KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

KARINCA KOLONİLERİ ALGORİTMASI İLE İŞ AKIŞ PROBLEMLERİNİN OPTİMİZASYONU

YÜKSEK LİSANS

Elif ÇATAL

Ana Bilim Dalı: Elektronik ve Bilgisayar Eğitimi

Danışman: Yrd. Doç. Dr. Mehmet YILDIRIM

KOCAELİ, 2009

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

KARINCA KOLONİLERİ ALGORİTMASI İLE İŞ AKIŞ PROBLEMLERİNİN OPTİMİZASYONU

YÜKSEK LİSANS TEZİ

Elif ÇATAL

Tezin Enstitüye Verildiği Tarih: 22 MAYIS 2009

Tezin Savunulduğu Tarih: 03 TEMMUZ 2009

Tez Danışmanı
Yrd.Doç.Dr. Mehmet YILDIRIM
(.....)

Üye
Doç.Dr. Melih İNAL
(.....)

Üye
Yrd.Doç.Dr. Erbil AKBAY
(.....)

KOCAELİ, 2009

ÖNSÖZ VE TEŞEKKÜR

Bu tezin hazırlanmasında ve akademik çalışmalarımın her safhasında danışmanlığımı yapan, değerli hocam ve danışmanım Yrd. Doç. Dr. Mehmet YILDIRIM' a, deneyim ve bilgi gibi birçok konuda desteğini esirgemeyen Ege Üniversitesi Bilgisayar Mühendisliği Ana Bilim Dalında görev yapmakta olan hocam Arş. Gör. Doğan AYDIN' a, hayatımın her anında sabırla anlayışla, tüm konularda desteklerini esirgemeyen, sıkıntılı zamanlarımda yürek genişletici davranışlarından dolayı aileme ve Kocaeli Üniversitesi Endüstri Mühendisliği Ana Bilim Dalında görev yapmakta olan arkadaşım Arş. Gör. Hatice ERİŞ' e, sabrı, anlayışı ve sevgi dolu muameleleri dolayısıyla sevgili eşime teşekkürlerimi sunarım.

Ayrıca Yurtiçi Yüksek Lisans Burs Programına kabulleri ile desteklerini esirgemeyen TÜBİTAK' a sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SEMBOLLER

$\Delta\tau_{ij}^{bs}$: En iyi tura ait her bir ij yolunun feromon miktarı
C_{bs}, : En iyi turu yapan karıncanın tur uzunluğu
C_i : Tamamlanma zamanı
cl : Aday listesindeki diğer işler
C_{max} : Toplam çizelge süresi
d_{ij} : i ve j şehirleri arasındaki uzaklık
d_i : Teslim tarihi
F_i : Akış zamanı
J_i^k : k karıncasının i şehrindeyken ziyaret etmesi gereken şehir seti
L_i : Gecikme zamanı
N : Düğüm (şehir) sayısı
n / m : n iş sayısı, m makine sayısı
N_i^k : K karıncası i şehrindeyken henüz ziyaret etmediği şehir seti
p_{ij} : İşlem süresi
Pj : Toplam işlem süresi
Q : 0-1 aralığında rastgele seçilen bir değişkendir
r_i : Hazır olma zamanı
S : Tavlama benzetimi için çözüm kümesi
T : Tavlama benzetimi için sıcaklık değeri
T^{bs} : En kısa turu yapan en iyi karıncanın turu
T_{max} : En fazla gecikme
W_{ij} : Ağırlık
α : Feromon miktarının bağıl ağırlıklandırması
β : Şehirler arasındaki uzaklığa ait bağıl önem
η_{ij} : Sezgisel fonksiyon
ξ : $0< \xi < 1$ aralığında olan bir parametre
$\tau_{ij}^{k}(t)$: t. tur sonunda k karıncasının kullandığı her (i, j) yoluna bıraktığı feromon miktarı

Kısaltmalar

ACO : Ant Colony Optimization
ACS : Ant Colony System
AS_{rank} : Rank Temelli Karınca Sistemi
CACO : Continuous ACO
DDR : Due Date Rule
EAS : Elitist Ant System

EKİS : En Kısa İşlenme Zamanı
EUİS : En Uzun İşlenme Süresi
F : Seri İş Akışlı
G : Karmaşık İş Akışlı
GKKO : Global Karınca Kolonisi Algoritması
GSP : Gezgin Satıcı Problemi
İ : İş
KDS : Komşu Düğüm Seçimi
KKA : Karınca Kolonileri Algoritması
KKS- : Karınca Koloni Sistemi
KS-AS : Karınca Sistemi-Ant System
LB : Taillard Tablolarındaki Problemlerin En Kısa Çözüm Süresi
LPT : Longest Processing Time
MMAS : Max- Min Karınca Sistemi
NEH : Nawaz, Enscore, Ham
NP : Non-Polinomal
O : Operasyon
P : Polinomal
PACO : Paralel Karınca Sistemi
PTİA : Permütasyon Tipi İş Akış
QAP : Kuadratik Atama Problemi
SO : Self Organization
SPT : Shortest Processing Time
TACO : Touring Ant Colony Optimization
TB : Tavlama Benzetimi
UB : Taillard Tablolarındaki Problemlerin Optimum Çözüm Süresi

KARINCA KOLONİLERİ ALGORİTMASI İLE İŞ AKIŞ PROBLEMLERİNİN OPTİMİZASYONU

Elif ÇATAL

Anahtar Kelimeler: Üretim Planlama, İş Akış, Karınca Kolonileri Algoritması, Karınca Koloni Sistemi, FRB3 Algoritması

Özet: Üretim sistemlerinde iş akış problemi, belli bir performans ölçütünü eniyileyecek iş akış sırasını, başka bir deyişle tezgahlara gelen işlerin yapılma sırasını, belirlemek olarak bilinir. Son yıllarda oldukça fazla araştırma yapılan alanlardan biri olan karınca kolonileri algoritmalar, iş akış problemlerini çözmek için kullanılan en etkili yöntemlerden biridir.

Karınca kolonileri algoritmaları, ilk olarak, Dorigo ve arkadaşları tarafından, ileri sürülmüştür [1]. Bu tekniğin geliştirilmesinde gerçek karınca kolonilerinin gıda arama tekniklerinden faydalanılmıştır [2].

Bu tezde, öncelikle iş akışı ile ilgili temel kavramlar açıklanmış daha sonra, konunun daha iyi anlaşılabilmesi amacıyla, karınca optimizasyonunun gezgin satıcı problemine nasıl uygulanabileceği anlatılmıştır. Daha sonra karınca optimizasyonunun açılımı olan karınca koloni sistemi, Rad, S.F., Ruiz, R., Boroojerdian, N. tarafından geliştirilen FRB3 algoritması [3] ile iyileştirilerek yeni bir algoritma önerilmiştir ve önerilen yeni algoritmaya KKS&FRB3 adı verilmiştir.

Bu çalışmada, önerilen KKS&FRB3 algoritmasıyla oluşturulan uygulama yazılımları, Taillard' ın örnek problem tablolarındaki 100 farklı problem üzerinde çalıştırılmış ve gerçekleştirilen test işlemleri sonucunda, klasik karınca koloni sistemi algoritmasının ortalama hata oranı %1,42 iken, önerilen KKS&FRB3 algoritmasının ortalama hata oranı %0,99' dur.

Önerilen KKS&FRB3 algoritması, karşılaştırma işlemi için kullanılan diğer metotlara göre, paralel karınca sisteminden sonra, ortalama hata oranı en düşük ikinci algoritmadır. Algoritma kendi içinde değerlendirildiğinde, 20 iş ve 5 makine içeren problemlerin çözümünde %0,10 hata oranı ile en başarılı, 100 iş ve 20 makine içeren problemlerin çözümünde %2,31 hata oranı ile en başarısız sonuç elde edilmiştir.

Bu tezde, klasik karınca kolonileri algoritmalarının, FRB3 ile iyileştirilerek daha başarılı sonuçlar elde ettiği gözlemlenmiştir.

ANT COLONY ALGORITHM FOR OPTIMIZING WORKFLOW PROBLEMS

Elif ÇATAL

Keywords: Production Planning, Work Flow, Ant Colony Algorithm, Ant Colony System, FRB3 Algorithm

Abstract: In production systems, it is known that job sequencing problem is the determination of workflow sequence (jobs that come to workbenches for operating) which improves a certain performance criterion. For realizing this situation, ant colony optimization algorithm is one of the most effective methods.

Ant colony algorithms were first proposed by Dorigo and colleagues [1]. Real ant colonies' seed searching technics were used for developing this method [2].

In this thesis, as a first thing the fundamental concepts were explained and then it has been explained that how ant optimization can be applied to the TSP problem. Then, ant colony system algorithm, which is a variant of ant colony optimization algorithm, was improved with FRB3 algorithm, which was developed by Rad, S.F., Ruiz, R., Boroojerdian, N.[3], and it was called KKS&FRB3.

Application softwares were created by KKS&FRB3 algorithm, working on flowshop problems and results have been observed. Taillard benchmarks were used for the selection of the flowshop problems. As a result of the test process performed on 100 different problems while the average error rate of ant colony system is 1.42% the average of the KKS&FRB3 algorithm is 0.99%.

Proposed KKS&FRB3 algorithm, according to other methods which were used for comparing, has the second lowest average error rate. When algorithm is evaluated on its own, the most successful results were obtained with the 0.10% average error rate while solving problems including 20 jobs and 5 machines and the least successful results were obtained with the 2.3% average error rate while solving problems including 100 jobs and 20 machines.

In this thesis more successful results were observed by improving the classical ant colony systems algorithms with FRB3.

1. GİRİŞ

Günümüzde gelişen teknolojiye, artan ihtiyaçlara ve değişen ekonomik koşullara bağlı olarak işletmelerin bu şartlara uyum sağlamaları gerekmektedir. Bir işletmenin ayakta kalabilmesi ve verimli çalışabilmesi üretimden önce yapılan planlama ile mümkündür. Bu planlama ne kadar gerçek koşullara uyar ise sistem o kadar gerçekçi olur, az zamanda ve az kayıpla tamamlanır. Bir işletme için önemli olan unsurlar; müşteri taleplerini eksiksiz ve zamanında tamamlama, istenen kalitede mal üretme ve en az maliyetle en verimli şekilde çalışmadır. Bu unsurlar, işletmenin ancak üretim öncesi yapmış olduğu üretim ve işin hangi makinede ve hangi sırayla işleme konulması gerektiğine karar verme bir başka deyişle atölye çizelgelemenin kalitesi ile mevcuttur. Çünkü planlama, istenilen üretim hedeflerine en ekonomik ve en verimli şekilde ulaşabilmenin yoludur. Bunun için en küçük işletmeden en büyük işletmeye kadar tüm üreticiler bu planlamaya gerekli önemi vermekte ve bu çizelgelemelere maddi kaynak ayırmaktadırlar. Teknolojinin gelişmesiyle de üretim hatlarında kullanılan makinelerde tamamen elektronikleşmiştir. Dolayısıyla insan hataları en aza indirgenirken sistemde insan faktörünün azalmasıyla atölye çizelgelemelerinin önemi artmaktadır. Bir çizelgelemenin kalitesi gerçek uygulamada karşılaşılan problemlerin sayısal olarak çözümü ve bunun uygulanmasıyla mümkündür [4].

Çizelgeleme, üretim planlama ile beraber bir imalat sisteminin etkinlik ve verimliliğini belirleyen önemli bir işlevdir. Bu yüzden atölye düzeyi ne kadar iyi çizelgelenebilirse imalat sisteminin etkinlik ve verimliliği o oranda artmış olur. Üretim çizelgeleme kolay çözülebilir problem tipi değildir ve bunu etkileyen birçok unsur vardır. İş öncelikleri, teslim tarihleri, üretim adedi, öncelik kuralları gibi kısıtlar örnek olarak verilebilir.

Çizelgeleme literatürü; parametrelerin belirgin (deterministik) olduğu durumdan belirsiz (stokastik) olduğu duruma, tek makineden çok makineye, geliş sürecinin durağandan (statik) dinamiğe değiştiği çeşitli problem yapılarını kapsar [5].

Günümüz çizelgeleme yazılımları çeşitli algoritma yapıları kullanarak, sistemleri gant şeması adı verilen bir şema üzerinde göstermeye çalışmaktadır [6]. Ancak, atölye çizelgeleme problemi için yapılan programlar her ne kadar gerçek uygulamaya yaklaşmak isteseler de bir takım şartları yok saymaktadırlar. Bu da yapılan çizelgelemenin tam anlamıyla uygulanamamasına sebep olur. Bu sorunu halledebilmek için çeşitli algoritmalar ve yaklaşımlar geliştirilse de tam bir çözüm üretilememektedir [4].

Çizelgeleme yazılımları, işlemleri gerçekleştirecek en iyi kaynağı kullanıcının müdahalesine gerek kalmadan gerçekleştirebilir. Çizelgeleme yazılımları; çalışma düzeni, kapasite, önceliklendirme, iş yükü gibi birçok özelliğe göre işi yapabilecek en iyi kaynakların seçimini sağlar. Hemen her üretim sisteminin kendine özgü kısıtları mevcut olduğu için, çizelgelemede kullanılacak yazılımın, sisteme göre veya kişisel sıralama kurallarına göre en iyi çözüm yöntemini kullanması gerekmektedir.

Çizelgeleme problemleri optimizasyon problemleridir ve literatürde optimizasyon problemlerinin çözümünde kullanılan birçok teknik mevcuttur [4, 7].

Bu tekniklerden olan doğrusal programlamada, klasik yöntemlerle sonuca ulaşılabilirken, problem boyutunun büyümesi bu yöntemler ile çözümü imkansız hale getirmektedir [8].

Bu yüzden, bir takım sezgisel algoritmalarla problemlere çözüm aranmaya başlanmıştır. Bunlar; karınca kolonileri algoritması [1,6], genetik algoritmalar [9], tabu arama [10], tavlama benzetimi [11] gibi bir takım sezgisel yöntemlerdir.

Karınca algoritmaları ilk olarak Dorigo ve meslektaşları tarafından, gezgin satıcı problemi (GSP) ve kuadratik atama (QAP) gibi zor optimizasyon problemlerinin çözümü için geliştirilmiştir. Optimizasyon problemlerinin çözümü amacıyla karınca

algoritmaları üzerine birçok çalışma halen devam etmektedir. Karıncalar, görme duyularını kullanmadan, yiyecek kaynaklarından yuvalarına en kısa yolu bulma yeteneğine sahiptirler. Aynı zamanda, çevredeki değişime adapte olma yetenekleri vardır. Dış etkenler sonucu takip ettikleri mevcut yol artık en kısa yol değilse, yeni en kısa yolu bulabilmektedirler [12].

Problem çözümünün başlangıcında rastsal hareket eden karıncaların, izleri kontrol ederek yüksek olasılılıkla izlerin yoğun olduğu yönü takip etmesi otokatalitik bir davranış şeklidir ve karıncaların karşılıklı etkileşiminde sinerjik bir etki yaratmaktadır. Algoritma, karınca kolonilerinden esinlenerek geliştirildiği için sisteme, karınca sistemi (KS), algoritma ise karınca kolonileri algoritması (KKA) olarak adlandırılmaktadır. KKA günümüzde, optimizasyon problemlerinin çözümünde kullanılan güçlü bir algoritma haline gelmiştir.

Bu tez çalışmasında KKS&FRB3 algoritması kullanılarak, Taillard' ın 100 adet örnek iş akış problemleri çözülmüş, sonuçlar incelenmiştir.

Tezin ikinci bölümünde iş akış problemleri tanıtılmış, iş akış problemlerine ait atölye çizelgeleme problemleri için çözüm yöntemleri incelenmiştir.

Üçüncü bölümde, doğal karıncalar, bunlardan esinlenerek geliştirilen karınca sistemi, KKA anlatılmış ve karınca optimizasyon yöntemlerinden bahsedilmiştir.

Dördüncü bölümde karınca kolonileri algoritmasının çalışma prensibinin daha iyi anlaşılması amacı ile karınca kolonisi algoritmasının gezgin satıcı problemine uygulanması anlatılmıştır.

Beşinci bölümde iş akış problemlerinin, FRB3 algoritması ile iyileştirilmiş karınca koloni sistemi algoritmasıyla çözümü anlatılmıştır.

Sonuç bölümünde, önerilen KKS&FRB3 algoritmasının, iş akış problemlerine uygulanması sonucunda elde edilen sayısal veriler incelenmiş, farklı algoritmalarla elde edilen sonuçların karşılaştırmaları yapılmış ve algoritmanın daha da iyi sonuçlar üretebilmesi için önerilerde bulunulmuştur.

2. İŞ AKIŞ PROBLEMLERİ

2.1. Problemin Tanımı

Atölye çizelgeleme problemlerinin, kendine özgü belirlenmesi gereken ve atölye ortamını tanımlayan unsurların çözülmesi gerekmektedir. Dolayısıyla bu unsurların belirlenmesinden sonra atölye tipi veya ortamının özelliğine göre çeşitli çözüm yöntemleri uygulanmalıdır. Atölye çizelgelemeye ait problemin iyi anlaşılması ve ayrıntılarının önceden belirlenmesi gerekmektedir.

İşletmelerde siparişlerin zamanında teslim edilmesi ancak iyi hazırlanmış bir çizelgeleme ile mümkündür. Bunu sağlayabilmek için birçok araştırma yapılmış ve bu sorunu gerçek çözüme en yakın sonuçlar bulan bilgisayar programları geliştirilmiştir.

Bilgisayar ortamında yapılacak olan bir çizelgelemeden önce atölye ortamının belirlenmesi ve özelliklerinin bilinmesi gerekmektedir. Çünkü üretim çizelgeleme gibi atölye çizelgeleme problemleri de çok sayıda ve birbirlerinden farklı olmaktadır. Bu farklılıktan dolayı her atölye için geliştirilen çizelgelemeler birbirlerinden farklı olacaktır. Bu çeşitlilikten dolayı her türlü atölye ortamının çizelgeleme işlemini yapacak olan sistemlerin tasarlanması hem zorlaşmakta hem de yapımının imkansızlaşmasına neden olmaktadır.

Yapılacak olan çizelgelemede aşağıdaki sıralamalar takip edilmelidir [4].

1. Problemin anlaşılması,

2. Problemin tanımlanması,

3. Problemin daha önceden çözülmüş teorik ve uygulamalı problemlerle olan benzerliklerinin ve farklılıklarının belirlenmesi,

4. Problemin zorluk derecesinin belirlenmesi,

5. Zorluk derecesine göre probleme en uygun algoritma ya da çözüm yapısının belirlenmesi,

6. Problemin çözülmesi,

7. Çözümün değerlendirilmesi.

Bu adımların takibinden sonra elde edilen çözümün istenilen düzeyde olup olmadığının kararı verilmelidir. Çözüm eğer yeterli değilse; 5., 6., ve 7. adımlar için en uygun çözüm bulunana kadar işlemler tekrarlanmalıdır.

Problemin çözüm yöntemi belirlendikten sonra, problem bilgisayar ortamına aktarılır. Problem için veri girdi ve çıktılarının kararından sonra problemin çözüm yönteminin ya da algoritmasının oluşturulması gerekir [13].

2.2. İşlerin Geliş Türlerine Göre Sınıflandırma

2.2.1. Statik ortam

Belli sayıda iş aynı zamanda atölyede hazır bulunuyor gibi kabul edilir ve çizelgeleme buna göre yapılır. Bu andan sonra başka işlerin gelmesine izin verilmez. Bu tür problemler belirli (deterministik) olarak da düşünüldüklerinden işlem süreleri ve parametreler sabit olarak kabul edilir [4].

2.2.2. Dinamik ortam

Atölyede işlerin farklı geliş zamanları ve farklı miktarları olmaktadır. İşlerin tamamının özellikleri tam olarak bilinmediğinden geçmişteki deneyimlerden yararlanılarak iş özellikleri belirlenir. Dinamik problemler, belirsiz (stokastik) yapıdaki problemlerdir. İşlerin, atölyeye gelişleri rastlantısallık özelliği taşıdığından atölye kuyruk özellikleri göstermeye başlar. Makine herhangi bir zamanda boş kalınca ne yapılması gerektiği kararı önem kazanır [14].

2.3. Problemin Gerçek Uygulama Şartlarıyla Karşılaştırılması

Teorik olarak önerilen çözüm modelleri, gerçek uygulama şartları ile kıyaslandığında farklılıklar gösterir. Bu farklılıklar aşağıda belirtilmiştir [4]:

Teorik modellerde sistemde N adet iş vardır ve bu işler çizelgelendikten sonra problemin çözüldüğü varsayılır. Gerçekte ise, sisteme sürekli bir iş girişi olmaktadır. Bu gerekçeyle çizelgeleme yapılmadan önce yakın gelecek hakkında bilgi sahibi olunması gerekir. Fakat beklenmedik durumların ortaya çıkmasıyla, örneğin; işlerin artması, makinelerin bozulması, işler arasındaki öncelik değerlerinin değişmesi v.b., yapılmış olan çizelgeleme tam istenen sonuca ulaşamamaktadır. Bunu önlemek amacıyla sistemin dinamik yapıda olması sağlanabilir.

1. Sistem için geçerli bir çizelgeleme bulunabilir ancak beklenmedik olaylar karşısında çizelgede değişiklikler olabilir.

2. Uygulamadaki makine sistemleri çok daha karmaşıktır. Kısıtlayıcı etkenler çok daha fazla olmaktadır.

3. Matematiksel modellerde işlerin ağırlığının eşit olduğu kabul edilir. Fakat uygulamalarda bu ağırlık değerleri değişebilmektedir.

4. Matematiksel modellemelerde bir iş bir makinede yapılıyor ya da yapılmıyor gibi kesinlik mevcuttur. Uygulamada ise bazı işler yapım maliyetine göre o makinede yapılması tercih edilir ya da edilmez.

5. Modellemelerde, makinelerin her zaman hazır olduğu kabul edilir. Ancak uygulamalarda bu durum söz konusu değildir. Atölyenin işleyiş durumuna göre bu değişmektedir. Atölyede vardiya sistemi olabilir ve makineler günün belirli saatlerinde çalışıyor olabilir. Makineler bozulabilir veya bakıma alınabilir. Bunun gibi durumlarda o makinenin kullanımı mümkün olmamaktadır.

6. İşlem süresinin rastlantısal bir değişken olduğu belirsiz (stokastik) modellerde işlem süresi çoğu zaman sabit değildir. Uygulamada, işin öğrenilme etkisi ve makinenin zaman içindeki yıpranma durumu iş süresinin rastlantısal olmasına neden olur. Örnek olarak; bir makinedeki yeni bir işçinin zamanla o işe alışması ve pratik kazanması işin yapım süresini azaltırken, makinenin zamanla yıpranması işin yapım süresini uzatmaktadır. Modellemelerde bu tür sorun yokmuş gibi düşünülür.

Teorik modeller, uygulamada karşılaşılan bu tür sorunlara rağmen, matematiksel modellemelerde doğru sonucun bulunmasında ipuçları sağlamaktadırlar.

2.4. Çizelgelemede Kullanılan Öncelik Kuralları ve Kabuller

Çizelgeleme problemlerinde literatürde on farklı öncelik kuralları bulunmaktadır [15].

