. 16).

Hodge, C., 1874. What is Darwinism? (Vol. 1). Scribner, Armstrong.

Holland, J. H., 1975. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press.

Joshi, G., 2014, Review of Genetic Algorithm: An Optimization Technique, International Journal of Advanced Research in Computer Science and Software Engineering, vol.4, no.4, pp.802-804.

Jünger, M., Reinelt, G., & Rinaldi, G., 1995. The traveling salesman problem.Handbooks in operations research and management science, 7, 225-330.

Kao, Y. T., & Zahara, E., 2008. A hybrid genetic algorithm and particle swarm optimization for multimodal functions. Applied Soft Computing, 8(2), 849-857.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P., 1983. Optimization by simmulated annealing. science.

Kuşcu, Ö., Küçüksille, E.U., 2011, Heuristic Methods in Vehicle Routing Systems, Elektronika ir Elektrotechnika, January, No.1, 65-70.

Mahajan, R., & Kaur, G., 2013. Neural networks using genetic algorithms.International Journal of Computer Applications, 77(14).

Maniezzo, V., Mingozzi, A., & Baldacci, R., 1998. A bionomic approach to the capacitated p-median problem. Journal of Heuristics, 4(3), 263-280.

Mastorakis, N., Bulucea, A., Tsekouras G., 2015, Computational Problems in Science and Engineering, ISBN 978-3-319-15765-8, Springer Book.

Masutti, T. A., & de Castro, L. N.,2009. A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem.Information Sciences, 179(10), 1454-1468.

Mavrovouniotis, M., & Yang, S., 2015. Training neural networks with ant colony optimization algorithms for pattern classification. Soft Computing, 19(6), 1511-1522.

Mavrovouniotis, M., Müller, F. M., & Yang, S., 2016 . Ant Colony Optimization With Local Search for Dynamic Traveling Salesman Problems.

Merlot, L. T., Boland, N., Hughes, B. D., & Stuckey, P. J., 2002. A hybrid algorithm for the examination timetabling problem. In International Conference on the Practice and Theory of Automated Timetabling (pp. 207-231). Springer Berlin Heidelberg.

Mıtras, B., & Aboo, A. K., 2014. Hybrid of Genetic Algorithm and Continuous Ant Colony Optimization for Optimum Solution.

Mishra, R., & Jaiswal, A., 2012. Ant colony optimization: A solution of load balancing in cloud. International Journal of Web & Semantic Technology, 3(2), 33.

Monteiro .M., Fontes. D, and Fontes.F., 2012. Ant Colony Optimization: a literature survey, Economics and managements, no.474, pp.1-28.

Moscato, P., & Cotta, C., 2003. A gentle introduction to memetic algorithms. In Handbook of metaheuristics (pp. 105-144). Springer US.

Onwubolu, G. C., & Babu, B. V., 2013. New optimization techniques in engineering (Vol. 141). Springer.

Pan, W., & Wang, L., 2009. An Ant Colony Optimization Algorithm Based on the Experience Model. In International Conference on Natural Computation (Vol. 3, pp. 13-18). IEEE.

Pasti, R., & de Castro, L. N., 2006. A neuro-immune network for solving the traveling salesman problem. In The 2006 IEEE International Joint Conference on Neural Network Proceedings (pp. 3760-3766). IEEE.

Penev, M. K. V. S. S., 2005. Genetic operators crossover and mutation in solving the TSP problem. In International Conference on Computer Systems and Technologies.

Puris, A., Bello, R., & Herrera, F. 2010. Analysis of the efficacy of a Two-Stage methodology for ant colony optimization: Case of study with TSP and QAP. Expert Systems with Applications, 37(7), 5443-5453.

Putha, R., Quadrifoglio, L., & Zechman, E., 2012. Comparing ant colony optimization and genetic algorithm approaches for solving traffic signal coordination under oversaturation conditions. Computer-Aided Civil and Infrastructure Engineering, 27(1), 14-28.

Reinhelt, G., 2014. TSPLIB: a library of sample instances for the TSP (and related problems) from various sources and of various types. URL: http://comopt. ifi. uniheidelberg. de/software/TSPLIB95.

Saiyed, A. R. 2012. The Traveling Salesman problem.

Shang, G., Xinzi, J., & Kezong, T., 2007. Hybrid algorithm combining ant colony optimization algorithm with genetic algorithm. In 2007 Chinese Control Conference (pp. 701-704). IEEE.

Shi, X. H., Liang, Y. C., Lee, H. P., Lu, C., & Wang, L. M., 2005. An improved GA and a novel PSO-GA-based hybrid algorithm. Information Processing Letters, 93(5), 255-261.

Solnon, C., & Bridge, D., 2006. An ant colony optimization meta-heuristic for subset selection problems. System Engineering using Particle Swarm Optimization, Nova Science, 729.

Somhom, S., Modares, A., & Enkawa, T., 1997. A self-organising model for the travelling salesman problem. Journal of the Operational Research Society,48(9), 919-928.

Srour, A., Othman, Z. A., & Hamdan, A. R., 2014. A Water Flow-Like Algorithm for the Travelling Salesman Problem. Advances in Computer Engineering.

Streichert, F., 2002. Introduction to evolutionary algorithms. paper to be presented Apr, 4.

Stützle, T., and Hoos, H. H., 2000. MAX–MIN ant system. Future generation computer systems.

Sultan Kuzu, Onur Önay, Uğur Şen, Mustafa Tunçer, Bahadır Fatih Yıldırım, Timur Keskintürk., 2014 .Gezgin satıcı problemlerinin metasezgiseller ile çözümü , İstanbul Üniversitesi İşletme Fakültesi Dergisi, Cilt:43, Sayı:1, 1–27 ISSN: 1303–1732.

Sutton, R. S., and Barto, A. G., 1998. Reinforcement learning: An introduction(Vol. 1, No. 1). Cambridge: MIT press.

Takahashi, R., 2009. A hybrid method of genetic algorithms and ant colony optimization to solve the traveling salesman problem. In Machine Learning and Applications. ICMLA'09. International Conference on.IEEE.

Tang K.S, Man K.F, Kwong S, He Q., 1996. Genetic algorithms and their applications. In IEEE SignalProcessing Magazine.

Vishwakarma. H. D., 2012. Genetic Algorithm based Weights Optimization of Artificial Neural Network, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering , vol.1, no.3, pp. 206-212.

Wang, Y. ,. 2014. The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem. Computers & Industrial Engineering,70, 124-133.

Zukhri, Z., & Paputungan, I. V., 2013. A hybrid optimization algorithm based on genetic algorithm and ant colony optimization. International Journal of Artificial Intelligence & Applications, 4(5), 63.

ÖZGEÇMİŞ

Adı Soyadı : Raed Ashraf kamil AL-BADRI

Doğum Yeri ve Yılı : IRAK, 1982

Medeni Hali : Evli

Yabancı Dili : Türkçe, İngilizce, Arapça

E-posta : raed.albadri@gmail.com

Eğitim Durumu

Lise :Irak, Samarra, 2002

Lisans : Irak Tikrit Üniversitesi**,** Bilgisayar Bilimleri Bölümü, 2007

Mesleki Deneyim

Çalışan eğitimi Irak'ta 2008-…….. (halen)