1. Erken teslim tarihli,

2. İlk giren ilk çıkar,

3. Kısa işlem zamanlı,

4. Kalan işlem zamanlı,

5. Toplam işlem zamanlı,

6. Dinamik değişkenli,

7. Rastlantısal öncelik kurallı,

8. Uzun işlem zamanlı,

9. Az operasyonu kalan öncelik kuralı,

10. Çok operasyonu kalan öncelik kuralı

Akış ve atölye tipi çizelgeleme problemlerinin çözümünde genel olarak aşağıdaki kabuller yapılmaktadır [16];

1. Tezgah hazırlık süreleri bilinmektedir ve işlem sürelerinin içerisinde yer almaktadır.

2. Bütün n adet iş sıfırıncı zamanda işlenmeye hazır durumda beklemektedir.

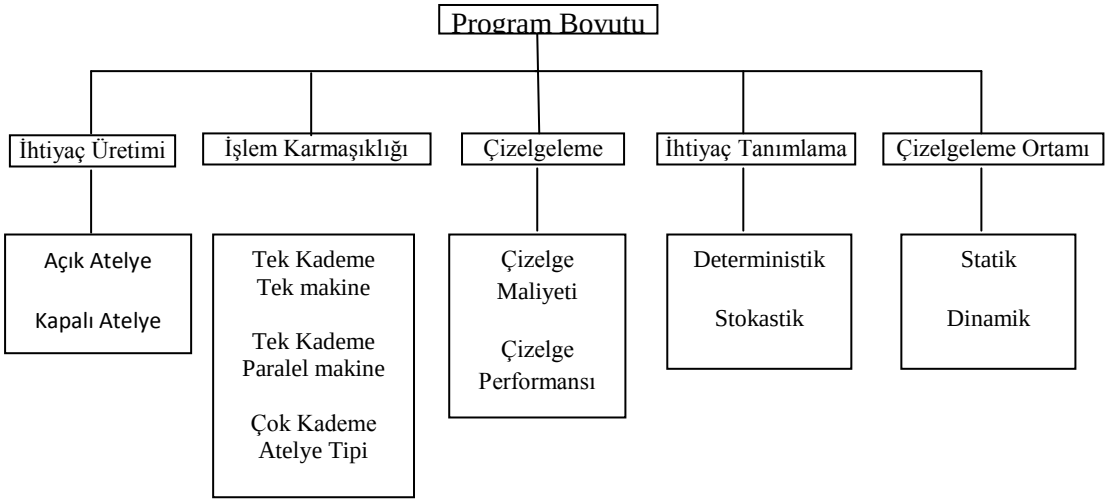
3. Her iş m operasyona ve her operasyonda farklı makinelere ihtiyaç duymaktadır.

4. Makinelerin önceden rezervasyon yapılmasına izin verilmez. Her işin ilk operasyonu önce başlar, ikinci operasyonun başlayabilmesi için birincinin tamamlanması gerekir.

5. Çizelgeleme periyodu boyunca makineler sabittir ve mevcut yerinde kapasiteleri aynıdır.

2.5. Üretim Çizelgeleme ve Sınıflandırılması

Üretim çizelgelemede dikkat edilmesi gereken unsurlar Şekil 2.1' de şema üzerinde belirtilmektedir [14].



Şekil 2.1: Çizelgelemede problem boyutu [4]

2.5.1. İhtiyaç üretimi

İhtiyaç üretiminde gereksinimler açık ve kapalı atölyede üretilir. Açık atölyede gereksinimler doğrudan doğruya müşteri siparişiyle üretilir. Kapalı atölyede ise gereksinimler stoktan karşılanır [4].

2.5.2. İşlem karmaşıklığı

Tek kademe, tek makine problemi; en basit problem biçimidir. Bütün işler sadece bir makinede yapılır. Bu işlem basit olmasına karşın diğer karmaşık problemlerin temelini oluşturmaktadır.

Tek kademe, paralel makineler problemi; her bir iş paralel makinelerden birisinde işlem görmektedir. Aynı işi yapan makinelerin sayısı fazladır. Cıvata üretimin yapıldığı atölye tipi buna örnek gösterilebilir.

Çok kademeli problemler; her bir işin işlem sırasında çok kesin bir sıranın olduğu problem tipidir. Her bir iş, makineler grubunda öncelik ilişkisine göre işlem görürler. Bu tip problemler akış tipi ve atölye tipi olmak üzere gruplandırılabilir.

İş çizelgeleme problemi; işin hangi makinede ve hangi sırayla işleme konulması gerektiğine karar verme sürecidir. Bu yaklaşımlar iki başlık altında incelenebilir. Atölye tipi çizelgeleme ve akış tipi çizelgelemedir.

Akış tipi çizelgelemede bütün işler aynı sırayla aynı makine grubunda işlenir. Atölye tipi çizelgeleme, sınıflandırmada en genel ve en karmaşık olan çizelgeleme problemidir. Belli bir işe ait işlem kademeleri sayısı hakkında hiçbir kısıt yoktur ve alternatif olarak kabul edilebilecek rotalar mevcuttur. Atölye tipi çizelgelemede her iş farklı makinelerde işlenmek üzere kendine özgü işlem ve işlem sıralarına sahiptir.

Atölye tipi üretim çizelgeleme probleminde m tane makinede (M) işlenmek üzere n tane iş (İ) mevcuttur. Her bir işin her bir makinede sadece ve sadece bir kez işlem gördüğü varsayılır. Makinede işin işlenmesine operasyon (O) denir. İşler makinelerde belirli bir sırayla işlenir. Genel atölye tipi üretim için teknolojik kısıtların oluşumuna dair bir sınırlama yoktur. Her iş kendi işlem sırasına sahiptir ve diğer işlerin işlem sıralarından bağımsızdır. Bununla birlikte bütün işler aynı sıraya sahip olduğunda oluşacak çizelgelemeye de akış tipi çizelgeleme olarak adlandırılır. Atölye tipi çizelgelemede ana unsur makineler ve bunlar üzerindeki işlerdir [4].

Her bir operasyon belli bir zaman uzunluğuna sahiptir. Bu zamanın içinde; makineyi ayarlama ve hazırlama zamanı (setup time), işi makineye taşımak için geçen sürede dahil edilmektedir.

Atölye tipi çizelgeleme problemini genelleştirmek için bazı tanımların yapılmasına gerek vardır [4]. Bunlar;

1. Her bir iş bütündür; iş farklı operasyonlardan oluşmasına rağmen aynı işin iki operasyonu hiçbir şekilde aynı anda işlenmez.

2. İş bölme yoktur; her bir operasyon başladığı zaman diğer operasyon o makinede başlatılmadan önce tamamlanmalıdır.

3. Her iş bir makinede bir tane olmak üzere m tane farklı operasyona sahiptir, işin aynı makinede iki defa işlem görme olasılığı hesaba katılmaz.

4. İş iptali söz konusu değildir.

5. İşlem zamanları çizelgeden bağımsızdır; işe ait makineyi ayarlamak için gereken zaman en son işlem gören işten bağımsızdır. Makineler arasında işleri taşımak için gereken zaman ihmal edilmektedir.

6. Ara stoğa izin verilir.

7. Makinenin her bir tipinden sadece bir tane vardır. Aynı işi yapan makineden sadece bir tane vardır. Böylelikle beklemekten kaçınmak için belli makinelerin çoğaltılması durumu ortadan kaldırılır.

8. Makineler boş kalabilir. Fakat bu durumda makinelerin verimlilik kısıtı ihmal edilir.

9. Hiçbir makine aynı anda birden fazla operasyonu işleyemez.

10. Makineler asla bozulmaz ve çizelgeleme periyodu boyunca kullanıma hazırdır.

11. Teknolojik kısıtlar önceden bilinir ve sabittir.

12. Rastlantısallık söz konusu değildir.

a) İşlerin sayısı bilinir ve sabittir.

b) Makinelerin sayısı bilinir ve sabittir.

c) İşlem zamanları bilinir ve sabittir.

d) Hazırlık zamanları bilinir ve sabittir.

e) Belli bir problemi tanımlamak için gereken her türlü nicel değerler bilinir ve sabittir.

2.5.3. Çizelgeleme kriterleri

İki tane ana kritere sahiptir. Bunlar çizelgeye ait maliyet ve etkinlik maliyetidir. Belli bir çizelgeye ait maliyet; üretim hazırlıklarıyla ilgili sabit maliyetleri, değişken ve fazla mesai maliyetlerini, stok maliyetlerini, siparişleri karşılayamamanın sonucunda oluşacak olan maliyetlerdir [4].

2.5.4. İhtiyaç tanımlama

İhtiyaç tanımlama, problemin nümerik değerlerin önceden bilinip bilinmemesiyle ilgilidir. Eğer tüm parametreler miktar olarak önceden biliniyorsa ve sabitse belirgin olarak tanımlanır. Aksi halde bilinmiyorsa belirsiz olarak tanımlanır. Örnek olarak her işin her bir kademedeki ve her bir makinedeki işlem süreleri önceden biliniyorsa bu tipteki atölye belirgin (deterministik) olarak tanımlanır. Aksi takdirde; işlem süreleri tam olarak bilinmiyorsa ve belli bir olasılık dağılımıyla rastgele değişken olursa bu tipteki atölye belirsiz (stokastik) olarak tanımlanır [4].

2.5.5. Çizelgeleme ortamı

Çizelgeleme ortamı, üretilecek gereksinimler için gerekli olan girdiler üzerine varsayımlarla ilgilenir. Çizelgeleme zamanı boyunca üretilecek gereksinimlerin miktarı ve buna bağlı olarak atölye ortamına giren işlerin miktarı belirlenir ve sonradan atölye ortamına ek bir iş girişi yapılmaz. Bu durumda atölye ortamı statik olarak tanımlanır. Diğer yandan çizelgeleme zamanı boyunca üretilecek olan gereksinimlerin miktarı ve buna bağlı giren iş miktarına sonradan ilave yapılabilecek

şekilde problem tanımlanabilir. Atölye ortamına herhangi bir anda ya da özel bir durumda yeni iş girdileri olabilir. Bu durumda atölye ortamı dinamik olarak tanımlanır.

Gerçek uygulamalarda ve çizelgelemelerde problemlerin çoğu belirsiz (stokastik) ve dinamiktir. Üretilen çözüm modellerinin çoğu ise belirli ve statik olarak kabul edilir. Böyle kabullerin sebebi ise; statik ve belirli problemleri anlamadan dinamik ve belirsiz (stokastik) problemleri anlamak zordur. Aynı zamanda günümüzdeki teknolojik gelişmelerin sonucunda işlem sürecine katılan elektronik devreler, robotlar ve mikro denetleyiciler sayesinde işlem sürelerinde belirlilik sağlanmıştır [4].

2.5.6. Performans ölçütleri

Çizelgelemede amaçları ifade etmek her zaman kolay değildir. Amaçlar çok karmaşıktır ve genellikle birbirleriyle bağdaşmazlar. Ancak çizelgelemede ne derece başarılı olunduğuna karar vermek için bir takım kriterleri tanımlamak gerekir. Aksi takdirde matematiksel olarak çizelge oluşturmak imkansızlaşır.

Örnek olarak; kararlaştırılmış teslim tarihlerine uymak zorunda kalınabilir. Aksi takdirde güvenirlilik kaybına uğranabilir ve finansal ceza maliyeti söz konusu olabilir. Bazen teslim tarihi önemli olmayabilir ve çizelgeleme zamanını uzunluğu enazlanmak istenebilir. Tüm işler tamamlandıktan sonra bazı makineler başka işler için kullanılabilir. Böylece makinelerin aylak (boş) kalma zamanları enazlanmak istenebilir. Aylak makine aylak sermaye demektedir. Bunlara ek olarak stok maliyeti enazlanmak istenebilir [14].

2.6. Problemin Zorluk Derecesinin Belirlenmesi

Problemin zorluk derecesinin bilinmesi problemin çözümü için en iyi yöntemin uygulanmasını sağlar. Polinomial (P) olan denklemler çözümlenmesi, incelenmesi kolay olan denklemlerdir ve kısa sürede sorunu çözen yöntemleri mevcuttur. Eğer bir denklem ya da sistem polinomial değilse (NP; Nonpolynomially Bounded)

çözümlenmesi zor sistemlerdir. Ancak NP (polinomial olmayan, kesin çözümü olmayan) problemler için ise kısa sürede gerçek çözümü bulan yöntemler mevcut değildir. Bu nedenle, NP problemler için gerçek çözüme en yakın sonucu bulmak amacıyla yaklaşık çözüm algoritmaları geliştirilmiştir [4].

Yaklaşık çözüm algoritmaları, problemin gerçek olmayan ancak geçerli bir çözümü (gerçek sonuca yakın olan) kısa sürede bulabilirler. Pratikte karşılaşılan problemlerin çoğu için kesin çözümden ziyade kısa sürede yaklaşık bir çözümün bulunması istenmektedir. Bu nedenle, pratikte karşılaşılan NP problemlerin çözümünde probleme özgü olarak sezgisel yöntemler yardımıyla geliştirilen algoritmalar kullanılır. Polinomial algoritmalar pratikteki problemlerin çözümünde iyi performans gösterirler. NP problemlerinde kullanılan polinomial algoritmalar ise sorunu çözememektedir. Bu tip problemlerin kesin sonuçlarına makul sürelerde ulaşılmadığından; yerel arama ve rastlantısal arama ile yaklaşık çözümler elde edilir. Rastlantısal arama yöntemleri; yerel arama yöntemlerinin, yerel optimumda takılıp kalma dezavantajlarını ortadan kaldırmak için geliştirilmiştir.

Problemlerin çözümü için kullanılan algoritmaların sonuca kısa sürede ulaşması esastır. Bir algoritmanın en yaygın performans ölçütü, algoritmanın sonucu bulana kadar ki geçen süredir.

NP problemlerinin çözümünde kullanılan yöntemler [17];

1. Yerel arama metotları

2. Yapay sinir ağları

3. Tabu araştırmaları

4. Tavlama benzetimi

5. Karınca kolonileri iyilemesi

6. Yapay bağışıklık sistemleri

7. Genetik algoritmalardır.

2.7. Atölye Çizelgeleme

2.7.1. Çizelgelemenin tanımı

Çizelgeleme; " İşlerin belirlenen bir sırada gerçekleşmesi için program yapılması ve bu programın çeşitli kriterler altında çeşitli performans ölçülerini iyilemesi faaliyetidir."

Çizelgelemede iş; herhangi bir kaynak tarafından o anda üzerinde çalışılan kısım olarak ta tanımlanır.

Çizelgelemede kaynak makine olarak kabul edilir ve makine; herhangi bir zamanda üzerinde en çok bir faaliyetin gerçekleştirildiği kaynak olarak tanımlanır.

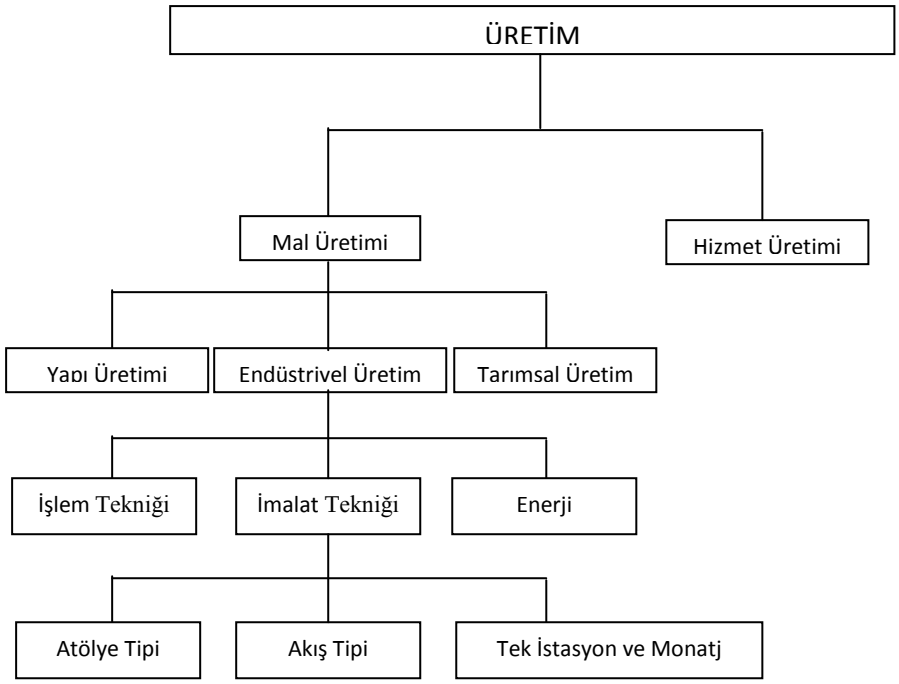
Çizelgelemenin bir tanımı da, belirlenen görevleri yerine getirmek için kaynakların zaman içindeki kullanımını göstermeye denir. Ayrıca hangi kaynakların ne zaman ve nasıl kullanılacağı da belirtilmektedir. Çizelgelemedeki asıl amaç, işi daha az kaynakla ve daha az sürede istenilen kriterlere uyacak biçimde problemin çözümünü göstermektir [18].

2.7.2. Atölye tipi çizelgelemenin üretim işlem çizelgelemedeki yeri

Üretim işlemlerinden biri olan atölye tipi üretim, organizasyon yönünden karmaşık üretim tiplerinden biridir. Atölye tipi üretimin yeri Şekil 2.2' de gösterilmiştir.

Atölye tipi üretimde çok amaçlı işleme makineleri birbirinden farklı işleri tek veya seri kademeler halinde işlerler. Takım tezgahı ve kalıp imalatı; tipik bir atölye tipi

üretimdir. Akış tipi üretim ise atölye tipi üretimden elde edilen ürünlerin seri halde yapılmasını amaçlar [14].



Şekil 2.2: Atölye tipi üretimin yeri [4]

Atölye çizelgelemede, gerçek uygulamada karşılaşılan bir takım kısıtlayıcı varsayımlar mevcuttur. Aşağıda bu varsayımlar belirtilmiştir.

2.7.3. Kısıtlayıcı Varsayımlar

Çizelgeleme yapılırken gerçek ve/veya gerçeğe yakın çözümlerin elde edilmesi için işler ve makinelerle ilgili bir takım unsurların gözden geçirilmesi gerekmektedir. Bazı durumlarda problemin çözümü açısından bazı kısıtların önemi azaltılır veya kaldırılır.

2.7.4. İşlerle ilgili kısıtlayıcı varsayımlar

Gerçek ve/veya gerçeğe yakın çözümler elde edebilmek için işlerle ilgili bazı kısıtların önemini azaltmak veya tamamen kaldırmak gerekir. Aşağıda bazı kısıtların önemi maddeler halinde verilmiştir [4]:

1. $İ_i$ kümesi i işini tanımlayan operasyonların oluşturduğu kümedir. İ sabit ve bilinmektedir.

2. Bütün işler aynı anda işlenmek için hazır durumdadırlar ve birbirlerinden bağımsızdırlar.

3. Bir sınırsız zaman diliminde bütün işler hazır durumdadırlar.

4. Her iş şu durumda olabilir:

a) İş, bir sonraki makinede işlenmek üzere beklemektedir.

b) İş, bir makinede işlenmektedir.

c) İş, sonuncu makinede de işlendi ve tamamlandı.

5. Bütün işler eşit önem derecesine sahiptir.

6. Her iş, o iş için tahsis edilmiş bütün makinelerde işlenmelidir.

7. Bir iş aynı anda yalnızca bir makine tarafından işlenebilir.

Bu kısıtlardan, 1. unsur problemin deterministik yapıda olmasını sağlar, 2. unsur ise çok dikkat edilmesi gerekmeyebilir. Bu unsur olmadan da yapılan çizelgelemeler mevcuttur. 2. unsurun sağlanmadığı duruma örnek, işlerin birbirine bağımlı olduğu durumdur. İşlerin bağımlı olması iş önceliklerini önemli hale getirmektedir. 3. unsur ise geçerli olmamakla birlikte işlerin teslim edilmesi gereken bir tarih vardır. 4. unsur ara stoklara izin verilmeyen yerlerde geçerli değildir. 5. unsurda işler her zaman aynı önem derecesine sahip olmayabilir. 2 farklı siparişte 2. siparişe ait işler daha fazla önem derecesine sahip olabilir. 6. unsurun sağlanmaması durumunda atölye ortamına ait çizelgelemede büyük değişiklikler oluşacaktır.

2.7.5. Makinelerle ilgili kısıtlayıcı varsayımlar

Gerçek ve/veya gerçeğe yakın çözümler elde edebilmek için makinelerle ilgili bazı kısıtların önemini azaltmak veya tamamen kaldırmak gerekir. Aşağıda bazı kısıtların önemi maddeler halinde verilmiştir [4]:

1. M_i kümesi i işini tanımlayan makinelerin oluşturduğu kümedir.

2. M sabit ve bilinmektedir.

3. Bütün makineler aynı anda işlem yapmak için hazır durumdadırlar ve birbirlerinden bağımsızdırlar.

4. Bir sınırsız zaman diliminde makineler hazır durumdadırlar.

5. Her makine şu durumda olabilir:

a) Makine, bir sonraki iş için beklemektedir.

b) Makine, bir işi yapmaktadır.

c) Makine, sonuncu işi de tamamlamıştır.

6. Bütün makineler eşit öncelik derecesine sahiptir.

7. Her makine, kendisine tahsis edilmiş bütün işleri tamamlar.

8. Bir makine aynı anda yalnızca bir iş yapabilir.

1. unsur problemin deterministik yapıda olmasını sağlar. 2. ve 3. unsur gerçeğe tam anlamıyla uymaz. Çünkü gerçek ortamda makineler arıza, bakım ve vardiya gibi durumlardan dolayı hazır halde olmamaktadır. 4. unsurda makine, 2. ve 3. unsurlarda

olduğu gibi hazır halde olmayabilir. Yalnız 4. unsur problemden çıkarılamaz. 5. unsurda tüm makineler aynı önem derecesine sahip olmayabilir. 7. unsurda ise teknolojinin gelişmesiyle bir makine birden fazla işi aynı anda yapabilmektedir.

2.7.6. İşler ve makinelerle ilgili kısıtlayıcı varsayımlar

Gerçek ve/veya gerçeğe yakın çözümler elde edebilmek için hem işler hem de makinelerle ilgili bazı kısıtların önemini azaltmak veya tamamen kaldırmak gerekir. Aşağıda bazı kısıtların önemi maddeler halinde verilmiştir [4]:

1. Bütün işlerin işlem süreleri sabit ve sıralamadan bağımsızdır

2. Başlayan her operasyon bitene kadar kesinti olmaz.

3. Bir işteki operasyonların sırası sabit ve bilinmektedir.

4. Bir makinenin işlem sırası bilinmekte, ancak bu işlem sırası sabitlenmek istenmektedir [13, 19].

1. unsurda insan ve makinelerin eskime faktöründen dolayı işlem süreleri kesin bir sabitliği yoktur. Ancak gelişen teknoloji ve kullanılan mikro denetleyici gibi elemanların üretim sürecine katılmasıyla bu unsur sağlanmaktadır. 2. unsur çok önemlidir, çünkü başlayan bir işin bitene kadar işlem görmesi gerekmektedir. Ancak bazı çizelgelemelerde işin yarıda kesilip sonra devam ettirilmesi sorunun çözümünde yardımcı olmaktadır. 3. ve 4. unsurlar çok önemlidir ve problem çözümünde göz ardı edilemez.

2.8. Atölye Çizelgeleme Değerlendirmesi

Bir çizelgeleme probleminin gösterimi n / m / A / B şeklindedir. n iş sayısını, m makine sayısını, A makinedeki akış türünü, B başarım ölçütünü gösterir.
Makinelerdeki akış 3 farklı türde olabilir [20].

1. Seri iş akışlı (F)

2. Permütasyon seri iş akışlı (P)

3. Karmaşık iş akışlı (G)

Çizelge değerlendirme de;

$$C_i = \sum_{k=1}^{m} (W_{ik} + P_{ij(k)}) \qquad i = 1, 2, 3, \ldots, n \tag{2.1}$$

Denklem 2.1' de:

C_i: Tamamlanma zaman. Bir işin en son işleminin tamamlandığı zamandır.

P_{ij}: i işinin j makinesindeki işlem süresidir.

W_{ij}: i işinin j. operasyonu için bekleme zamanını belirtir.

$$F_i = C_i - r_i = W_i + P_i \tag{2.2}$$

Denklem 2.2' de:

F_i: İşlem görmek için hazır olduğu andan tamamlanmasına kadar geçen zamandır.

r_i: i işinin işlem görmesi için gerekli olan hazırlık zamanıdır.

$$L_i = C_i - d_i \tag{2.3}$$

Denklem 2.3’ te:

L_i: Gecikme zamanı. İşin tamamlanma zamanı ile teslim edildiği zaman arasındaki farktır.

d_i: i işinin teslim edilmek zorunda olduğu zamandır.

2.8.1. Ortalama akış süresinin enazlanması

Permütasyon tipi bir çizelgelemede en kısa işlem süresine göre (EKİS; Shortest Processing Time SPT) işlerin makinelere yüklenmesiyle ortalama akış süresi kısaltılabilir. Bulunan tüm işler artan süre durumuna göre sıralanırlar. En kısa işlem süresine sahip işten başlanır ve çizelgelenmemiş iş ilk boşalacak olan makineye yüklenir [4].

2.8.2. Toplam çizelge süresinin ve ortalama akış süresinin azaltılması

n adet iş ve m adet makinenin bulunduğu çizelgelemede C_{max} (toplam çizelge süresi) ve ortalama akış süresi azaltılması için en uzun işlem süresi (EUİS; Longest Processing Time LPT) yöntemi kullanılır. Bu yöntemde; sistemdeki tüm işler azalan işlem süresine göre sıralanır. En uzun işlem süresine sahip işten başlanır ve çizelgelenmemiş iş ilk boşalacak olan makineye yüklenir. Her makinedeki işlerin sırası ters çevrilerek EKİS sırasında çizelgelenir [4].

2.8.3. En fazla gecikmenin (T_{max}) azaltılması

Bu unsurun sağlanabilmesi için teslim tarihi kuralı (TTK; Due Date Rule DDR) uygulanır. Bu kuralda; tüm işler teslim tarihine göre artan sırada sıralanır. Teslim tarihi en erken olan işten başlanır ve çizelgelenmemiş iş ilk boşalacak olan makineye yüklenir [4].

2.8.4. Gecikme süresinin azaltılması

İşlerin gecikme süresinin azaltılması için boş süre (slack time) kuralı uygulanır. Bu kurala göre; tüm işler artan boş süreye göre sıralanırlar. En küçük boş işten başlanır ve çizelgelenmemiş iş ilk boşalacak olan makineye yüklenir [13].

2.9. Seri İş Akışlı Atölye Çizelgeleme Problemleri

2.9.1. İki makineli – çok işli problem

n / 2 / F / C_{max} olarak gösterilen bu tür atölye çizelgeleme de tamamlanma zamanını en aza indirecek olan en iyi iş sırasını veren algoritma S.M. Johnson [4] tarafından geliştirilmiştir.

Algoritmanın uygulanması için [4];

- Her işin birinci ve ikinci makinedeki en düşük işlem zamanı bulunur.

- En düşük zamanı birinci makinede olan işler, birinci makinedeki iş sürelerine göre küçükten büyüğe doğru sıralanır.

- En düşük zamanı ikinci makinede olan işler, ikinci makinedeki iş sürelerine göre büyükten küçüğe doğru sıralanır.

- Bu sıralar bozulmadan önce birinci makineye göre bulunan sıradaki işler, sonra da ikinci makineye göre bulunan sıradaki işler olmak üzere en iyi sıralama elde edilir.

2.9.2. Üç makineli – çok işli problem

Seri iş akışının olduğu üç makineli sistemde belirli şartların sağlanması durumunda Revize Johnson yöntemi kullanılarak en kısa işlem zamanlı çizelgeleme yapılabilir. Bu şartlar [4];

- Birinci makinedeki işlem zamanlarının en küçüğünün ikinci makinedeki işlem zamanlarının en büyüğünden büyük veya eşit olması gerekir.

- Üçüncü makinedeki işlem zamanlarının en küçüğünün ikinci makinedeki işlem zamanlarının en büyüğünden büyük veya eşit olması gerekir.

2.9.3. m makineli – n işli problem

Seri iş akışının olduğu ve 3' den fazla makinenin bulunduğu çizelgelemede en kısa işlem süresini bulma işlemi çok zordur. Brooks ve White' ın [22] geliştirdiği dal-sınır yöntemi basit çizelgelemelerde başarılı sonuç vermiştir. Bu problemlerde n! adet mümkün çözüm olması ve bunlardan birkaçının verilen başarım ölçütünü en iyilemesi çözüm açısından çok büyük zorluklar yaratır.

2.10. Permütasyon Tipi İş Akış (PTİA)

Bu tip iş akış tekniği seri imalat yapan işletmelerde ortaya çıkan problemlere çözüm üretmek için kullanılır. Bu tip üretim atölyesinde makineler bir hat boyunca dizilirler ve işler bu hattı takip ederler. Bu tür iş akış problemine tek kaynak kısıtı makinedir. PTİA' da kabul edilen unsurlar şunlardır [22];

1. Tüm işlerin ve tüm makinelerin t = 0 anında hazır oldukları kabul edilir.

2. Bir iş bir makineye atandığında, bitirilmeden aynı makinede başka bir iş yapılamaz.

3. Bir makinede operasyonun başlatılması için işin bir önceki makinede tamamlanmış olması gerekir.

4. Makinelerde bozulma ve aksama yoktur.

5. Makinelerin hazırlanma süreleri işlem süresinin içindedir.

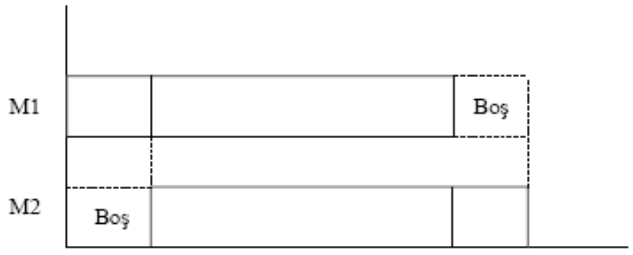
6. İşlerin makineler arasındaki taşıma süresi ihmal edilir.

2.11. Atölye Çizelgeleme Problemi Çözüm Metotları

Atölye çizelgeleme problemlerinin çözüm algoritmaları incelendiğinde üç ve daha az sayıdaki makinenin olduğu durumlarda optimizasyonu gerçekleştirmek için çeşitli algoritmalar geliştirilmiştir. Dört makine ve üstündeki atölye ortamlarının çözümü için ise sezgisel yöntemler kullanılmıştır [23].

2.11.1. Johnson algoritması

Tamamlanma zamanına dayalı performans kriterlerine göre makine sayısına bağlı olarak gerçek ve/veya gerçeğe yakın çözümü veren algoritmadır. Hazırlık zamanlarının sıfır olduğu kabul edilir. $C_{max}= F_{max}.M_2$ makinesinde işleme başlamak için M_1 makinesinde en küçük işlem zamanlı işlerin çizelgelenmesi gerekir.



Şekil 2.3: İki makineli bir gant diyagramı [22]

Şekil 2.3' e bakıldığında M_1 makinesinde kısa işlem süresine sahip olan işler ön sıralara M_2 makinesinde kısa işlem süresine sahip işler son sıralara bırakılmaktadır [25].

2.11.2. CDS algoritması

Champell ve arkadaşlarının geliştirdiği bu algoritma yapısında 3 makineli n işli sistem (m-1) adet yani 2 makine haline getirilerek Johnson algoritması uygulanır. İşlem süreleri dikkate alınarak iki farklı çözüm algoritması uygulanır. 1. ve 2. makinelere ait işlem süreleri toplanır, 2. ve 3 makinelere ait işlem süreleri toplanır ve oluşan sistem 2 makineli sistem gibi Johnson algoritmasına göre çözülür [25].

2.11.3. Dal-Sınır algoritması

Ignall ve Schrage [26] tarafından geliştirilen dal-sınır algoritması, problemin bir karar ağacı şeklinde tanımlanmasını temel alır. Problem bu şekle dönüştürülünce her seçenek, en iyi çözümü vaat eden seçenekler incelenerek değerlendirilir. Bunun için her düğüm noktası bir alt-sıralama olarak düşünülür ve bu düğüm noktasında kendiliğinden oluşan diğer düğümler için bir alt-sınır süresi hesaplanır.

Her düğümde 1' den n' e kadar işler yer alır. Ağacın yapısındaki ilk düğüm başlangıç durumu olup hiç bir iş, sıralama konusu yapılmaz. Bu düğümden çıkan dalların sayısı, iş sayısı (n) kadardır ve her işin ilk sıraya konması olanaklıdır. Bu düğümlerin her birinden de (n–1) tane iş, ikinci sıraya konabilir. Dolayısıyla (n!) tane sıralama yapılabilecek bir problemde düğüm sayısı [1+n+n*(n–1)+…+n!] tane olacaktır [22].

2.11.4. Tabu araştırmaları

Yerel iyilemenin tıkanıklığından kaçınmak için geliştirilmiş, en iyiye yakın çözüm veren bir sezgisel arama metodudur. Tabu araştırmaları, hedefe kısıt ve tabu kullanarak belirli bir olasılıkla ulaşır. Hedefe ulaşma olasılığını arttırmak için kısıt ve tabu sayılarının artması gerekmektedir.

Tabu listesi, istenilen değişimleri saklamak ve istenmeyen durumların ortadan kaldırılması için kullanılır. Tabu araştırmaları üç strateji içerir [27, 28].

1. Yasaklama stratejisi; hangi hareketlerin, tabu listesine girip girmeyeceğini belirleyen ve döngüyü önlemek için, belirli hareketleri yasaklama mekanizmasını oluşturan bir stratejidir.

2. Yeterlilik kriteri, yeterli koşulları sağlayan çözümlerin tabu durumundan çıkarılmasını sağlar.

3. Kısa dönem stratejisi, önceki döngülerdeki davranışları depolayan bir stratejidir.

2.11.5. Tavlama benzetimi

Tavlama benzetimi, NP zor problemlerinin çözümünde iyi performans gösteren sezgisel bir yöntemdir. Fiziksel tavlama işlemi, ısı banyosu içerisindeki katı bir cismin düşük enerjilerini elde etmek için kullanılan bir yöntemdir. Katı bir cisim erime noktasına kadar ısıtılır ve sonra katı cisim hızla soğutulmaya başlanırsa katı cismin moleküler yapısı soğutma oranına bağlı olarak değişir.

Tavlama algoritması, yerel arama metotlarına benzer. Yerel arama metodunda sistem çözümünü genel eniyileme yerine yerel eniyilemeyi bulmaktadır ve bu olumsuz bir olgudur. Tavlama benzetiminde bu olumsuzluk ortadan kalkmaktadır. Örnek bir algoritma yapısı aşağıda adım adım belirtilmiştir [29].

2.11.6. Yapay bağışıklık sistemleri

Bağışıklık sistemleri ile bir problem çözülürken, işlem içsel ve dışsal mesajlardan oluşur. İçsel mesajlar; hücre ve moleküller, dışsal mesajlar; bakteri, parazit ve virüs gibi yabancı maddelerdir. Problem çözümlerinde içsel ve dışsal mesajların ayırt edilmesi birçok açıdan zordur. Yapay bağışıklık sistemleri bir çok endüstriyel problemlerin çözümünde kullanılabilen yeni bir yapay zeka algoritmasıdır [17].

2.11.7. Genetik algoritma

Genetik Algoritma (GA), rastsal arama tekniklerini kullanarak çözüm bulmaya çalışan, parametre kodlama esasına dayanan bir arama tekniğidir ve bir veri grubu içinde özel bir veriyi bulmak için kullanılır [15].

GA' lar doğada geçerli olan en iyinin yaşaması kuralına dayanarak sürekli iyileşen çözümler üretir. Bunun için "iyi" nin ne olduğunu belirleyen bir uygunluk fonksiyonu ve yeni çözümler üretmek için yeniden kopyalama, değiştirme gibi operatörleri kullanır. GA' ların bir diğer önemli özelliği de bir grup çözümle uğraşmasıdır. Bu sayede çok sayıda çözümün içinden iyileri seçilip kötüleri elenebilir.

2.11.8. NEH (Nawaz, Enscore, Ham) algoritması

m makineli – n işli problemlerin çözümü için NEH algoritması üç temel adımdan oluşur [31]:

- Tüm işlerin toplam işlenme süreleri hesaplanır:

$$P_j = \sum_{i=1}^{m} P_{ij} \tag{2.4}$$

- İşler artan sıraya göre sıralanır.

- j=1, …, n olmak üzere j işi alınır ve sıralanmış işler arasındaki tüm olası pozisyonlara yerleştirilerek en iyi sıralama bulunur.

Örnek olarak; Sıralanan n iş içerisinden toplam işlem zamanları en yüksek olan iki iş seçilir. Seçilen iki iş sıraya konarak tamamlanma süreleri bulunur. Kısmi olarak bulunan bu sürelerden en küçük olan iş sırası seçilerek bu iki işin birbirlerine olan önceliği belirlenmiş olur. Bu sıra sabitlenerek daha sonraki kullanımlarda da ele alınır. Daha sonra toplam işlem zamanı en büyük olan üçüncü iş seçilir. Bu iş; başa, ortaya veya sona konarak tekrardan kısmi tamamlama süresi bulunur ve bu süre bir

sonraki adımlarda kullanılmak üzere sabitlenir. Her defasında yeni eklenen iş için aynı işlemler uygulanarak bütün işlerin sıralamasının belirlenmesine kadar devam eder [17].

2.11.9. FRB3 algoritması

Rad, S.F., Ruiz, R., Boroojerdian, N. tarafından bugüne kadar en iyi çözümleri veren algoritmalar arasında görülen NEH algoritmasının eksikliklerini gidermek ve daha iyi çözümler elde etmek amacıyla geliştirilmiştir.

NEH yönteminin eksikliklerinden bir tanesi; algoritmanın işlem süresince yeni yerleştirilecek iş için yer aranırken daha önceden yerleştirilmiş olan işlerin sırası sabit kalır. Fakat çoğunlukla yeni işi önceden yerleştirilmiş olan işlerin yerine yerleştirmek daha karlı olmaktadır. NEH tüm işler için tüm pozisyonları denemeden, ilk bulduğu en iyi çözümde işlerin yerini sabitlediğinden dolayı daha iyi çözümleri atlamaktadır.

FRB3 algoritması geliştirilmeden önce bir biri ile aynı temel mantığa dayanan FRB1 ve FRB2 algoritmaları geliştirilmiştir. Zamanla görülen eksiklikleri giderebilmek için ortaya çıkarılan FRB3 algoritmasının yapısı aşağıda adım adım belirtilmiştir[31].

Adım1. Tüm işlerin toplam işlenme süreleri denklem 2.4' e göre hesaplanır.

Adım2. İşler artan sıraya göre sıralanır. (Ps dizisi)

Adım3. j=1, …, n olmak üzere j işi alınır ve sıralanmış işler arasındaki tüm olası pozisyonlara yerleştirilerek toplam işlem süresinin en az olduğu pozisyon (possiblePosition) belirlenir ve j işi o pozisyona yerleştirilir (pi dizisi).

Adım4. pi dizisi içerisinde h=1, …, possiblePosition olmak üzere h işi alınır ve sıralanmış işler arasındaki tüm olası pozisyonlara yerleştirilerek toplam işlem süresinin en az olduğu pozisyon (possiblePosition) belirlenir ve h işi o pozisyona yerleştirilir.

Adım5. Durma kriteri: Tüm işler yerleştirilmişse, dur; aksi halde Adım3' e geri dön.

Şekil 2.4' te FRB3 algoritmasının kaba kodu verilmektedir:

```
for (int step = 0; step < Ps.Count; step++)
    {
       Job j = (Job)Ps[step];
       int possiblePosition = TestAllPossiblePositions(pi, j, problem.Matrix);
       InsertJobAtPossiblePosition(pi, possiblePosition);

       for (int step2 = 0; step2 < step; step2++)
       {
          Job h = (Job)pi[step2];        //  h işini step2 den al
          pi.RemoveAt(step2);            // h işinin step 2 deki yerini değiştir
                                         // Taillard İvmelendirmesi
          possiblePosition = TestAllPossiblePositions(pi, h, problem.Matrix);
          InsertJobAtPossiblePosition(pi, possiblePosition);
       }
    }
```

Şekil 2.4: FRB3 algoritmasının kaba kodu [31]

3. KARINCA KOLONİLERİ ALGORİTMASI

3.1. Temel Kavramlar

3.1.1. Rastgelelilik

Doğadaki adaptasyon, “rastgele” olarak ifade edilebilen durumları içeren bir süreçtir. Rastgelelilik doğada ihtimale dayalı bir olgudur ve aynı olguya ait tekrar eden olaylar farklı sonuçlanabilir. Nedensellik zincirlerini bilmediğimiz durumlar hariç, rastgele gözüken şeyler her zaman deterministiktir. Deterministik çözümler, zamanı tam olarak tahmin edilebilen ve nedenleri bilinen bir sistemi ifade eder.

Adaptif bilgisayar programlarında, rastgeleliliğin iki önemli işlevi vardır. Bu işlevlerden ilki, basit olarak belirsizliğin veya kararsızlığın ifadesidir. Bir yere gidilme zorunluluğu olan fakat nereden başlanacağı bilinmeyen durumlarda rastgele bir yön seçilir. Rastgele seçimlerin ikinci önemli işlevi ise yeni durumların denenmesini sağlamalarıdır. Kimi zaman rastgelelilik, bazı yeni ve başarılı sonuçların ortaya çıkmasına neden olmaktadır [32].

3.1.2. Kendi kendine organizasyon (Self Organization - SO)

Birçok ayrı bireyden oluşan bir topluluğun, değişen şartlardan dolayı, kendi durumunu değiştirerek yeni bir düzen kazanmasına kendi kendine organizasyon adı verilmektedir. Böceklerin kendilerine göre basit, etkileşimli yaratıklar oldukları farzedilerek karmaşık kolektif davranışı açıklamanın mümkün olduğu ifade edilmektedir [32].

Kendi kendine organize olmayı esas alan modeller, mantıksal olarak basit etkileşim işlemleri olarak görünmektedir. Daha karmaşık kabuller modelde ele

alındığında, bu açıklamanın yetersizliği anlaşılır. Bal arısının yuvaya geri dönmesi basit bir davranış olarak ele alınabilir, fakat nörobiyolojik seviyedeki tanımlaması tabii ki basit değildir.

Kendi kendine organize olabilen böcekler hakkındaki bilgimizi zeki sistem tasarımına taşıdığımızda, bize çok güçlü etki ve araçlar sağlar. Esneklik değişen çevreye uyumu sağlar.

3.1.3. Etkileşim (Stigmergy)

Etkileşim kelimesi ilk olarak, bir böcek türü olan termitlerin yuvalarını yeniden inşa etme sürecindeki düzenlerini ve görev koordinasyonlarını açıklamak için kullanılmıştır [33].

Etkileşim, genel bir mekanizmayı açıklar. Bu mekanizma bireysel davranışın çevreyi geliştirmesi, sonuçta diğer bireylerin davranışının ve koloni seviyesindeki toplam davranışların gelişmesidir. Çevre bir tür haberleşme ortamı olarak iş görür. Çevre, dıştan bir etki sonucu değişirse, bu değişim sanki koloni aktiviteleri sonucu oluşmuş gibi değerlendirilir. Koloni, bireylerin aynı davranışı göstermesiyle kolektif bir cevap verebilir.

Sosyal böceklerdeki kendi kendine organizasyon, böceklerin kendi aralarında etkileşimlerini gerektirir. Dolaylı bir etkileşim, bir böceğin çevre şartlarını değişikliğe uğratması, diğerlerinin de bu yeni çevre şartlarına sonraki bir zamanda cevap vermesidir. Bu, bir etkileşim örneğidir.

Kendi kendine organizasyon teorileri, kimi zaman tek bir ajanla veya robotla elde edilebileceklerden daha farklı şekillerde sonuçlanan kolektif davranışları da açıklar.

Bireysel davranıştaki rastgelelilik veya dalgalılıklar, gerçekte sistemin yeni davranışlar keşfetme ve yeni çözümler bulma kabiliyetini büyük oranda arttırabilmektedir. Büyük sayıdaki ajanları ele alan merkezi bir kontrol genellikle iyi

bir çözüm değildir. Bu bir telekomünikasyon ağı için de doğrudur. Çünkü sadece robottan, kontrolöre veya bunun tersine haberleşme gerekliliği değil, aynı zamanda kontrolörün bir bozukluğunun, bütün bir sistemin çökmesi sonucuna varmaması istenmektedir [34].

Heterojen sürülerin, yani farklı görevleri yapan ve farklı uyarılara cevap veren veya aynı uyarana farklı bir cevap veren sürülerin tasarımı için, gerçek olarak teorik bir kılavuz yoktur.

3.1.4. Optimizasyon

Optimizasyon, genellikle bir sistemi, ihtimal dahilindeki en iyi sonucu alabilme yönünde ayarlama işlemini ifade eder. Optimizasyon, bir sistemi matematiksel yöntemler kullanarak mümkün olan en iyi duruma getirme ve fonksiyonel kılmadaki yöntemlerdir. Genel olarak "problem" kavramını, bazı gerçeklerin ve bilgilerin arandığı bir durum olarak tanımladığımızda, optimizasyon işlemini sistemi uyarlayarak eksik bilgileri araştırma işlemi olarak tanımlayabiliriz.

Optimalite fikri, biyolojide güvenilmez bir kavramdır ve bazı kısıtlamalarla tanımlanmıştır. Bunlar, diğer kolonilerle yarış veya yağma gibi çevreyle ilgili kısıtlamalardır. Bazı durumlarda koloninin bir şey arama aktivitelerini, daha iyi bir alan olsa da verilen bir alanda yapması avantajlıdır, çünkü tek bir izin takibi kolaydır ve diğer alanlara fırsatçı bir yolla gitme, koloninin korunma seviyesinde bir azalma veya koloninin tehlikeye düşmesi gibi bazı maliyetler oluşturabilir [35].

Bir optimizasyon problemi, problemin hatalarını minimize etme veya iyi yönlerini maksimize etme şeklinde verilebilir. Nitel değişkenli, yani ayrık durumlu veya değerli optimizasyon problemleri, nicel yani miktara bağlı ve sayısal olan optimizasyon problemlerinden yapısal olarak farklıdır. Nitel değişkenli bir problemdeki yaklaşım, bir sonucu minimize veya maksimize etmede elemanları düzenleme problemidir. Bazı durumlarda bazı elemanların bile elenmesi gerekebilir,

böylece yeniden düzenlenecek şeylerin sayısı kendi kendine problemin bir parçası olabilir [32].

Kombinasyonel optimizasyon, özellikle geleneksel yapay zeka alanıyla ilgilidir. Özellikle ve çoğunlukla da bazı önerileri mantığın bazı kurallarına uyacak şekilde düzenleme işleminin olduğu alanla ilgilidir. Önerilerin sırasını düzenleme, kombinasyon hesapları içeren bir optimizasyon problemidir. Kombinasyonel optimizasyon problemlerinin ifade edilmeleri kolay fakat çözümleri çok zordur.

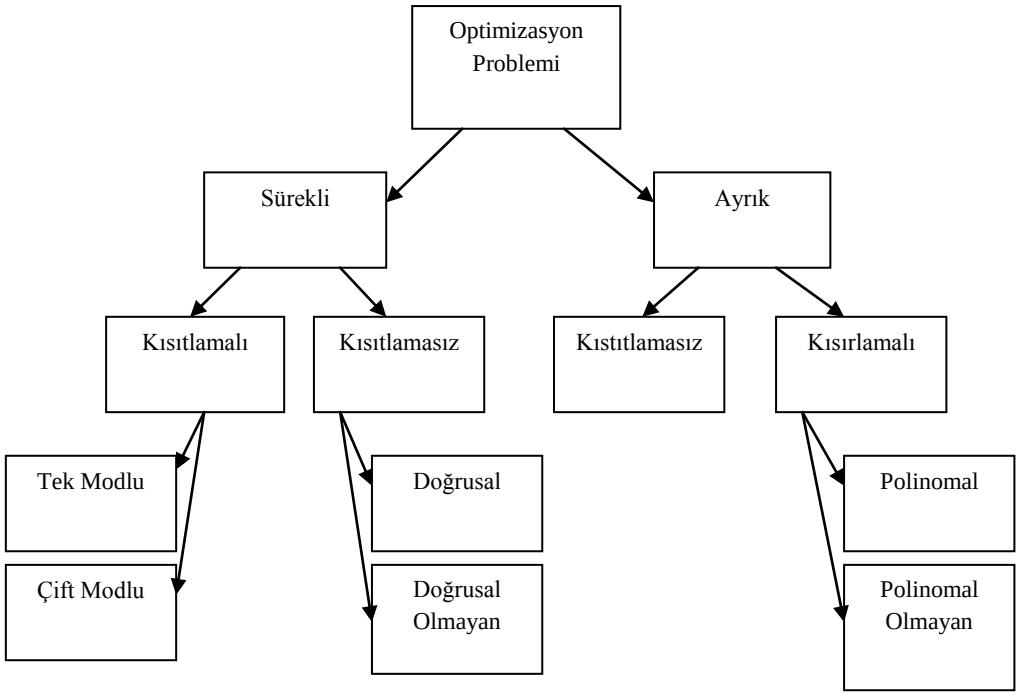
Uygulamalarda karşımıza çıkan problemlerin birçoğu karmaşık yapıdadır. Bu yapı, polinomial olarak sınırlanmış hesaplama süresi içinde problemin optimal olarak çözülemeyeceği durumunu yansıtan bir yapıdır. Başka bir deyişle, örneğin GSP' deki düğüm sayısının her artışında, problem çözümündeki hesaplama zorluğu logaritmik olarak artar. Bu nedenle, bağıl olarak kısa bir zamanda optimal çözüme yakınlaştırıcı yöntemler kullanılmasını gerektirir. Bu türdeki algoritmalar sezgisel olarak adlandırılır. Sezgiseller, incelenmesi gereken alanın büyüklüğünü azaltan, araştırma stratejisindeki kestirme yollardır. Sözlük anlamı, araştırmaya kılavuzluk etmede genel bir formülasyon kullanma durumudur [36].

Uygulamada, bağıl olarak kısa bir zamanda optimale yakın çözümler veren yakınlaştırma yöntemleri içeren türdeki algoritmaların geniş anlamdaki adıdır [37].

3.1.5. Optimizasyon problemlerinin sınıflandırılması

Optimizasyon problemleri kullanılan değişken tipine, kısıtlama varlığına ve hesaplama karmaşıklığına göre Şekil 3.1' deki gibi sınıflandırılabilir [38].

Optimizasyon problemleri değişken tipine göre sürekli ve ayrık problemler olmak üzere ikiye ayrılır. Problem değişkenleri üzerinde sınırlama varlığına göre sınırlamalı ve sınırlamasız olarak bölünür. İşlem karmaşıklığına (complexity) göre tek modlu, çok modlu, polinomial (P) ve belirleyici olmayan polinomial (NP) problemler olarak gruplandırılırlar.



Şekil 3.1: Optimizasyon problemleri [32]

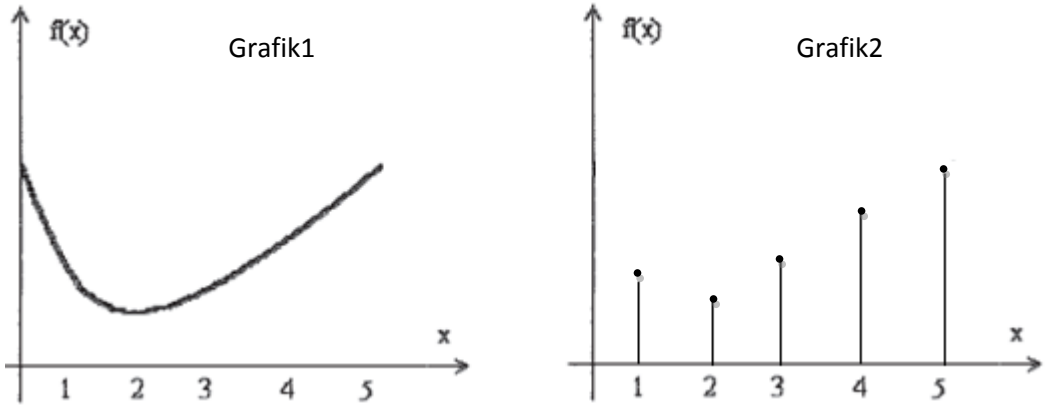
3.1.5.1. Sürekli ve ayrık problemler

Bir optimizasyon problemindeki bilinmeyenler uygulanabilir bölgede tüm değerleri alabiliyorsa, bir başka deyişle problem sonsuz sayıda giriş ve çıkış değerleri alabiliyorsa bu optimizasyon problemi sürekli problem olarak adlandırılır. Eğer değişkenler sonlu sayıda belirli değerleri alabiliyorsa optimizasyon problemi ayrık problem veya kombinasyonel problem olarak adlandırılır. Diğer bir deyişle ayrık problem, nesnelerin bir amacı gerçekleştirecek biçimde gruplandırılması, seçilmesi veya sıraya konması problemi olarak tanımlanır.

Bir sürekli problemin bilinmeyenleri gerçek değerler kümesinde tanımlı tüm değerleri alabilir veya sınırlamalar sürekli değerler üzerinde geçerlidir. Ayrık problemlerde bilinmeyenler birer tamsayı olarak veya grafik üzerinde birer koordinat değeri olarak sadece ayrık değerler alabilir. Sürekli ve ayrık problem kavramı Şekil 3.2’ de verilen iki amaç fonksiyonu ile örneklenebilir.

Şekil 3.2' de grafik1’ de dikkate alınan amaç fonksiyonun parametresi x, $0<x<5$ aralığında istenilen ondalık hassasiyette sonsuz sayıda değer alabilir. Grafik2' de

ise 0 < x < 5 aralığında x parametresinin alabileceği değerler {1, 2, 3, 4, 5} kümesi ile sınırlıdır ve bu değerler sonlu sayıdadır. Uygulanabilir bölgede tanımlı bu tamsayılar dışında problem değişkenlerinin değer alması mümkün değildir.



Şekil 3.2: Sürekli araştırma uzayı (Grafik1) ve ayrık araştırma uzayı (Grafik2) [32]

3.1.5.2. Tek modlu ve çok modlu problemler

Bazı optimizasyon problemlerinin yalnızca bir adet bölgesel minimumu vardır ve bu değer aynı zamanda problemin küresel minimum değeridir. Çözümü kolay bulunan bu tür problemlere tek modlu problemler denir.

Gerçek hayat karmaşıktır ve genellikle çözüm aranan problemler doğrusal olmayan yapıya sahiptir. Bu yapıdaki problemlerde genellikle çok sayıda bölgesel minimum bulunur. Çözümü zor olan bu tür problemlere de çok modlu problemler denir [39].

3.1.5.3. Doğrusal (linear) ve doğrusal olmayan (nonlinear) problemler

Grafiği doğrulardan oluşan başka bir deyişle; y=Ax+B şeklinde birinci dereceden denklem eşitliği veya eşitsizliği ile tanımlanan fonksiyonlara doğrusal fonksiyon denir. Grafiği eğrilerden oluşan ve "y=e(x), log(x), x*x, x" şeklinde tanımlanan fonksiyonlara da doğrusal olmayan fonksiyon denir [25].

Eğrisellik veya doğrusallık durumu sürekli ve sınırlamalı optimizasyon problemleri için söz konusudur. Sınırlamalı sürekli problemler, sınırlama fonksiyonlarına bağlı olarak eğrisel veya doğrusal olurlar. Optimizasyon probleminde kısıtlamaları oluşturan fonksiyonlar doğrusal ise problem sınırlamalı doğrusal optimizasyon problemi (constraint linear optimization problem) olarak adlandırılır. Sınırlama fonksiyonlarından en az birisi eğrisel olan problem sınırlamalı doğrusal olmayan optimizasyon problemi (constraint nonlinear optimization problem) olarak adlandırılır yada bu tür problemler doğrusal olmayan programlama (nonlinear programming) olarak bilinir.

Doğrusal optimizasyon problemleri arasında hem sınırlama fonksiyonları hem de amaç fonksiyonu doğrusal olan problemlere doğrusal programlama problemleri (Linear Programming) denir. Sınırlamaları doğrusal olup amaç fonksiyonu karesel bir başka deyişle ikinci dereceden olan problemler karesel programlama problemleri (Quadratic Programming) olarak bilinir.

Doğrusal optimizasyon problemleri arasında hem doğrusal programlama hem de karesel programlama problemleri oldukça yaygındır. Doğrusal programlama problemleri bir adet bölgesel minimuma sahiptir ve bu aynı zamanda problemin küresel minimum değeridir [41].

3.2. Karınca Koloni Optimizasyonu (ACO- Ant Colony Optimization)

3.2.1. Tanımı ve tarihsel gelişimi

KKO, bir algoritmalar sınıfıdır. Geniş bir perspektiften bakıldığında, KKO algoritmaları model tabanlı araştırma algoritmaları sınıfındandır. Bu sınıf ayrık optimizasyon problemlerini çözmede gittikçe artan popülariteye sahip algoritma sınıfıdır. Model tabanlı araştırma algoritmaları, parametreleri bulunan ihtimale dayalı model kullanımıyla diğerlerinden ayrılır. KKO teknikleri, kombinasyonel optimizasyon problemlerini çözmek için karıncaların sosyal davranışını modeller. Karınca algoritmaları, gerçek karıncaların davranışının gözlenmesinden türetilen

modelleri inceleyen bir bilim dalıdır. Bütün KKO algoritmaları aynı zamanda birer karınca- algoritmasıdır [37].

Bazı problemleri çözmenin tek yolu cevapla ilgili bütün ihtimallerin tek tek test edilmesi ile gerçekleşir. Buna verilebilecek klasik bir örnek, şehir şehir dolasan bir iş adamının izlemesi gereken en kısa rotanın bulunmasıdır. Bu tür problemler arkalarında iz bırakarak ilerleyen karıncaların uyguladığı yöntemle çözülebilir. 1989 ve 1990 yıllarında, ABD' nin Mexico eyaletindeki Santa Fe Enstitü' sünde Bonabeau ve arkadaşları "sanal karıncalar" oluşturarak benzer problemlerin bilgisayarlarla daha kolay çözülebileceğini gösterdiler [41].

Buna göre sanal karıncalar arkalarında buldukları rotanın uzunluğunu da simgeleyen bir nevi koku izi bırakacak ve diğer sanal karıncalar da kestirme rotaları bu sayede bularak tercih edeceklerdir. Koku izinin, kokusunu veren maddenin belirli bir hızda buharlaşması da simüle edilerek tercih edilmeyen uzun rotalardaki koku izleri de yavaş yavaş yok olacak ve bu da sanal karıncaların kestirme yol dışındaki uzun rotalara sapmasını önleyecektir.

Aslında, daha 1946 yılında, Grasse, böceklerin " anlamlı uyaran " olarak adlandırılan uyaranlara karşı genetik olarak kodlanmış bir reaksiyonu başlatan cevaplar verebildiklerini gözlemiştir. Sosyal böceklerde ki bunlara termitler ve karıncalar en iyi örnektirler, bu reaksiyonların etkileri, hem bu reaksiyonları oluşturan böcek için hem de kolonideki diğerleri için, yeni bir uyarıcı olarak işlev görür. Örneğin Grasse, Cubitermesler gibi Bellicositermes Natalensis cinsi termitlerin de, yeni bir yuva inşa ederken, toprağın rastgele bir şekilde koordinatları bulunmayan bir küre şeklinde bir araya getirildiğini gözlemlemiştir. Fakat küre şeklinde toplanmış toprak, sınırlanmış bir alanda bir kez belirli bir yoğunluğa eriştiğinde, bu durum yeni bir anlamlı uyaran haline gelir ve daha fazla termitin küre halinde toprak eklemesine sebep olur.

Sonuçta, sütun ve kemer ve en nihayetinde de bütün yuva inşa edilmiş olur. Araştırmacılar bu yeni problem çözme metodunu, " Karınca Koloni Optimizasyonu Algoritması" olarak adlandırmaktadırlar. Bu algoritma bilgisayar problemlerinin

çözülmesinde kullanılan sürü zekası yaklaşımına yeni bir örnektir. Kombinasyonel problemlerin birçoğu statiktir ve karakteristikleri zamanla değişmemektedir. Karınca tabanlı algoritmaların zaman–değişimli bir özelliğe sahip olan problemler sınıfına da uygulamaları yapılmıştır. Örnek olarak telekomünikasyon ağlarındaki yönlendirme verilebilir. Burada, çözümlerin anlık olarak değişmekte olan şartlara adapte edilmesi gerekmektedir. Dolayısıyla bu sınıfta kendini uyarlayabilen yapıdaki karınca tabanlı algoritmalar daha başarılı olabilir [32].

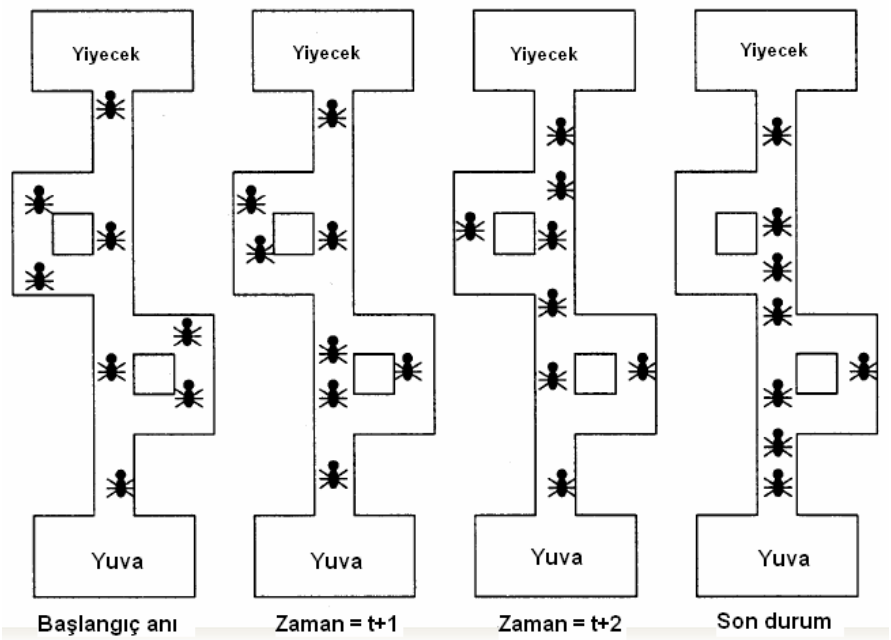
3.2.2. Feromonlar

Bilim dünyası, bu kimyasallar hakkında yeterli sayılabilecek bir bilgiye sahip değildir. Dolaylı deneylerle, farklı miktardaki ve yoğunluktaki feromonlara, değişen zaman aralıklarında karıncaların gösterdikleri davranışlar gözlenmiş ve feromonların buharlaşma oranı, emilim oranı, difüzyon sabiti gibi kaba ve yaklaşık miktarlara ve sonuçlara ulaşılmıştır. Ömürleri, karınca türlerine, enzim yapısına, koloni büyüklüğüne, hava şartlarına bağlı olarak birkaç saatten birkaç aya kadar olabilmektedir. Ortak bir akıl bu feromon buharlaşmasını kontrol altında tutar, çünkü uzun bir yolda iyi işaretlenmiş feromon izlerinin sürekliliğini sağlamak zordur.

Gerçek karınca kolonilerinin gözlenmesi sonucunda, sadece yuvaya dönerken feromon bırakan karıncaların, yuvalarıyla yiyecek kaynağı arasındaki en kısa yolu bulamadıkları anlaşılmıştır.

Bir probleme ait KKO çözümlerinin, o problemin gerçek optimal çözümleriyle karşılaştırıldıklarında ne kadar iyi olduklarını araştırmak için, arzu edilen en iyi sonuçları almamızı sağlayan kullanıcı-kontrollü parametre değerleri seçilmelidir.

3.2.3. KKO ile doğal optimizasyon



Şekil 3.3: Feromon izinin zamanın ilerlemesine bağlı olarak dinamiği [32]

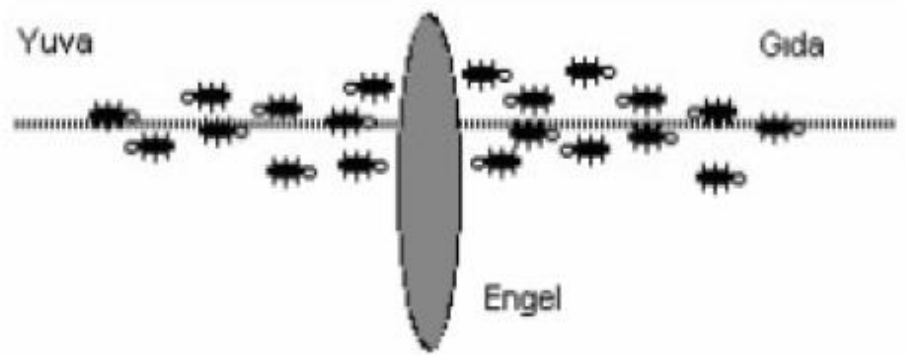
Şekil 3.3' te görüldüğü gibi, bir karınca yuvasından yiyeceğe giden farklı yollar düşünelim. Başlangıçta, zaman aralığı 0 iken, karıncalar yuvadan rastgele bir şekilde yiyecek aramak üzere ayrılırlar. Bu karıncaları yiyecek kaynağına doğru yönlendiren şey sadece kendi lokal perspektifleridir. Karıncalar yuvayı terk ettiklerinde, yollarının üzerine küçük miktarlarda feromon bırakırlar. Bu kimyasal, gelecek karıncalara bu yolun daha önceden yiyeceğe doğru gidişte kullanıldığına ilişkin bir mesajdır. Karıncalar feromonu takip etme yatkınlığına sahiptirler.

İzdeki feromon miktarı ne kadar yüksek olursa, karıncanın o izi takip etme olasılığı o kadar yüksektir. Bu kimyasal mesaj karıncaların başarısının anahtar etkenidir. Bu yapı kullanışlıdır, çünkü en iyi kararları veren karıncalar yuvaya ilk dönenler olacağından, geçtikleri yollara, henüz yiyecek aramakla meşgul diğer karıncalara göre iki kat fazla feromon bırakmış olurlar.

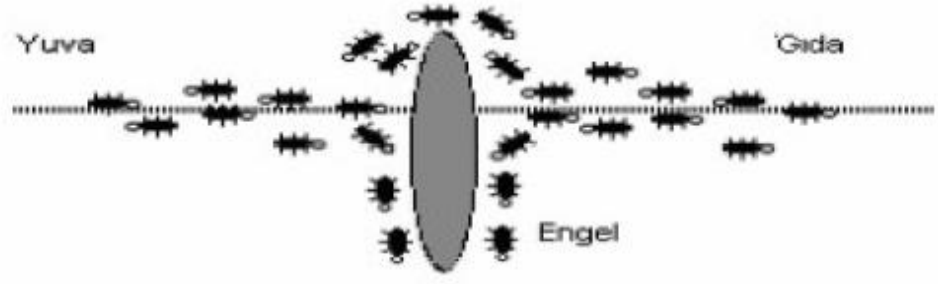
Karıncalar yol seçiminde, bir olasılık dağılımını kullandıklarından, bir karıncanın keşfedilememiş bir yolu seçme ihtimali az da olsa vardır. Bu lokal bir optimum çözüme karşı bir güven unsurudur.

Lokal olarak oluşabilecek bir çözüme kilitlenerek daha iyi çözümlerin bulunamaması ihtimali azaltılmalıdır. Aşağıdaki Şekil 3.4, 3.5, 3.6 ve 3.7' de, gerçek karıncaların en kısa yolu bulmaları gösterilmektedir.

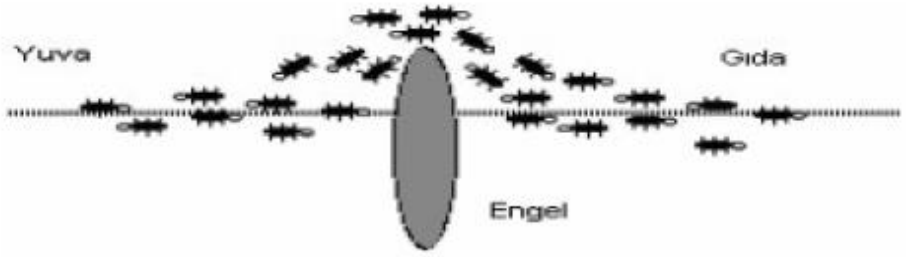
Şekil 3.4: Karıncaların izlediği yol [37]



Şekil 3.5: Karıncaların bir engelle karşılaşması [37]



Şekil 3.6: Engelle karşılaşan karıncaların seçimi [37]



Şekil 3.7: Karıncaların zamanla kısa yolu bulmaları [37]

Şekil 3.4, karıncaların yuvalarından doğrusal bir yol boyunca gıdaya gidişlerini göstermektedir. Gıdaya giden yolda herhangi bir engel meydana geldiğinde (Şekil 3.5), bu engelin hemen önündeki karınca devam edemez ve yeni gidiş yönü için bir tercih yapmak zorunda kalır. Bu konumda, karıncanın yeni yön seçeneklerinin seçilme olasılıkları eşittir. Eğer karınca sağ ve sol yönlerinden bir tanesini seçebiliyorsa, bu yönlerin seçilme şansları eşittir. Karınca yaptığı seçime göre yoluna devam eder ve kendi yolunu çizer (Şekil3.6).

Kolonide engel karşısında yol için eşit olasılıklı seçim yapan karıncalar seçtikleri yolun gıdaya giden en kısa yol olmaması durumunda, güzergahlarını çok hızlı bir şekilde yeniden yapılandırırlar. Sonuçta koloni güzergahı yeniden optimize edilmektedir. Yapılan seçimler de, bu yol üzerindeki feromon miktarını artıracak ve sonradan gelen karıncalar için tercih sebebi olacaktır.

Karıncaların gerçekte en kısa yolu bulmak için kullandıkları araç olan feromonlar, bazı hayvanların kendi cinslerinden olan diğer hayvanları etkilemek için kullandıkları bir tür kimyasal salgılardır. Karıncalar ilerlerken, depo ettikleri belirli miktardaki feromonlarını geçtikleri yollara bırakırlar ve olasılığa dayanan bir yöntemle feromonun daha çok olduğu yolu az olduğu yola tercih ederler. Böylece bırakılan feromonların etkisiyle, kendilerinden sonraki karıncalara yol seçiminde yardımcı olurlar. Bu içgüdüsel davranış, onların gıdaya giden en kısa yolu, önceden var olan bir yolun kullanılamaz olması durumunda dahi nasıl bulduklarını açıklar.

Sonradan gelen karıncaların, yeni en kısa yolu seçmelerinde feromonun pozitif etkisinin oluşabilmesi için, karınca ile yol üzerindeki engel arasındaki etkileşim çok hızlı bir şekilde gerçekleşmelidir. Her karıncanın, ortalama aynı hızda ve aynı miktarda feromon bıraktığı göz önüne alınırsa, karıncanın engeli fark edip en kısa yolu seçmesi, normal süreçten biraz daha uzun sürebilir. Fakat sonradan gelen karıncaların feromona dayalı yol seçimi toplamda gıdaya giden süreci kısaltır (Şekil 3.7).

Şekil 3.4' te, yuvadan gıdaya kısa ve uzun olmak üzere iki yol bulunmaktadır. Yiyeceğe kısa yoldan gidip dönenler ilk dönecek olanlar olacağından, uzun yoldan gidip dönenlere göre daha çok sayıda karınca gidip-dönmüş olur, dolayısıyla da en kısa yola daha fazla feromon bırakılmış olur. Sonuçta zaman geçtikçe kısa yol daha çok tercih edilir.

Karınca optimizasyon teknikleri, sanal karıncalara gerçek karıncalardan farklı bazı özellikler kazandırmaktadır. Bu özellikler yardımıyla, sanal karıncalar kendilerini yiyecek kaynaklarına yönlendirecek " daha iyi " yollar hakkında karar verebilirler. Çünkü bu karıncalar sadece lokal bilginin avantajını değil, aynı zamanda araştırmanın içerisine örneklenebilecek bütün global bilginin ve verilen kararların geçmişine erişebilme avantajını da kullanabilirler. Örneğin, feromon buharlaşması gibi gelişimler zincirleme olarak gelen karınca kararlarının olasılık dağılımını değiştirebilir. Buharlaşma ile yakın zamanda üzerinden geçilmemiş yollardaki feromon izleri azalmaya başlar. Düşük seviyedeki feromon izleri ile birlikte, bir karıncanın bu yolu seçme ihtimali azaltılmış olur. Buharlaşmayla önceki karınca neslinin bıraktığı feromon yavaş yavaş yok olur, böylece yeni karıncalar lokal optimaliteye düşmeyecekler ve değişen ortam şartlarına adapte olabileceklerdir.

3.2.4. Yapay karıncalar

Gerçek karıncalar, kör olmalarına rağmen yuvalarından yiyeceğe giden en kısa yolu bulabilmektedir. Karıncaların bu temel özellikleri kullanılarak ve bazı eklemeler de

yapılarak, gerçek problemlerin çözümünde kullanılabilen yapay karıncalar tanımlanmıştır [32]. Gerçek karıncalardan aynen alınan özellikler:

- Karıncalar arasında feromon aracılığı ile kurulan iletişim.

- Feromon miktarının fazla olduğu yolların öncelikle tercih edilmesi.

- Kısa yollar üzerinde feromon miktarının daha hızlı artması.

Eklenen özellikler:

- Zamanın ayrık olarak hesaplandığı bir ortamda yaşarlar ve hareketleri ayrık durumlardan ayrık durumlara geçişleri içerir.

- Tamamen kör olmayıp, problem ile ilgili detaylara erişebilirler.

- Belli bir miktar hafıza ile problemin çözümü için oluşturdukları bilgileri tutabilirler.

- Oluşturdukları çözüm kalitesinin bir fonksiyonu olarak belli bir miktarda feromon bırakırlar.

- Feromon bırakmadaki zamanlama.

- Sistemin genel performans ve etkinliğini arttırabilmek için geri iz sürebilme ve ileriye bakış gibi gerçek karıncalarda bulunmayan özellikler yapay karıncalara eklenmiştir.

Geri iz sürebilme, karıncanın takip ettiği yolu hafızasında tutmasından dolayı feromon bırakacağı yolları da bilmesi anlamındadır. İleriye bakış ise, karıncanın bulunduğu noktaya doğrudan irtibatlı olan bütün noktaların uzaklıklarını bilmesi ve böylece yol seçiminde yakınlığı-uzaklığı da hesaba katmasını ifade eder.

3.2.5. KKO algoritması

İlk karınca koloni optimizasyonu algoritması olan karınca sistemi (AS – Ant System) Marco Dorigo tarafından 1991' de incelenmeye başlanmış, doktora tezi olarak da 1992' de sunulmuştur [42].

Birkaç gezgin satıcı probleminde denenmiş ve küçük boyuttaki (75 şehirden daha az olan) gezgin satıcı problemlerinin çözümünde başarılı olmuştur. Yapı olarak da paralel çalıştırılmaya çok uygundur.

Koloni yapısında, birden fazla asenkron veya paralel çalışan karınca vardır. KKO algoritmalarında, çözümlerin oluşturulması yönünden paralel ve ardışıl olmak üzere iki farklı çalışma mantığı bulunmaktadır.

Paralel çalışmada, çözümün her adımda bütün karıncalar aynı anda bulundukları düğümden bir sonrakine hareket ederler.

Ardışıl çalışmada, bir karınca tam bir tur oluşturduktan sonra, diğer bir karınca başka bir tur oluşturmaya geçer. Böylece karıncalar sıralı olarak hareket etmiş olurlar. Karınca sisteminde turlar oluşturulurken her iki seçimde de, algoritmanın davranışında anlamlı bir fark bulunmamaktadır. Fakat Karınca Koloni Sistemi gibi diğer karınca koloni optimizasyonu algoritmalarında bu böyle değildir ve anlamlı bir farklılık bulunmaktadır. Bunun sebebi, karınca sisteminde her karıncanın her turdaki her bir adımında feromon bırakması, fakat karınca koloni sisteminde sadece en iyi karıncanın her tur sonunda bir defa feromon bırakmasıdır [43].

3.3.Karınca Optimizasyon Yöntemleri

3.3.1. Karınca sistemi yöntemi ve temel özellikleri

Karınca sistemi yöntemi, karınca optimizasyonu kavramının ilk uygulaması olduğundan kendi alanında büyük bir önem taşımaktadır. Bu kavram ve temel özellikleri aşağıda ayrıntılarıyla incelenmektedir.

3.3.2. Düğüm seçme ve tur oluşturma

Karınca sisteminde, m adet yapay karınca GSP' de bir tur oluştururular. Başlangıç olarak, karıncalar rastgele seçilen şehirlere bırakılırlar. Çözüm oluşturulan her adımda, k karıncası, bir sonraki adımda hangi şehri ziyaret edeceğine karar vermede, rastgele oransal kural olarak adlandırılan ihtimale dayalı bir hareket etme kuralını uygular. GSP' yi çözmede kullanılan karınca sistem algoritmasının iki ana fazını karıncaların çözüm yapısı ve feromon güncellemesi oluşturur. Belirli olarak, k karıncasının i şehrindeyken j şehrini gideceği şehir olarak seçme ihtimali, aşağıdaki bağıntı ile tanımlıdır:

$$P_j = \frac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{il}(t)]^{\alpha}.[\eta_{il}]^{\beta}}, j \in N_i^k \quad \text{ise, şeklinde ifade edilir.} \qquad (3.1)$$

Bu bağıntı, geçiş kuralı (transition rule) olarak bilinir ve k karıncası için i şehrinden j' ye t' ninci turunu yaparken j' yi seçme ihtimali, başka bir deyişle i ve j şehirleri arasındaki yolun seçilme ihtimalinin hesabıdır.

3.1 denklemindeki değişkenler aşağıda tanımlanmıştır:

N: Düğüm (şehir) sayısı.

J_i^k: k karıncasının i şehrindeyken ziyaret etmesi gereken şehir seti. J_i^k nın kullanılmasıyla k karıncasının bir şehre birden fazla gitmesi önlenir.

d_{ij}: i ve j şehirleri arasındaki uzaklık.

η_{ij}: Uzaklığın tersi, görüş netliği olarak da adlandırılır. ηij =1/dij formülü ile hesaplanır. i şehri varken j' yi seçme istekliliğinin genel kuralını gösterir. Görüş netliği karıncaları araştırmaya yönlendirmede kullanılabilir. Genel kurallar statiktir ve problem çözümü esnasında değişmemektedir.

α ve β: Feromon izinin ve görüş netliğinin bağıl etkisini tanımlayan değişkenlerdir. (α, feromon miktarının bağıl ağırlıklandırmasını, β şehirler arasındaki uzaklığa ait bağıl önemi vurgular.)
τ_{ij}^k (t): t. tur sonunda k karıncasının kullandığı her (i, j) yoluna bıraktığı feromon miktarı

N_i^k: k karıncası i şehrindeyken henüz ziyaret etmediği şehir seti.

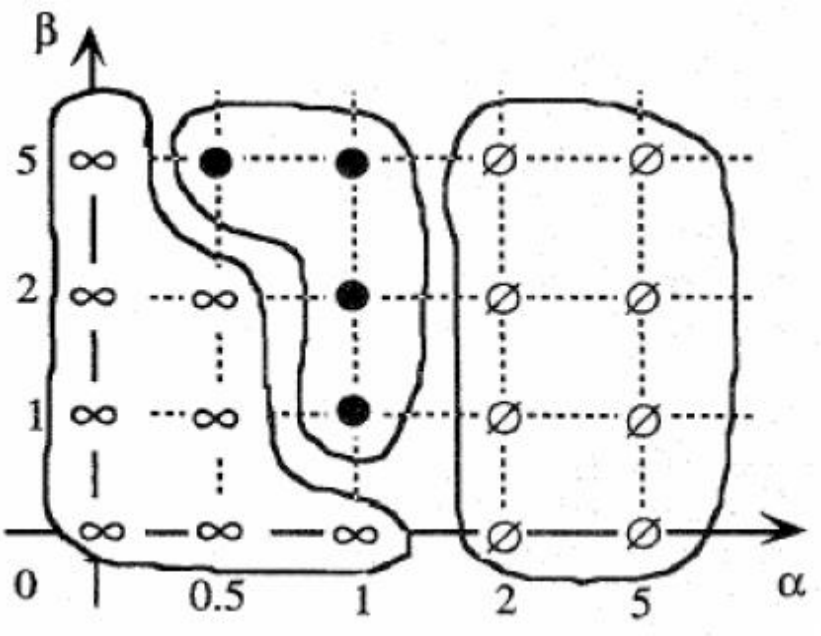
α ve β parametrelerinin farklı kombinasyonları, karınca algoritmasının işleyişini etkilemektedir.

Şekil 3.8' de α ve β değerlerinin değişiminin etkileri görülmektedir:

- Çok yüksek α değerlerinde algoritma, çok iyi çözümleri bulamadan durgunluk davranışına girer. Bu durum olumsuz sonuçlara yol açmaktadır ve Şekil 3.8' de Ø sembolüyle gösterilir.

- Çok düşük α değerlerinde, ki bu da feromon izine çok az değer veya önem verildiği anlamına gelir, algoritma durgunluk davranışı göstermemekle birlikte çok iyi çözümleri de bulamaz. Bu durum şekilde ∞ sembolüyle gösterilir.

- α ve β değerlerinin uygun seçilmesi durumunda çok iyi çözümler bulunur. Şekil 3.8' de merkezi alanda yer alan bu seçim için ● sembolü kullanılmıştır. Bu durumda α ve β' nın farklı kombinasyonları aynı performans seviyesi sonucunu verebilmektedir.



Şekil 3.8: Farklı α ve β parametre değerlerinde oluşan karınca-döngü davranışı [32]

Şekil 3.8' de kullanılan semboller, şunları tanımlamaktadır:

●- Algoritma durgunluk davranışına girmeden en iyi bilinen çözümü bulur.

∞- Algoritma durgunluk davranışına girmeden iyi çözümleri bulamaz.

Ø- Algoritma iyi çözümleri bulamaz ve durgunluk davranışına girer.

Buradan elde edilen sonuçlar algoritmanın formülüyle örtüşmektedir. Yüksek bir α değeri feromon izinin çok önemli olduğu anlamına gelir ve bu yüzden karıncaları geçmişte diğer karıncaların seçtiği yolları seçmeye meyilli kılar.

α ve β parametrelerinin rolü biraz daha incelendiğinde, şunlar söylenebilir:

Eğer $\alpha = 0$ ise, en yakın olan şehirlerin seçilmesi daha olasıdır. Bu klasik rastgeleliliği büyük oranda dikkate alan bir algoritma anlamına gelir. Karıncalar başlangıçta rastgele bütün şehirlere dağıtıldıklarından, birçok başlangıç noktası seçilme olasılığı bulunmaktadır. Eğer $\beta = 0$ ise, sadece feromon yükseltimi devam eder. Sezgisel bir etki olmadan, sadece feromon izi hesaba katılır. Bu genellikle yetersiz sonuçların elde edilmesine yol açar ve özellikle $\alpha > 1$ değerlerinde iken hızlı bir duraklama durumu ortaya çıkar. Duraklama durumu, bütün karıncaların aynı yolu takip ettiği ve aynı turu oluşturduğu bir durumdur. Bu durum da, genellikle, kuvvetli bir ihtimalle optimal olmayan bir turdur.

Her k karıncası, daha önceden ziyaret edilen bütün şehirleri ziyaret edildikleri sıra ve düzende içeren bir M^k hafızasına sahiptir. Bu hafıza, tur oluşturulurken ziyaret edilebilir şehirleri tanımlamada kullanılır. Ek olarak, M^k hafızası, k karıncasına oluşturduğu T^k turunun uzunluğunu hesaplamaya ve yola bırakacağı feromonu belirlemeye yarar.

3.3.3. Feromon izlerinin güncelleştirilmesi

Bütün karıncalar turlarını oluşturup tamamladıklarında, feromon izleri güncellenir. Bu, ilk olarak bütün yollardan sabit bir değer olan feromon buharlaşma değerinin düşülmesi ve daha sonra da karıncaların turlarında geçtikleri yollara feromon eklemeleriyle yapılır. Feromon buharlaşması, algoritmaya şu şekilde dahil edilir:

$$\tau_{ij} \leftarrow (1-\rho).\tau_{ij}, \forall (i, j) \in L \qquad (3.2)$$

Denklem 3.2' deki ifadede [32], ρ feromon buharlaşma oranıdır ve $0<\rho\leq 1$ aralığında seçilir. p parametresi, feromon izlerinin sınırsız bir şekilde birikimini önler ve algoritmanın önceden verdiği kötü kararları "unutmasına" imkan sağlar. Aslında, eğer bir yol karıncalar tarafından seçilmemişse, bu yolun kendi feromon değeri, algoritmanın artan tekrar sayılarıyla logaritmik olarak azalır. Başlangıçta, bütün yollardaki τ_{ij} feromon miktarı, küçük bir pozitif değere ayarlanır. Buharlaşmadan sonra, bütün karıncalar kendi turlarında geçtikleri yollara feromon bırakırlar.

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau^{k}_{ij}, \forall (i, j) \in L \qquad (3.3)$$

Denklem 3.3' deki ifadede, $\Delta\tau_{ij}^{k}$, k karıncasının ziyaret ettiği yollara bıraktığı feromon miktarıdır. Bu miktar şöyle tanımlanır:

$\Delta\tau_{ij}^{k} = 1 / C^{k}$; eğer (i,j) yolu k karıncasının turu (T^{k}) içinde varsa,

(3.4)

$\Delta\tau_{ij}^{k} = 0$; eğer (i,j) yolu k karıncasının turu (T^{k}) içinde yoksa.

Denklem 3.4' deki ifadede, C^{k}, k. karınca tarafından oluşturulan T^{k} turunun uzunluğudur ve T^{k} turuna ait olan yolların uzunluğunun toplanmasıyla hesaplanır.

Denklem 3.4. vasıtasıyla, karıncanın turu daha kısa ise, bu tura ait olan yolların daha fazla feromon alacağı görülebilir. Genellikle, çok karınca tarafından kullanılan ve kısa turların birer parçası olan yollar, daha fazla feromon alırlar ve bu yüzden algoritmanın gelecekteki tekrarlarında bu yolların karıncalar tarafından seçilme ihtimalleri artar.

(i, j) yoluyla ilişkili olan τ_{ij} feromon miktarı, i şehrindeyken j şehrini seçmedeki öğrenilmiş istekliliği ifade eder. Bu, aynı zamanda bir karıncanın oluşturduğu tura ait olan (i, j) yolunu seçmedeki isteklilik anlamına gelir. Feromon izi bilgisi, problem çözümü esnasında karıncaların elde ettiği deneyimi yansıtarak, problem çözümü esnasında değişir. Karıncalar oluşturdukları çözümün kalitesiyle orantılı bir miktarda feromon bırakırlar. Bir karınca tarafından oluşturulan tur ne kadar kısa olursa, bu turu oluştururken kullandığı yollara o kadar fazla miktarda feromon bırakır. Bu seçim, araştırmayı iyi çözümlere doğru yönlendirmeye yardımcı olur. Feromon buharlaşmasının temel rolü, durgunlaşmayı önlemedir.

Her karıncanın hafızası, tabu listesi olarak adlandırılan ve daha önceden ziyaret edilen şehirleri içeren listeyi içerir. Hafıza, i şehrinde bulunan her k karıncası için, ziyaret etmesi gereken şehir setini tanımlamada kullanılır. Bu yüzden her bir k

karıncası, hafızasını kullanarak her bir şehre tam olarak sadece bir defa uğrar. Ayrıca hafıza, karıncanın ziyaret ettiği yollara feromon bırakmasında da rol oynar.

Dorigo, deneysel olarak, başlangıçta bütün yollara sabit pozitif bir feromon değeri atandığında ve toplam karınca sayısı şehir sayısına eşit iken (m = n) ve son olarak da α, β, ve ρ sabitlerine sırasıyla 1,5 ve 0,5 değerleri verildiğinde çalıştırılan algoritmanın iyi sonuçlar verdiğini gözlemiştir [42].

Karınca sistemi, diğer genel amaçlı olan ve bağıl olarak küçük boyuttaki GSP (30-75 arası şehir içeren problemler) ile karşılaştırılmıştır. Sonuçlar çok ilginç ve aynı zamanda hayal kırıklığı oluşturacak niteliktedir. Oliver 30 probleminde, genetik algoritmanın bulduğu en iyi çözümü, karınca sistemi de bulabilmekte ve geliştirebilmektedir. Karşılaştırıldığı bazı genel amaçlı sezgisellere benzer veya daha iyi bir performans verdiği de olmuştur. Maalesef, büyüyen boyutlardaki problemler için, karınca sistemi, izin verilen 3000 tekrar sayısında, bilinen en iyi çözümlere hiçbir zaman erişememiştir. Bu cesaret verici, fakat olağanüstü de olmayan sonuçlar, birçok sayıdaki araştırmacıyı optimizasyon için KKO yaklaşımını daha fazla incelemeye sevketmiştir. Bu çabalar, aşağıda bahsedeceğimiz birçok başarılı uygulamayla sonuçlanmıştır.

3.4. Elit Karınca Sistemi

Başlangıçtaki karınca sistemi yapısına ilave edilen ilk gelişme, karınca sistemi için elit strateji olarak adlandırılır (EAS- Elitist Ant System). Bu yöntem, Dorigo tarafından 1992' de bilim dünyasına tanıtılmış ve önerilmiş olup, yine kendisince 1996' ya kadarki süreçte geliştirilmiştir. Bu yapıdaki temel fikir, algoritmanın başlangıcından bu yana bulunan en iyi tura ait olan yollara kuvvetli bir şekilde feromon eklenmesinin sağlanmasıdır. Bu tur T^{bs} (o ana kadarki en iyi tur) şeklinde ifade edilir.

Eklenen feromon miktarı " e/ C^{bs}" kadardır. Burada e, o ana kadar bulunan en iyi tura verilen ağırlığı tanımlayan bir parametre, C^{bs} de en iyi turun uzunluğudur [44].

3.5. MAX-MIN Karınca Sistemi

1997' de Stützle ve Hoose tarafından tanıtılmıştır. MAX- MIN Karınca Sistemi (MMAS), klasik karınca sistemiyle ilişkili olarak 4 ana farklılık ve gelişim getirir [41-45].

Birincisi, bu yapı kuvvetli bir şekilde olağanüstü bir başarıyla en iyi turları bulur. O turdaki veya o ana kadar olan turlardaki en iyi karıncanın feromon bırakmasına izin verilerek bu başarılır. Bu her iki durumun, her ikisi de dahil edilebilir. Maalesef, böyle bir strateji, bütün karıncaların aynı turu izlediği bir durgunluk durumuna yol açabilir. Çünkü iyi fakat optimal olmayan turda bulunan yollardaki feromon izlerinin aşırı yükselmesi söz konusudur. Bu etkiyi önlemek için, MMAS' in getirdiği ikinci farklılık, ihtimal dahilindeki feromon izi değerlerinin, [τ_{min} τ_{max}] şeklindeki gibi bir aralıkta sınırlanmasıdır. Üçüncüsü, başlangıçta yollarda bulunan feromon izi değerleri, feromon izinin öngörülen maksimum sınırına getirilir. Bu şekilde düşük feromon buharlaşma oranıyla birlikte araştırmanın başlangıcında turların keşfedilme ihtimali arttırılmış olmaktadır. Son olarak, MMAS yönteminde, sistem durgunluğa veya duraklamaya her yaklaştığında veya belirli bir önem sayısındaki tekrarlar boyunca geliştirilmiş tur bulunamazsa, feromon izleri yeniden başlangıç seviyelerine getirilir.

3.6. Rank Temelli Karınca Sistemi

Karınca sistemindeki diğer bir gelişim, karınca sisteminin rütbeye veya dereceye dayalı olan (AS_{rank}) versiyonudur. Bu gelişim Bullnheimer, Hartl ve Strauss tarafından 1999' da tanıtılmıştır [46, 47].

Bu yapıda, her karınca rütbesine göre artan bir miktarda feromon bırakır. İlave olarak, elit karınca sisteminde olduğu gibi, o ana kadarki en iyi karınca her tekrarda daima en yüksek feromonu bırakır [44].

Bu sistemde her turda bir rütbelendirme söz konusudur [48].

3.7. En iyi- En kötü Karınca Sistemi (Best- Worst AS)

Max-Min karınca sistemi ile benzeşen bir feromon güncelleme tekniği vardır. Farkı ise bir tur sonunda en kötü sonucu bulan karıncaya ait feromon izlerin silinmesidir. Bu algoritmanın en büyük farklılığı problemin başlangıç konumunda feromon iz miktarlarının değişken olabilmesidir [49].

3.8. Melez KKO

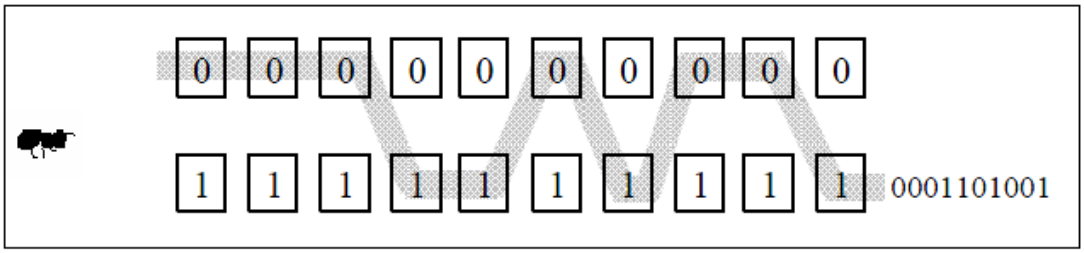
KKO yaklaşımı ile diğer yaklaşımların birleştirilmesinden elde edilen yeni algoritmaların genel adıdır. Gambardella ve Dorigo çalışmalarında, KKA ile MPQ/AI algoritmasını birleştirerek ardışık sipariş problemlerine uygulamışlar ve işlem zamanından %2 ile %30 arasında kazanç sağlamışlardır. Lee [51] kaynak atama problemlerinin çözümünde genetik algoritma ile KKO' yu birleştirerek elde ettiği yeni algoritma ile çözerek iyi sonuçlar elde etmiştir [50, 51].

3.9. GACO (Global Ant Colony Optimization) Algoritması

GACO, tüm düğümlere uğrama kısıtını içermeyen problemlerin çözümüne yönelik olarak geliştirilmiş olan bir KKO algoritmasıdır. Bu problemlere örnek olarak en kısa yol problemleri gösterilebilir. Geliştirilen algoritma, en kısa yol problemi olarak modellenen [52] GSP' ye uygulanmıştır.

3.10. TACO (Touring Ant Colony Optimization) Algoritması

Bu algoritma Hiroyasu ve arkadaşları tarafından özellikle sürekli optimizasyon problemlerinin çözümü için önerilmiştir [57]. Bu algoritmada çözümler ikili sayılarla temsil edilmiş tasarım parametrelerinin bir vektörüdür. Dolayısıyla bir çözüm, ikili sayıların alt gruplarından oluşan bir vektördür. Bu nedenle, her bir yapay karınca dizideki ikili sayıların değerini araştırır. Başka bir deyişle ikili sayının değerinin 1 ya da 0 olup olmadığına karar vermeye çalışır. TACO algoritması kavramı Şekil 3.9' da gösterilmiştir.



Şekil 3.9: Bir karınca tarafından bulunan yapay bir yol (çözüm) [49]

Bir ikili sayının değeri için karar verme aşamasında, karıncalar sadece feromon bilgisini kullanırlar. Bir dizideki tüm ikili sayıların değeri için karar verdiğinde problem için bir çözüm üretiliş demektir. Bu çözüm, problem için değerlendirilir ve kalite fonksiyonu olarak adlandırılan bir fonksiyon aracılığı ile çözme ilişkin bir kalite değeri elde edilir. Bu değere bağlı olarak bir yapay feromon maddesi miktarı ikili sayılar arasında oluşan yapay bütün yollara yapıştırılır [53].

3.11. Sürekli KKO (CACO)

Bu algoritma literatürde Continuous ACO (CACO) olarak bilinmektedir. CACO, Mark Wodrich ve George Bilchev tarafından 1997 yılında geliştirilmiştir [54, 55]. CACO karınca algoritmaları içerisinde araştırma uzayı sürekli olan problemlerde kullanılmak üzere geliştirilmiş ilk karınca algoritmasıdır.

Bu algoritmada küresel ve bölgesel olmak üzere iki ayrı arama metodu kullanılır ve buna uygun olarak karıncalar küresel ve bölgesel olmak üzere iki sınıfa ayrılır. Küresel arama küresel karıncalar tarafından icra edilir. Küresel karıncalar toplam karınca sayısının %80' ini oluşturur. Başlangıçta önce küresel karıncalar araştırma uzayında rastgele dağılır. Daha sonra küresel karıncaların vektörel büyüklüklerle belirledikleri bölgelere bölgenin uygunluk değeri ile orantılı sayıda bölgesel karıncalar gönderilir. Bölgesel karıncalar umut verici görülen bölgede daha hassas arama yapar. İyi çözüm bulunduğu zaman çözümü bulan bölgesel karınca seçtiği yönün feromon değerini günceller ve daha iyi çözümün bulunduğu noktaya daha çok sayıda bölgesel karınca çekilmek istenir.

Küresel aramada araştırma uzayı bölgelere ayrılır. Her bir bölge gerçek sayı değerli vektörel büyüklüklerle tanımlanır. Başlangıçta bölgeler araştırma uzayında

rastgele seçilir. Daha sonra bölgenin uygunluğu değerlendirilir ve bu uygunluk değerine göre bölgeyi temsil eden vektörler değişim gösterir. Bölgesel minimumdan kaçınmak için küresel karıncalar geniş alanlara uzanabilme yeteneğine sahiptir. Bölgeleri temsil eden vektörlerden yeni vektörler üretmek ve küresel aramayı mümkün kılmak için GA' dan esinlenen çaprazlama benzeri bir fonksiyon kullanılır. Algoritmada kullanılan küresel arama metodunun karınca sistemi ile bir ilgisi yoktur [56, 57].

Bölgesel karıncalar küresel karıncalar tarafından uygun bulunan bölgelerde kokuya dayalı arama yapar. Bölgesel bir karınca koku miktarı ile orantılı bir olasılıkla kendine bir bölge seçer. Seçilen bölge kısa mesafeli vektörel büyüklüklerle aranır. Her bölgede arama yapan son bölgesel karıncanın arama vektörü muhafaza edilir. Eğer son karınca daha iyi bir çözüm noktası bulduysa yeni karınca bu noktadan itibaren aramaya başlar. Gelişim sağlanmadıysa yeni karınca rastgele bir yön seçer.

Sonradan gelen bölgesel karıncaların ne kadar uzağa erişebileceğini belirlemek üzere bir sayaç parametresi kullanılır. Çözümü geliştiren bölgesel karınca seçtiği yönün feromon miktarım arttırır ve bu yönün sayaç değerini bir azaltır. Bir yönün sayaç değeri azaldıkça o yönü seçen karıncanın aynı yönden itibaren uzanabileceği mesafe miktarı artar. Bölgesel karıncaların uzanabileceği mesafe miktarım değiştiren bir diğer yaklaşım BT tabanlıdır. Bu yaklaşıma göre BT' nin sıcaklık parametresine benzer biçimde arama mesafesi iterasyon sayısı arttıkça azaltılır.

Bu algoritmanın zayıf tarafı karıncaların yanlışlıkla bir noktayı birçok defa arama hatasına düşmesidir. Bu durum geniş araştırma uzayında bütün bölgeleri hafızada tutmanın zor olmasından kaynaklanmaktadır [58].

3.12. Apicalis Algoritması (API)

API algoritması Monmarche ve arkadaşları tarafından 2000 yılında önerilmiştir. Bu algoritma Pachycondyla Apicalis adında özel bir karınca türünün davranış

biçimine dayanır. Koloni araştırma uzayında önce rastgele konumlandırılır ve karıncalar başlangıç bölgesinden itibaren araştırma uzayını rastgele seçilmiş yönlerde aramaya başlar. Karıncalar yönlendirildikleri noktada bölgesel aramayı paralel yapıda gerçekleştirir ve her karınca bulduğu en iyi noktayı hafızasında tutar. Daha sonra bütün karıncalar bilgi değişimi yapar ve yuva olarak adlandırılan koloni merkezi, bulunan en iyi noktaya taşınır ve bu yeni nokta etrafında arama süreci tekrar başlar.

Algoritma karınca tabanlıdır ve bütün karıncalar için bir tür küresel hafıza kullanılır. Metot rastgele arama yöntemine benzer ve az sayıda parametre kullanır. Fakat bu algoritmada aramayı yönlendirmek için koku kullanılmaz. Çevreye dayalı etkileşim (stigmergy) yoktur. Bu nedenle algoritma tam bir karınca modeli olarak görülmemektedir [59, 60].

3.13. Karınca Çevrim, Karınca Yoğunluk ve Karınca Miktar Yöntemleri

Köken olarak karınca sisteminden türetilmiş üç algoritma seti daha vardır. Bunlar orijinal isimleriyle verilirse Ant Cycle-Karınca Çevrim, Ant Density- Karınca Yoğunluk ve Ant Quantity-Karınca Miktar algoritmalarıdır. Bu sistemler arasındaki farklılık da feromon bırakımından kaynaklanmaktadır. Karınca Çevrimi' nde feromon, her karıncanın yaptığı tam bir tur sonunda tur uzunluğunun tersi kadar olacak şekilde, bir önceki aynı karıncanın turunda geçilen yollardan tekrar geçilenlerine bırakılır.

Karınca ajanlarının gerçek karıncalardan farklı olarak turları esnasında geçtikleri yolları tuttukları hafızaları vardır. Karınca Yoğunluk ve Karınca Miktar' da ise, her karınca tur sonunu beklemeden her adımında feromonunu bırakır. Karınca Yoğunluk modelinde, her bir karınca i' den j' ye her gidişinde Q kadar sabit bir miktardaki feromonu (i, j) yoluna bırakır. Karınca Miktar modelinde ise, i' den j' ye giden karınca Q/ d_{ij} kadar bir feromonu (i, j) yoluna bırakır. Karınca Çevrim algoritması birkaç test probleminde diğer ikisinden daha iyi bir sonuç verdiğinden, karınca sistemiyle ilgili bir referans verildiğinde kullanılan genel bir sistem olmuştur. Paralel

karınca kolonisi sisteminde birden fazla düğüm var olduğu düşünülürse ve her bir karıncanın belirli bir anda bir düğümde çalıştığı varsayılırsa, karıncalar paralel sistemlerdeki akıllı aracılar olarak nitelendirilebilirler.

Karıncalar, rastgele seçilen bir şehirden başlayarak bütün şehirleri ziyaret ederler ve bu sırada geçtikleri yollara feromon bırakırlar. Bu feromonlar, sonraki karıncaların yollarını belirlemelerinde etkili olmaktadır. Aracılar arasındaki iletişim, ortak çözümü oluşturmaktadır. Karıncaların her birini bir düğüme gönderirken dikkat edilmesi gereken faktör, yarışma şartlarının meydana gelmesini engellemektir. Yarışma şartları, aynı anda farklı düğümlerde çalışan karıncaların, feromon ve seçilen yol verilerini tutan global veri yapısını aynı anda değiştirememeleri durumudur. Buradaki temel amaç, KKO algoritması içerisinde olasılık hesabında kullanılan parametrik α, β ve ρ değerlerinin en uygunlarını bulmak olduğu için kolonideki karınca sayısının artırılması, bu değerlerin en uygunlarını bulma olasılığını artıracaktır [32].

3.14. Karınca Koloni Sistemi (ACS-Ant Colony System) Algoritması

Karınca sisteminde çalışan KKO algoritmalarının yapısında bulunmayan yeni mekanizmalar ve yaklaşımlar, algoritmaya dahil edilerek karınca koloni sistemi algoritması oluşturulmuştur. Bu şekilde daha yüksek optimizasyon başarısı elde edilmektedir.

Karınca koloni sistemi, 1996' da Dorigo ve Gambardella tarafından tanıtılmıştır [34-36]. Sadece sınırlı sayıda düğüm içeren problemler için gerçekçi bir sürede iyi çözümler sunabilen karınca sisteminin performansının geliştirilmesi amaçlanmıştır. Karınca koloni sistemi, karınca sisteminden dört ana noktada farklılık gösterir [12, 41, 61].

İlki, karınca sisteminden daha atılgan bir seçme kuralı uygulanmasıyla, karınca sistemindeki karıncaların araştırma birikimi olağanüstü bir başarıya dönüştürülmektedir.

İkincisi, feromon buharlaşması ve feromon bırakımı, sadece o ana kadarki en iyi tura ait olan yollarda gerçekleşir. Bir tur sonunda bütün karıncalar bir çözüm oluşturduklarında, feromon izi, tekrarın başlangıcından o yana en iyi turu bulan karıncanın kullandığı yollara ilave edilir.

Üçüncüsü, karıncaların i şehrinden j şehrine hareket ederken kullandıkları bir (i, j) yolundaki feromonun bir kısmı, her defasında, alternatif yolların keşfini arttırmak için, azaltılır. Bu şekilde farklı lokal bir feromon güncelleme gerçekleştirilir.

Dördüncüsü, ziyaret edilecek bir sonraki şehrin seçimini sınırlamak için aday listeleri kullanılır.

Tablo 3.1' de, karınca sistemi ile karınca koloni sisteminin özellikleri karşılaştırmalı olarak verilmiştir.

Tablo 3.1: Karınca sistemi ile karınca koloni sisteminin genel olarak karşılaştırılması [32]

	Karınca Sistemi	**Karınca Koloni Sistemi**
Düğüm Seçme	Feromonu yüksek olan yol seçilir, yollardaki feromon eşitse rastgele bir yol seçilir	Karınca Sistemi ile aynı
Feromon Güncelleme	Bütün yollarda, yol uzunluklarının tersi kadar feromon yollara bırakılır, sabit bir miktarda buharlaştırılır.	Sadece o anda tanımlanan yollardaki en kısa yola ait olan yollara feromon bırakılır ve buharlaşma uygulanır.
Düğüm Sayısı	30 düğüme kadar çok iyi sonuç verir, 70 ve üzerindeki düğümlerde iyi sonuçlardan uzaklaşır.	100 düğüm ve üzerindeki daha büyük problemlerde iyi sonuç verir.
Başarı	Çözümlenen düğüm sayısına göre başarı sınırlı olabilmektedir.	Yüzlerce düğüme kadar oldukça yüksek bir başarı sağlamaktadır.

4. KARINCA KOLONİSİ ALGORİTMASININ GEZGİN SATICI PROBLEMİ (GSP)' NE UYGULANMASI

Bu bölümde, karınca kolonisi algoritmalarının daha iyi anlaşılması amacıyla, Söyler, H. ve Keskintürk, T. tarafından 2007 yılında yayınlanan "Karınca Kolonisi Algoritması ile Gezen Satıcı Probleminin Çözümü" adlı makaledeki uygulamaya yer verilmiştir.

GSP, başlanan noktaya dönülmek üzere, n adet düğümün sadece bir kere ziyaret edilerek, toplam mesafe, maliyet veya sürenin minimizasyonunu sağlama problemi olarak tanımlanır [45].

Yolların uzunlukları d_{ij} olmak üzere, simetrik GSP' de $d_{ij} = d_{ji}$, asimetrik GSP' de $d_{ij} \neq d_{ji}$' dir. Amaç uygun yollardan ilerleyerek tur uzunluğunu minimize etmektir. GSP' nin çözümüne yönelik birçok yöntem geliştirilmiştir. Dal-Sınır yöntemi de, bütün alternatif rotalar kıyaslanarak içlerinden en iyi olanı seçilir. Ancak düğüm sayısı arttıkça, kıyaslanacak devre sayısı üstel olarak artmaktadır.

Örneğin n=10 düğümlü yönlü olmayan bir şebekede (n -1)! / 2 = 181.440 kadar farklı devre, aynı düğüm sayısına sahip yönlü bir şebekede ise (n -1)! = 362.880 farklı devre mevcut olacaktır. Bu örnekten de anlaşılacağı gibi şebeke yönlü veya yönsüz olsun, düğüm sayısı arttıkça devrelerin tümünü kıyaslamak imkansız hale gelmektedir [12].

Örnek uygulama için kıyaslanacak rota alternatifi sayısı 81! = 5,79712602074737E+120 olacaktır. Bunun dışında optimum çözümü garanti etmeyen yaklaşık çözüm yöntemleri mevcuttur.

Bunlardan bazıları İlave Etme Algoritması (Insertion Algorithm), En Yakın Komşu Algoritması (Nearest Neighbor), İki Yol Değiştirme Algoritması (Two-way

Exchange Improvement Heuristic)' dır. Ayrıca GSP' nin çözümünde Tamsayılı Programlama ve Dinamik Programlama da kullanılmaktadır.

Karınca koloni algoritmasının GSP' ye uygulanmasında feromon izleri ve sezgisel bilgiler kullanılır. İz yoğunluğu (τ^t_{ij}) algoritmanın adımları boyunca değiştirilen nümerik bilgilerdir.

Başlangıçta her m karınca rastsal olarak seçilen şehirlere yerleştirilir ve iteratif olarak her şehre geçiş kuralı uygulanır. i şehrindeki karınca henüz gidilmemiş j şehrini, iz yoğunluğu (τ^t_{ij}) ve şehirler arasındaki uzunluğun fonksiyonuna bağlı bir olasılığa bağlı olarak seçer. Karınca büyük olasılıkla kendisine daha yakın olan şehri ve yüksek feromon izine sahip hattı seçer. Uygun bir çözümde her karınca tabu listesi olarak adlandırılan sınırlı bir hafızaya sahiptir. Hafıza henüz gidilmemiş şehirlere gidilmesini sağlar ve uygun çözümün sağlanmasını garanti eder. Tüm karıncalar bir turu tamamladıktan sonra feromonlar yenilenir. Başlangıçta çok düşük sabit bir feromon düzeyi alınır [62].

4.1. Uygulama

Karınca kolonisi algoritmasının ilk uygulaması gezgin satıcı problemi üzerine olmuştur. Bunun nedeni en kısa yol mantığının karınca kolonisi davranışlarıyla birebir uyumlu olması ve kolayca adapte edilebilir olmasıdır. Ayrıca kolayca anlaşılabilir olması ve çok karmaşık teknikler ve hesaplar gerektirmemesi de algoritmanın en çok uygulandığı problem olmasını sağlamıştır [52]. Geniş bir uygulama alanı olması ve bahsedilen avantajlarından dolayı bu çalışmada da karınca kolonisi algoritmasının GSP' ye uygulamasına yer verilmiştir.

Örnek problem, Türkiye' nin herhangi bir ilinden yola çıkılarak ve tüm şehirler ziyaret edilerek tekrar başlangıçtaki noktaya dönüş şeklinde düşünülmüştür.

Gerçekleştirilecek böyle bir Türkiye turunun, en kısa şekilde nasıl yapılabileceği GSP olarak ve karınca kolonisi algoritması ile belirlenmiştir. Öncelikle Türkiye' deki

81 ile ait mesafeler tablosu oluşturulmuştur. Tüm illerin birbirlerine olan karayolu uzaklığı kilometre cinsinden tablolaştırılmıştır. Simetrik GSP olduğundan bütün d_{ij} ve d_{ji} ler birbirlerine eşittir. 81 ile ait mesafeler tablosunun bir kısmı örnek olarak Tablo 4.1' de gösterilmiştir.

Tablo 4.1: Mesafeler tablosundan örnek bir parça [62]

	Adana	Adıyaman	Afyon	Ağrı	Amasya	Ankara
Adana	0	330	573	966	612	490
Adıyaman	330	0	903	648	636	757
Afyon	573	903	0	1314	593	257
Ağrı	966	648	1314	0	734	1057
Amasya	612	636	593	734	0	336
Ankara	490	757	257	1057	336	0

Mesafeler tablosu oluşturulduktan sonra problemin çözümü için tasarlanan karınca kolonisi algoritması Microsoft Excel' de Visual Basic Macro diliyle yazılmıştır. Problemin çözümünde lokal feromon güncellemesiyle birlikte global feromon güncellemesi de kullanılmıştır. Bilinen algoritmalardan farklı olarak her bir iterasyondaki en uygun turu gerçekleştiren karınca bir sonraki iterasyona aktarılmış, böylelikle bulunan en iyi turun korunması sağlanmıştır. En iyi tur korunduğundan q değerinin düşmesinin avantaj sağlayacağı düşünülmüştür. Bununla aramanın hızlandırılması sağlanmıştır. Tüm iller plaka numaralarına göre 1' den 81' e kadar kodlandıktan sonra algoritma ile ilgili parametreler yapılan araştırmalar ve denemeler sonucunda Tablo 4.2' deki gibi belirlenmiştir.

Tablo 4.2: Kullanılan parametreler [62]

Parametre	Değer
Global güncelleme paydası	100
Buharlaşma oranı	0,1
q değeri	15
Karınca sayısı	81
İterasyon sayısı	1000
Alfa	0,80
Beta	4

Programın çalışırken izlediği yol aşağıda adım adım belirtilmiştir:

Adım 1: Başlangıç feromon değerleri belirlenir.

Adım 2: Karıncalar her düğüme rastsal olarak yerleştirilir.

Adım 3: Her karınca, sonraki şehri denklemde verilen lokal arama olasılığına bağlı olarak seçmek suretiyle turunu tamamlar.

Adım 4: Her karınca tarafından katedilen yolların uzunluğunu hesaplanır ve lokal feromon güncellemesi yapılır.

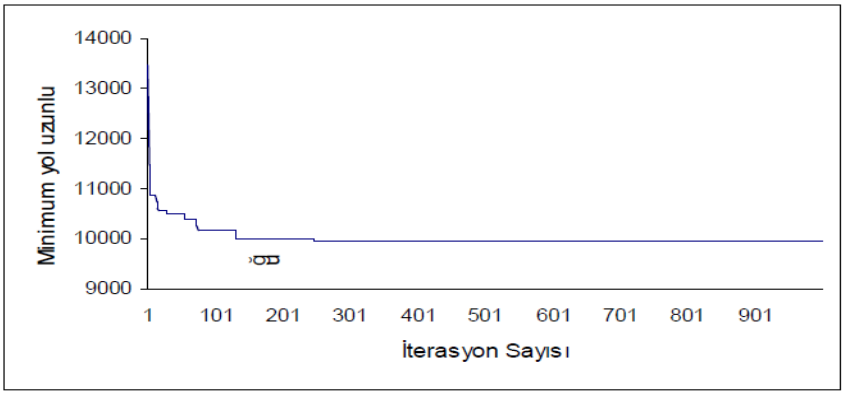
Adım 5: En iyi çözüm hesaplanır ve global feromon yenilemesinde kullanılır.

Adım 6: Maksimum iterasyon sayısı ya da yeterlilik kriteri sağlanana kadar Adım 2' ye gidilir.

Program belirlenen iterasyon sayısı kadar çalıştırılmış ve optimum sonuç olan 9954 değeri 241 inci iterasyonda bulunmuştur. Bulunan bu en iyi sonuca ait çözüm değerleri Tablo 4.3' te gösterilmiştir [62].

Tablo 4.3: İllere göre en kısa Türkiye turu [62]

Çıkış noktası		Varış noktası	Çıkış noktası		Varış noktası	Çıkış noktası		Varış noktası	Çıkış noktası		Varış noktası
Çorum	-	Yozgat	Edirne	-	Kırklareli	Rize	-	Artvin	Batman	-	Diyarbakır
Yozgat	-	Kırşehir	Kırklareli	-	Tekirdağ	Artvin	-	Ardahan	Diyarbakır	-	Mardin
Kırşehir	-	Kırıkkale	Tekirdağ	-	İstanbul	Ardahan	-	Kars	Mardin	-	Şanlıurfa
Kırıkkale	-	Çankırı	İstanbul	-	Kocaeli	Kars	-	Iğdır	Şanlıurfa	-	Adıyaman
Çankırı	-	Ankara	Kocaeli	-	Yalova	Iğdır	-	Ağrı	Adıyaman	-	K.Maraş
Ankara	-	Eskişehir	Yalova	-	Bursa	Ağrı	-	Erzurum	K.Maraş	-	Gaziantep
Eskişehir	-	Kütahya	Bursa	-	Bilecik	Erzurum	-	Bayburt	Gaziantep	-	Kilis
Kütahya	-	Afyon	Bilecik	-	Sakarya	Bayburt	-	Gümüşhane	Kilis	-	Hatay
Afyon	-	Uşak	Sakarya	-	Bolu	Gümüşhane	-	Erzincan	Hatay	-	Osmaniye
Uşak	-	Isparta	Bolu	-	Düzce	Erzincan	-	Tunceli	Osmaniye	-	Adana
Isparta	-	Burdur	Düzce	-	Zonguldak	Tunceli	-	Malatya	Adana	-	İçel
Burdur	-	Antalya	Zonguldak	-	Bartın	Malatya	-	Elazığ	İçel	-	Karaman
Antalya	-	Denizli	Bartın	-	Karabük	Elazığ	-	Bingöl	Karaman	-	Konya
Denizli	-	Muğla	Karabük	-	Kastamonu	Bingöl	-	Muş	Konya	-	Aksaray
Muğla	-	Aydın	Kastamonu	-	Sinop	Muş	-	Bitlis	Aksaray	-	Niğde
Aydın	-	İzmir	Sinop	-	Samsun	Bitlis	-	Van	Niğde	-	Nevşehir
İzmir	-	Manisa	Samsun	-	Ordu	Van	-	Hakkari	Nevşehir	-	Kayseri
Manisa	-	Balıkesir	Ordu	-	Giresun	Hakkari	-	Şırnak	Kayseri	-	Sivas
Balıkesir	-	Çanakkale	Giresun	-	Trabzon	Şırnak	-	Siirt	Sivas	-	Tokat
Çanakkale	-	Edirne	Trabzon	-	Rize	Siirt	-	Batman	Tokat	-	Amasya
									Amasya		Çorum



Şekil 4.1: İterasyonlar boyunca minimum yol uzunluğu [62]

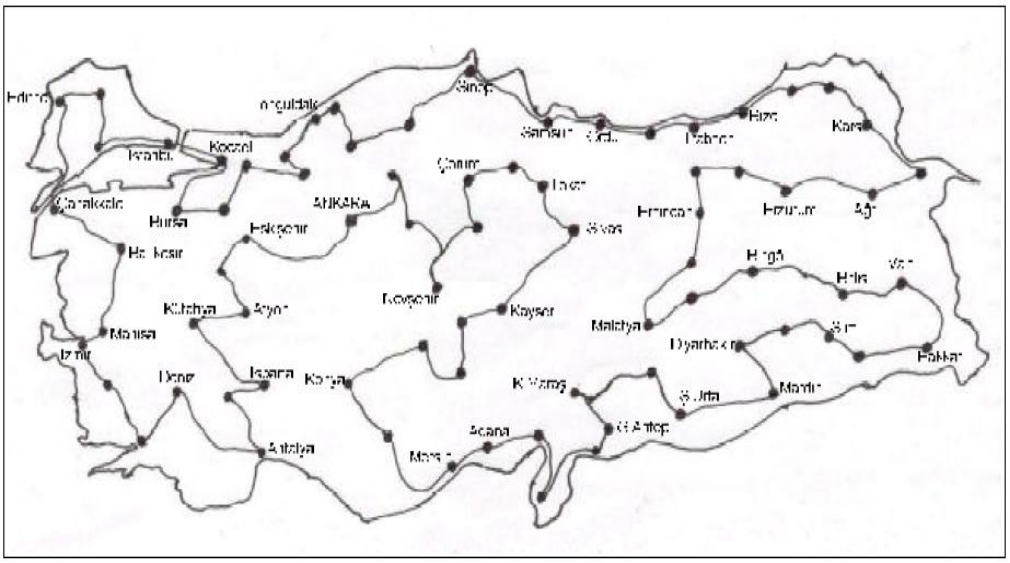
GSP' de karınca kolonisi algoritmasının, diğer sezgisel yöntemlere göre daha iyi ve daha hızlı sonuç verdiği bilinmektedir [53].

Tablo 4.4' te KKA ve diğer yöntemlerin sonuç ve süre karşılaştırmalarına yer verilmiştir.

Tablo 4.4: KKA ve diğer yöntemlerin karşılaştırılması [62]

Yöntem	Sonuç
KKA	9954
İlave Etme Algoritması	12049
En Yakın Komşu Algoritması	13120
İki Yol Değiştirme Algoritması	11136

Çözüme ait güzergah Şekil 4.2' de gösterilmiştir. En kısa yoldan Türkiye turu yapmak isteyen bir kişi şekilde belirtilen yolu takip etmek zorundadır. Başlangıç noktası olarak her bir il seçilebilir. Çünkü yol başlangıç noktasından başlayıp yine aynı yere döndüğünden tüm noktalar (iller) için de geri dönüş söz konusudur.



Şekil 4.2: Bulunan sonuca göre Türkiye turu [62]

4.2. Uygulama Sonuçları

Dağıtım, günümüz ekonomik hayatında hem rekabet avantajı hem de maliyetler üzerindeki etkisi bakımından önemini arttırmaktadır. Maliyetler ve zaman açısından dağıtımın en uygun şekilde yapılabilmesi için geliştirilmiş birçok teknikten faydalanılmaktadır. Bunlardan en yenisi olan karınca kolonisi algoritması GSP' ye benzer durumlar için oldukça hızlı ve iyi sonuçlar vermektedir.

İyi tasarlanmış ve uygulamaya yönelik kolaylıkları da bünyesinde barındıran, KKA ile geliştirilmiş bir dağıtım programı ile kurumların iyi ve hızlı sonuç alacağı muhakkaktır.

Birçok alanda ve kurumda kullanılmaya başlayan KKA' nın çalışmadaki uygulamaya benzer turistik gezilerin planlanmasında, lojistik firmalarının ve ürünlerini belli noktalara ulaştırmak zorunda olan tüm işletmelerin dağıtım planlarında, yol yapım çalışmalarında vb. alanlarda başarıyla kullanılabileceği düşünülmektedir [54].

5. İŞ AKIŞ PROBLEMLERİNİN KARINCA KOLONİ SİSTEMİ ALGORİTMASI İLE ÇÖZÜMÜ

Bu tez çalışmasında, seri imalat yapan işletmelerde kullanılan PTİA problemlerinin çözümü için, performans yönünden diğer karınca algoritmalarına kıyasla üstünlük gösteren "karınca koloni sistemi algoritması (KKS)" kullanılmıştır. Ayrıca KKS de ilk adımda oluşturulan sıralama işlemleri için rastgele bir dizilim oluşturmak yerine PTİA problemlerinde oldukça yüksek başarıya sahip olan FRB3 algoritması kullanılmıştır. Daha sonraki adımlarda oluşturulan çözümleri iyileştirmek için yine FRB3 algoritmasından yararlanılmıştır.

Problemlerin çözümünde kullanılan, KKS algoritmasını içeren program Microsoft Visual C# 2008 programlama dilinde kodlanmıştır. Problem çözümleri Intel Core 2 Duo CPU (P8600) 2.4 GHz işlemcili, 3 GB Ram bulunan bir bilgisayarda yapılmıştır. Öne sürülen algoritmanın işleyiş adımları ve klasik karınca algoritmasından farklılıkları 3. Bölümde incelenmiştir.

PTİA problemlerinde, tüm işlerin ve tüm makinelerin $t = 0$ anında hazır oldukları kabul edilir. Bir iş bir makineye atandığında, bitirilmeden aynı makinede başka bir iş yapılamaz. Makinelerin hazırlanma süreleri işlem süresinin içindedir ve işlerin makineler arasındaki taşıma süresi ihmal edilir.

KKS' nin PTİA problemlerine uygulanabilmesi için öncelikle KKS algoritmasında kullanılan parametre değerlerinin bazılarının PTİA problemlerine uyarlanması gerekmektedir. Tablo 5.1' de parametrelerin uyarlanmış hali gösterilmiştir.

Tablo 5.1: KKS' de GSP için kullanılan parametre isimlerinin PTİA problemlerine uyarlanması

KKS' de GSP için kullanılan parametre isimleri	KKS' de PTİA problemleri için kullanılan parametre isimleri
Karınca	İş
Şehir	Makine
Yol	İşlerin sıralaması
Toplam yol mesafesi	Toplam işlenme süresi

PTİA problemlerinin çözümü için geliştirilmiş KKS algoritmasının genel yapısı aşağıda verilmektedir:

Adım 1: Karınca kolonisi oluşturulur, parametre değerleri belirlenir. İlk karıncanın izleyeceği yol (makinelerde işlenecek olan işlerin sıralaması) FRB3 algoritmasına göre belirlenir. Bu tur için maliyet (toplam işlenme süresi) hesaplanır.

Adım 2: Düğümler üzerindeki feromon miktarı, global feromon güncelleme kuralına göre arttırılır. KKS algoritmasında, yalnızca en iyi çözüm için feromon güncelleme işlemi yapıldığından, feromon buharlaştırma işlemine gereksinim duyulmaz.

Adım 3: Karıncaların her biri, bir tur oluşturana kadar, düğüm seçme kuralını uygulayarak, bir sonraki işe yönlendirilir.

Adım 4: Karıncanın bir önceki konumundaki feromon miktarı, lokal feromon güncelleme kuralına göre arttırılır. Tüm karıncalar tam bir tur oluşturana kadar Adım 3' e dönülür.

Adım 5: Güncellenen feromon miktarları dikkate alınarak, oluşturulan sıralama işlemi FRB3 algoritması kullanılarak iyileştirilir.

Adım 6: Kolonideki karıncaların her biri bir çözüm ürettikten sonra maliyet diğer bir deyişle toplam işlenme süresi hesaplanır, varsa en iyi çözüm sıralaması güncellenir ve tüm karıncalar ilk pozisyonlarına geri dönerler.

Adım7: Yalnızca en iyi sonuç alınan sıralamadaki düğümler üzerindeki feromon miktarı, global feromon güncelleme kuralına göre arttırılır.

Adım 8: İstenilen kıstas sağlanana kadar Adım 3' e dönülür.

5.1. KKS' de İş Sıralama Kuralı

KKS' de iş sıralama kuralı, açık ve kesin bir keşfetme davranışının ortaya çıkmasını sağlamak için geliştirilmiştir. i makinesindeki k karıncası, j' ye geçişini aşağıdaki ifadedeki kurala göre seçimini yaparak gerçekleştirir:

$$P_{ij}^{k}(t) = \begin{cases} J, & \text{eğer } q > q_0 \text{ise} \\ \arg\max_{l \in N_i^k} \left\{ \tau_{il} \left[\eta_{il} \right]^{\beta} \right\} & \text{eğer } q \le q_0 \text{ise} \end{cases} \qquad (5.1)$$

Bu ifadedeki J, denklem 5.2. ile tanımlı düğüm seçme kuralı olan P_{ij}^{k}' dır.

$$P_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{il}(t)]^{\alpha}.[\eta_{il}]^{\beta}}, \; j \in N_i^k \text{ ise} \qquad (5.2)$$

N: Düğüm (makine) sayısı

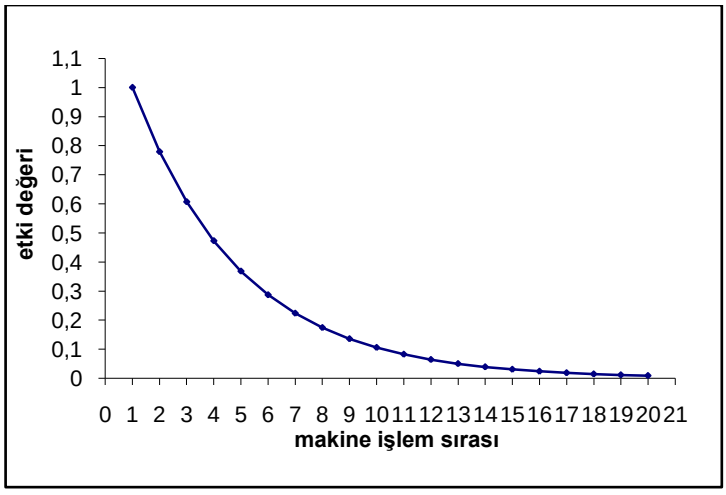
N_i^k: k karıncası (işi) i makinesinde işlenirken henüz işlenmediği makine seti

J_i^k: k karıncasının i makinesinden sonra takip edeceği makine seti. J_i^k' nın kullanılmasıyla k karıncasının bir makineye birden fazla uğraması önlenir.

η_{ij}: Sezgisel fonksiyon. Shortest Process Time (SPT) & Johnson algoritmaları kullanılarak oluşturulmuştur.

$$\underbrace{1/C_j + \sum_{i=1}^{m}(1/p_{ij})e^{-5i/m}}_{\text{I. Kısım II. Kısım}} \qquad (5.3)$$

Şekil 5.1' de C_j, j işinin toplam işlem süresi, P_{ij}, j işinin i makinesindeki işlem süresi, m toplam makine sayısını ifade etmektedir. Formülde işlem süreleri $e^{-5i/m}$ üstel fonksiyonu ile çarpılmıştır. Bu üstel fonksiyon ile j işinin her bir i makinesindeki katkısı makinede ilerledikçe azalır ve ilk makinelerdeki iş kısalığı ne kadar fazla ise o işin sezgiseli daha kuvvetli olmaktadır. $e^{-5i/m}$ fonksiyonunun grafiği aşağıdaki gibidir:



Şekil 5.1: 20 makine için fonksiyonun etki değer grafiği

Formülün birinci kısmı SPT algoritması ("Tüm makinelerde en kısa sürede işlenen iş daha önce işlenmelidir"), formülün ikinci kısmı ise Johnson algoritması ("Başlangıçtaki makinelerde, sondaki makinelerden daha kısa sürede işlenen iş önce işlenmelidir") kullanılarak oluşturulmuştur [63].

α ve β: Feromon izinin ve sezgisel bilginin (değerin) bağıl etkisini tanımlayan değişkenlerdir. α, feromon miktarının bağıl ağırlıklandırmasını, β makinelerdeki

işlem sürelerine ait bağıl önemi vurgulamaktadır. KKS de $\alpha = 1$ alınmıştır. β parametresinin farklı kombinasyonları, karınca algoritmasının işleyişini etkilemektedir.

$\tau_{ij}^{k}(t)$: t. tur sonunda k işinin işlendiği her (i, j) makinesine bıraktığı feromon miktarı.

Çalışmamızda kullanılan feromon miktarı bırakma işlemi aşağıdaki örnekle açıklanmıştır:

k karıncası 2-3-1-5-4 şeklindeki turunu güncellerken 1-2, 2-3, 3-1, 4-5, 5-4 (birinci indeks işin kaçıncı sırada işletildiği, ikinci indeks kaçıncı işin işletildiği) şeklinde depolanmaktadır. Feromon depolama tablosundaki güncelleme Tablo 5.2' de gösterilmiştir.

Tablo 5.2: Feromon depolama işleminde güncellemelerin gösterimi

İş sırası / işler	1	2	3	4	5
1					
2					
3					
4					
5					

Denklem 5.1. ifadesindeki q, 0-1 aralığında rastgele seçilen bir değişkendir. q_0 ise, $(0 \leq q_0 \leq 1)$ aralığındaki değerlerde seçilebilen bir parametredir. Burada bütün karıncalar, q_0 ihtimali ile, sezgisel bilginin ve öğrenilmiş feromon izlerinin gösterdiği sonuca göre ihtimal dahilindeki en iyi hareketini yaparlar. Denklem 5.1' deki "arg max" bu anlamdadır. Bu durumda karıncalar da öğrenilmiş bilgiye önem vermiş, bu bilgiye göre hareketlerini düzenlemiş olurlar. $1-q_0$ ihtimali ile de isteklice yolların keşfini yaparlar. q_0 parametresini ayarlama, keşfetme derecesinin değişim imkanını sağlar ve araştırmanın o ana kadarki en iyi çözüm etrafına mı yoksa turların keşfi etrafına mı odaklanmasını seçmemize imkan sağlar [63].

Buradan çıkarılması gereken sonuç şudur:

Eğer $q \leq q_0$ ise, yeni bir kurala göre seçme ihtimalimiz hesaplanmakta;

Eğer $q>q_0$ ise, karınca sistemindeki hesaplama sadece $\alpha=1$ alınarak aynen kullanılmaktadır. Başka bir deyişle, $q \leq q_0$ olması durumu, problemle ilgili mevcut bilginin, yani işlerin işlem süreleri ile ilgili olan sezgisel bilgi ve feromon izleri şeklinde hafızaya alınan öğrenilmiş bilginin, önem verilerek kullanılması ile ilişkilidir.

$q>q_0$ olması durumu ise, problemdeki olasılık hesabının daha da yüksek bir önemle desteklenmesi ile ilişkilidir.

q' nun değerini ayarlayarak önem vermeyi azaltma, sistemin kararlı olarak keşfetmeye bırakılması yani kendi halinde işlemesi yerine, en iyi çözümler üzerine odaklanması eylemine izin verir.

q_0, 1' e yakın olduğunda, sadece lokal olarak optimal olan çözümler seçilir. Fakat lokal olarak optimal olan çözümlerin bir kombinasyonunun da global bir optimum çözüm sonucunu doğurmayabildiği de bilinmelidir.

q_0, 0' a yakın olduğunda ise, lokal optimal çözümlere daha büyük bir ağırlık verilmesine rağmen bütün lokal çözümler incelenir. Bu yüzden prensipte, q_0' ı, 0' dan 1' e ayarlayarak, algoritmaya başlangıç aşamasında keşfetmeyi destekleme ve daha sonra da önemi verme imkanını sunarak, sistemi zamanla gelişerek ilerleyen ve iyiye giden bir şekilde dondurmak mümkündür [32].

5.2. KKS' de Global Feromon Güncelleme Kuralı

KKS' de, programın her tekrarından sonra, sadece o ana kadarki en iyi olan karıncanın feromon eklemesine izin verilir. Bu yüzden feromon güncellemesi, aşağıdaki ifadeye göre gerçekleşir:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \forall(i, j) \in T^{bs} \quad (5.4)$$

Bu ifadedeki T^{bs}, en kısa sürede işlem yapan en iyi karıncanın iş sıralamasıdır. $\Delta\tau_{ij}^{bs}$ ise, en iyi sıralamaya ait her bir ij makinesi arasına bırakılan feromon miktarıdır.

Önemle ifade edilmelidir ki, KKS' deki feromon izi güncellemesi, hem buharlaşma hem de yeni feromon bırakımı, sadece en iyi karıncanın sıralamasına uygulanır. Karınca sisteminde ise bütün sıralamalara uygulanmaktadır. Ayrıca karınca koloni sisteminde karınca sisteminden farklı olarak, bırakılan feromon miktarı ρ faktörü kadar azaltılarak bırakılır. Bu da yeni feromon değerinin, eski feromon değeri ile bırakılan feromon miktarı arasında ağırlıklandırılmış ortalama bir değere getirilmesi sonucunu verir.

5.3. KKS' de Lokal Feromon Güncelleme Kuralı

KKS' de karıncalar, tur oluşturma esnasında, global feromon izi güncellemesine ek olarak, herhangi bir (i, j) yolunu geçtikten hemen sonra lokal bir feromon güncelleme kuralını uygularlar.

$$\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0 \quad (5.5)$$

Burada ξ, $0 < \xi < 1$ aralığında olan bir parametredir ve deneysel olarak ξ için en iyi değerin 0.9 olduğu bulunmuştur. τ_0 parametresi ise, feromon izlerinin başlangıç anındaki değerleriyle aynıdır. Deneysel olarak, τ_0 için en iyi değerin $1/ (n.C^{nn})$ olduğu ortaya konulmuştur. Buradaki n, iş akış problemi uygulamasında makine sayısı, C^{nn} ise o anda işlenmekte olan işin, en kısa sürede işlenmesi için, sezgisel olarak oluşturulan tur süresidir [32]. Lokal güncelleme kuralının etkisi, her seferinde bir karıncanın kullandığı bir i, j yolundaki τ_{ij} feromon miktarının azaltılmasıdır. Böylece bu yol gelecek olan diğer karıncalar için daha az isteklilik arzedecektir. Bu şekilde henüz ziyaret edilmemiş olan yolların keşfedilme ihtimali arttırılarak uygulamada algoritmanın bir durgunluk davranışı gösterme etkisi giderilmiş olur ve

karıncaların çoğunlukla ortak bir makineye yönelmesi önlenir. Başka bir deyişle, koloni sistemindeki lokal güncelleme kuralının rolü, turları karıştırmadır. Bu şekilde bir karıncanın turunda bulunan öncelikle uğranılmış bulunan makineler, daha sonradan diğer karıncaların turlarında keşfedilebilir. Bu yüzden lokal güncelleme kuralının etkisi, yolların öğrenilmiş istekliliğinin dinamik olarak değiştirilmesidir [32].

Bir karınca aynı yolu her kullandığında, bu yol karınca için daha az isteklilik arzedecektir. Bu şekilde karıncalar feromon bilgisini daha iyi kullanmış olacaklardır. Lokal güncelleme olmasaydı bütün karıncalar önceki en iyi turun yakınlarındaki dar bir alanda araştırma yapmış olacaklardı. Oysa lokal güncelleme ile daha iyi tura yönlendirme sağlanmaktadır [7].

Diğer bir önemli nokta da, karınca sisteminde karıncalar turlarını oluştururken paralel veya ardışıl çalışma önemli değildi. Fakat koloni sisteminde, lokal feromon güncelleme kuralından dolayı bir farklılık oluşmaktadır. Gerçekleştirilen çoğu karınca koloni sistemi uygulamasında, deneysel olarak hangisinin seçiminin daha uygun olduğuyla ilgili bir kanıt bulunmamasına rağmen, genellikle bütün karıncaların paralel hareket ettirilmesi tercih edilmektedir.

5.4. KKS' de Aday Listesi Kullanımı

Karınca koloni sistemi, bir aday listesi yapısını kullanır. Bu genellikle büyük iş akış problemlerini çözmede kullanılan bir veri yapısıdır.

Aday listesi, bir işten sonra öncelikle veya en kısa sürede işlenmek istenen işlerin listesidir. Verilen herhangi bir işten, ihtimal dahilinde işlenebilecek bütün işleri incelemek yerine, öncelikle aday listesindeki işlenmemiş işler incelenir. Aday listesindeki bütün işler işlenmişse diğer işler incelenir. Aday listesindeki diğer işler cl ile gösterilir. cl algoritmada bir parametredir ve en kısa sürede işlenebilecek işler anlamına gelir. Aday listesindeki işlerin işlem süreleri, artan sıraya konulmuşlardır

ve liste düzenli olarak taranır. Aday listesi yapısına sahip KKS modeliyle çözülen iş akış problemi şu şekilde çalışmaktadır:

Bir karınca öncelikle bir sonraki iş seçimini listedekilerle sınırlar. Diğer işleri, bu listedeki işlerin tamamı işlenmişse hesaba katar. Eğer aday listesinde hala işlenmemiş iş varsa, bir sonraki j işi 5.1 ve 5.2 no' lu denklemlere göre seçilir. Aksi halde j için, henüz işlenmemiş liste dışı işlerden işlem süresi en kısa olan seçilir [32].

Yukarıda açıklanan dört farklılığa ek olarak şunlar ifade edilebilir:

- Karınca koloni sistemi uygulamalarında, feromon miktarları hiçbir zaman τ_0 başlangıç değerinin altına düşemez. Çünkü denklem 5.4. ve 5.5' deki her iki feromon güncelleme kuralı da, τ_0' a eşit veya daha büyük olan bir miktar feromonu daima eklemektedir.

- C_{bs}, en iyi turu yapan karıncanın tur uzunluğudur ve feromon izleri hiçbir zaman $1/ C^{bs}$' den daha yüksek bir değere sahip olamaz. Bu yüzden koloni sisteminde, her (i, j) yolunda şu ifade geçerlidir:

$$\forall(i, j) : \tau_0 \leq \tau_{ij} \leq (1/C^{bs}) \qquad (5.6)$$

- İş sıralama problemi örneğinde aday listesi, her i işi işlem süresi en kısa j işlerini içerir. Bu durumda aday listesi bir iş akış problemini çözmeden önce oluşturulabilir ve bütün çözüm işlemi boyunca sabit kalır. İş akış örneğinde aday listesinin kullanımının deneysel sonuçlarına göre, KKO algoritmalarının çözüm kalitesinin geliştiği gözlenmiştir.

Ayrıca algoritmada, çözüm işleminde önemli bir hızlanmaya da yol açmaktadır [7].

5.5.İş Akış Problemi İçin Yapılan Denemeler

Önerilen KKS algoritması modelinin test edilmesi amacıyla, literatürde yer alan, Êric Taillard tarafından hazırlanmış 100 adet örnek problem üzerinde 10' ar deneme yapılmıştır. Taillard tarafından oluşturulan tablolarda 20, 50, 100, 200, 500 iş ve 5, 10, 20 adet makine kullanılmıştır. Örnek problemlerin gösterimi (iş sayısı*makine sayısı) şeklinde gösterilmiştir.

Simülasyon çalışmalarında kullanılan KKS parametrelerinin değerleri Tablo 5.3' te gösterilmektedir.

Tablo 5.3: Simülasyonda kullanılan parametre seti

Karınca Sayısı	5
Tur Sayısı	100
α	1
β	10^{-4}
ρ	10^{-1}
q_0	9.10^{-1}
σ	9.10^{-1}

Tablo 5.3' te:

σ : Yerel buharlaşma katsayısıdır.

Çözülen problemlerdeki iş sayıları (n), makine sayıları (m), önerilen FRB3 algoritması ile iyileştirilmiş KKS algoritması (KKS&FRB3) ile bulunan en iyi sonuçlar, her bir problem için 10' ar çözümden sonra bulunan ortalama sonuçlar, Taillard' ın tablolarında yer alan en iyi sonuçlarla kıyaslanarak hesaplanmış ortalama sapma değeri yüzdesi (yüzde fark) yer almaktadır. Yüzde fark değeri aşağıdaki 5.7 bağıntısı ile hesaplanmıştır.

$$\text{Yüzde Fark} = \frac{C_{max} - LB}{LB} x100 \qquad (5.7)$$

LB: Taillard tablolarındaki problemlerin bugüne kadar hesaplanmış en kısa çözüm süresi.

C_{max}: İşlerin makinelerde işlendikten sonra geçen, önerilen KKS&FRB3 algoritması içinde hesaplanan toplam işlem süreleridir. C_{max} değerinin hesaplanması aşağıda örnekle açıklanmıştır:

Tablo 5.4: 3x3 şeklinde oluşturulmuş örnek problem

q	Makine1	Makine2	Makine3
İş1	3 dk	5 dk	2 dk
İş2	9 dk	6 dk	2 dk
İş3	8 dk	7 dk	5 dk

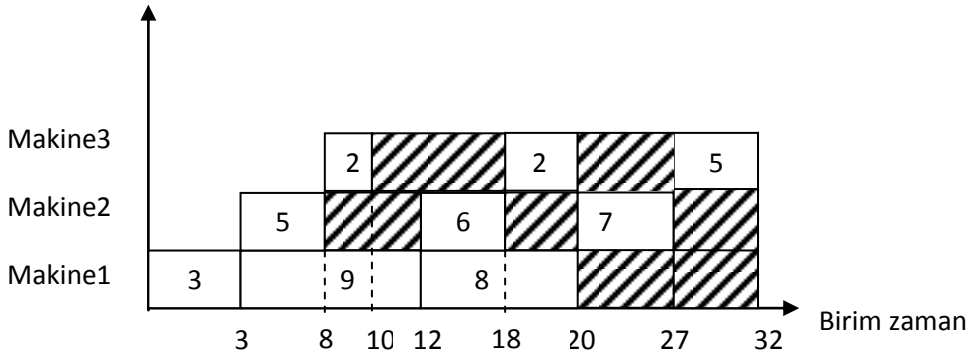
Tablo 5.4' te tek tek her işin her makinede tek başına kaç dakikada işlendiği verilmiştir.

Kod bloğunda her makine işi alırken, kendi işini ne kadar sürede işlemiş (Tablo 5.4' de hesaplanacak olan hücrenin bir üstündeki hücre, q[i-1, j]) ve alacağı işin bir önceki makinede işi kaç dakikada bitmiş (Tablo 5.4' de hesaplanacak olan hücrenin bir solundaki hücre, q[i, j-1]) kontrol edilir ve bu sürelerden büyük olanı seçilir, üzerine de makinenin o andaki işi işleme süresi eklenir. Böylece işin önceki makinelerle birlikte o makinedeki toplam işlenme süresi bulunur.

Tablo 5.5' te işler İş1, İş2, İş3 sırasında işlendiğinde makinelerden kaçıncı dakikalarda ayrıldıkları gösterilmiştir. En son Makine3' te İş3 işlendiğinde tüm işler tamamlanmış olur.

Tablo 5.5: İşlerin makinelerden ayrılma zamanları

Q	Makine1	Makine2	Makine3
İş1	3 dk	8 dk	10 dk
İş2	12 dk	18 dk	20 dk
İş3	20 dk	27 dk	32 dk



Şekil 5.2: İşlerin toplam tamamlanma sürelerını gösteren Gant şeması

Taillard' ın örnek problemlerine göre oluşturulmuş en iyi çözümlere göre hesaplanan ve iş ve makine sayısına göre değişim gösteren yüzde fark değerleri aşağıda tablolar aracılığıyla incelenmiştir. Tablolarda en düşük yüzde fark değerleri koyu renkte gösterilmiştir.

Tablo 5.6: 20 iş 5 makine (20*5) için elde edilen sonuçlar ve yüzde fark değerleri

20*5	C_{max}	LB	Yüzde Fark
Tai01	1278	1232	3,73
Tai02	1359	1290	5,35
Tai03	1081	1073	**0,75**
Tai04	1299	1268	2,44
Tai05	1236	1198	3,17
Tai06	1195	1180	1,27
Tai07	1236	1226	0,82
Tai08	1208	1170	3,25
Tai09	1238	1206	2,65
Tai10	1108	1082	2,40

20*5 için elde edilen sonuçlar arasında en düşük hata oranına sahip sonuç 20*5*Tai03 probleminde elde edilmiş olup yüzde fark = 0,75' tir. En yüksek hata oranına sahip sonuç ise 20*5*Tai02 probleminde elde edilmiş olup yüzde fark = 5,35' tir.

Tablo 5.7: 20 iş 10 makine (20*10) için elde edilen sonuçlar ve yüzde fark değerleri

20*10	C_{max}	LB	Yüzde Fark
Tai01	1583	1448	9,32
Tai02	1660	1479	12,24
Tai03	1509	1407	7,25
Tai04	1379	1308	5,43
Tai05	1422	1325	7,32
Tai06	1409	1290	9,22
Tai07	1487	1388	**7,13**
Tai08	1554	1363	14,01
Tai09	1600	1472	8,70
Tai10	1605	1356	18,36

20*10 için elde edilen sonuçlar arasında en düşük hata oranına sahip sonuç 20*10*Tai07 probleminde elde edilmiş olup yüzde fark = 7.13' tür. En yüksek hata oranına sahip sonuç ise 20*10*Tai10 probleminde elde edilmiş olup yüzde fark = 18,36' dır.

Tablo 5.8: 20 iş 20 makine (20*20) için elde edilen sonuçlar ve yüzde fark değerleri

20*20	C_{max}	LB	Yüzde Fark
Tai01	2303	1911	20,51
Tai02	2101	1711	22,79
Tai03	2332	1844	26,46
Tai04	2223	1810	22,82
Tai05	2300	1899	21,12
Tai06	2228	1875	18,83
Tai07	2278	1875	21,49
Tai08	2204	1880	17,23
Tai09	2239	1840	21,68
Tai10	2179	1900	**14,68**

20*20 için elde edilen sonuçlar arasında en düşük hata oranına sahip sonuç 20*20*Tai10 probleminde elde edilmiş olup yüzde fark = 14,68' dir. En yüksek hata oranına sahip sonuç ise 20*20*Tai04 probleminde elde edilmiş olup yüzde fark = 22,82' dir.

Tablo 5.9: 50 iş 5 makine (50*5) için elde edilen sonuçlar ve yüzde fark değerleri

50*5	C_{max}	LB	Yüzde Fark
Tai01	2724	2712	0,44
Tai02	2843	2808	1,25
Tai03	2622	2596	1,00
Tai04	2762	2740	0,80
Tai05	2863	2837	0,92
Tai06	2832	2793	1,40
Tai07	2725	2689	1,34
Tai08	2683	2667	0,60
Tai09	2561	2527	1,35
Tai10	2782	2776	**0,22**

50*5 için elde edilen sonuçlar arasında en düşük hata oranına sahip sonuç 50*5*Tai10 probleminde elde edilmiş olup yüzde fark = 0,22' dir. En yüksek hata oranına sahip sonuç ise 50*5*Tai09 probleminde elde edilmiş olup yüzde fark = 1,35' tir.

Tablo 5.10: 50 iş 10 makine (50*10) için elde edilen sonuçlar ve yüzde fark değerleri

50*10	C_{max}	LB	Yüzde Fark
Tai01	3071	2907	5,64
Tai02	2944	2821	4,36
Tai03	2916	2801	4,11
Tai04	3100	2968	4,45
Tai05	3052	2908	4,95
Tai06	3088	2941	5,00
Tai07	3165	3062	3,36
Tai08	3070	2959	3,75
Tai09	2949	2795	5,51
Tai10	3131	3047	**2,76**

50*10 için elde edilen sonuçlar arasında en düşük hata oranına sahip sonuç 50*10*Tai10 probleminde elde edilmiş olup yüzde fark = 2,76' dır. En yüksek hata oranına sahip sonuç ise 50*10*Tai01 probleminde elde edilmiş olup yüzde fark = 5,64' tür.

Tablo 5.11: 50 iş 20 makine (50*20) için elde edilen sonuçlar ve yüzde fark değerleri

50*20	C_{max}	LB	Yüzde Fark
Tai01	3968	3480	14,02
Tai02	3822	3424	11,62
Tai03	3816	3351	13,88
Tai04	3706	3336	11,09
Tai05	3420	3313	**3,23**
Tai06	3782	3460	9,31
Tai07	3834	3427	11,88
Tai08	3813	3383	12,71
Tai09	3851	3457	11,40
Tai10	3881	3438	12,89

50*20 için elde edilen sonuçlar arasında en düşük hata oranına sahip sonuç 50*20*Tai05 probleminde elde edilmiş olup yüzde fark = 3.23‘ tür. En yüksek hata oranına sahip sonuç ise 50*20*Tai01 probleminde elde edilmiş olup yüzde fark = 14,02‘ dir.

Tablo 5.12: 100 iş 5 makine (100*5) için elde edilen sonuçlar ve yüzde fark değerleri

100*5	C_{max}	LB	Yüzde Fark
Tai01	5493	5437	**1,03**
Tai02	5273	5208	1,25
Tai03	5206	5130	1,48
Tai04	5032	4963	1,39
Tai05	5255	5195	1,15
Tai06	5135	5063	1,42
Tai07	5263	5198	1,25
Tai08	5099	5038	1,21
Tai09	5455	5385	1,30
Tai10	5342	5272	1,33

100*5 için elde edilen sonuçlar arasında en düşük hata oranına sahip sonuç 100*5*Tai03 probleminde elde edilmiş olup yüzde fark = 1,03‘ tür. En yüksek hata oranına sahip sonuç ise 100*5*Tai03 probleminde elde edilmiş olup yüzde fark = 1,48‘ dir.

Tablo 5.13: 100 iş 10 makine (100*10) için elde edilen sonuçlar ve yüzde fark değerleri

100*10	C_{max}	LB	Yüzde Fark
Tai01	5807	5759	**0,83**
Tai02	5393	5345	0,90
Tai03	5682	5623	1,05
Tai04	5896	5732	2,86
Tai05	5572	5431	2,60
Tai06	5346	5246	1,91
Tai07	5660	5523	2,48
Tai08	5695	5556	2,50
Tai09	5960	5779	3,13
Tai10	5881	5830	0,87

100*10 için elde edilen sonuçlar arasında en düşük hata oranına sahip sonuç 100*10*Tai01 probleminde elde edilmiş olup yüzde fark = 0,83‘ tür. En yüksek hata oranına sahip sonuç ise 100*10*Tai09 probleminde elde edilmiş olup yüzde fark = 3,13‘ tür.

Tablo 5.14: 100 iş 20 makine (100*20) için elde edilen sonuçlar ve yüzde fark değerleri

100*20	C_{max}	LB	Yüzde Fark
Tai01	6460	5851	10,41
Tai02	6399	6099	4,92
Tai03	6459	6099	5,90
Tai04	6441	6072	6,08
Tai05	6522	6009	8,54
Tai06	6562	6144	6,80
Tai07	6493	5991	8,38
Tai08	6636	6084	9,07
Tai09	6515	5979	8,96
Tai10	6606	6298	**4,89**

100*20 için elde edilen sonuçlar arasında en düşük hata oranına sahip sonuç 100*20*Tai10 probleminde elde edilmiş olup yüzde fark = 4,89‘ dur. En yüksek hata oranına sahip sonuç ise 100*20*Tai01 probleminde elde edilmiş olup yüzde fark = 10,41‘ dir.

Tablo 5.15: 200 iş 10 makine (200*10) için elde edilen sonuçlar ve yüzde fark değerleri

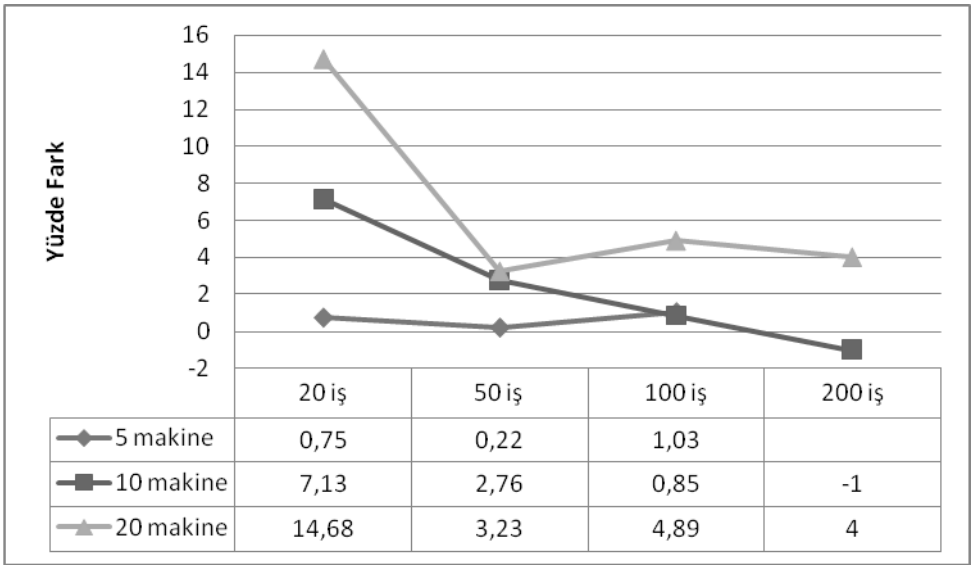
200*10	C_{max}	LB	Yüzde Fark
Tai01	10957	10816	1,30
Tai02	10611	10422	1,81
Tai03	11026	10886	1,29
Tai04	11057	10794	2,44
Tai05	10615	10437	1,71
Tai06	10400	10255	1,41
Tai07	10912	10761	1,40
Tai08	10834	10663	1,60
Tai09	10912	10348	5,45
Tai10	10510	10616	**- 1,00**

200*10 için elde edilen sonuçlar arasında en düşük hata oranına sahip sonuç 200*10*Tai10 probleminde elde edilmiş olup yüzde fark = -1,00‘ dır. En yüksek hata oranına sahip sonuç ise 200*10*Tai09 probleminde elde edilmiş olup yüzde fark = 5,45’ tir.

Tablo 5.16: 200 iş 20 makine (200*20) için elde edilen sonuçlar ve yüzde fark değerleri

200*20	C_{max}	LB	Yüzde Fark
Tai01	11545	10979	5,16
Tai02	11576	10947	5,75
Tai03	11652	11150	4,50
Tai04	11606	11127	4,30
Tai05	11590	11132	4,11
Tai06	11528	11085	**4,00**
Tai07	11725	11194	4,74
Tai08	11667	11126	4,86
Tai09	11574	10965	5,55
Tai10	11681	11122	5,03

200*20 için elde edilen sonuçlar arasında en düşük hata oranına sahip sonuç 200*20*Tai06 probleminde elde edilmiş olup yüzde fark = 4,00‘ dır. En yüksek hata oranına sahip sonuç ise 200*20*Tai02 probleminde elde edilmiş olup yüzde fark = 5,75‘ tir.



	20 iş	50 iş	100 iş	200 iş
5 makine	0,75	0,22	1,03	
10 makine	7,13	2,76	0,85	-1
20 makine	14,68	3,23	4,89	4

Şekil 5.3: İş ve makine sayısına göre KKS&FRB3 ile hesaplanan en düşük yüzde fark değerlerinin grafiği

Başarı oranı ile yüzde fark değeri ile ters orantılıdır. Diğer bir deyişle yüzde fark değerinin azalması, daha başarılı sonuçlar elde edildiği anlamına gelir. Şekil 5.3' teki grafikte 200 iş 5 makine için yüzde fark değeri boş bırakılmıştır. Bunun nedeni Taillard tablolarında 200*5 için örnek problem bulunmamasıdır.

Şekil 5.3' teki grafiği incelediğimizde elde ettiğimiz sonuçlar aşağıdaki gibidir:

- En yüksek başarı 200 iş 10 makine örnek probleminin çözümünde elde edilmiştir. Buradaki yüzde fark değerinin negatif (-1) olması Taillard tablolarındaki en kısa süreli iş sıralamadan daha kısa sürede işlerin tamamlandığını göstermektedir. Tablo 5.15' de Tai10 satırı incelendiğinde önerilen KKS&FRB3 algoritması ile yapılan çözümde işlerin toplam 10510 birim sürede tamamlandığı, Taillard tablolarında kayıtlı en iyi bitirme süresinin ise 10616 birim olduğu gözlenmiştir.

- En düşük yüzde fark değerleri içinde, en düşük başarı 20 iş 20 makine örnek probleminin çözümünde elde edilmiştir. Buradaki yüzde fark oranı 14,68' dir.

- 50 *10 örnek probleminde elde edilen yüzde fark oranı (2,76), 50*20 örnek probleminde elde edilen yüzde fark oranına (3,23) yakınlık göstermekte ve her ikisi de 50*5 örnek probleminde elde edilen yüzde fark oranından (0,22) daha büyük bir

oran vermektedir. Bu durum sonucunda 50 adet iş için makine değeri arttıkça algoritmanın başarım oranı düşmüştür.

- 20 adet iş için, makine sayıları 5,10 ve 20 için yüzde fark oranları sırasıyla 0.75, 7.13, 14.68' dir. Buradan çıkan sonuç 20 adet iş için makine sayısı arttıkça algoritmanın başarım oranı düşmektedir

- Grafik genel olarak incelendiğinde en yüksek yüzde fark diğer bir deyişle en düşük başarım 20 makine için oluşturulan problemlerin çözümünde görülmüştür. En yüksek başarım ise 5 makine ile çözülen problemlerde elde edilmiştir.

- Grafikten elde edilen bir diğer sonuç ise iş sayılarının artması veya azalması tek başına sonucu etkilememiştir. Çünkü aynı adet makine ile yapılan çalışmalarda iş sayıları arttığında, yüzde fark değerinde tutarlı bir oranda artma veya azalma olmamıştır.

Bugüne kadar yapılmış bir çok çalışmada, elde edilen sonuçları karşılaştırmak amacıyla hata değerinden yararlanıldığından, çalışmamızda KKS&FRB3 algoritması ile elde edilen sonuçların hata değerleri hesaplanmış ve diğer Karınca Algoritmaları ile karşılaştırılmıştır. Karşılaştırma sonucu elde elden sonuçlar Tablo 5.17' de gösterilmiştir.

$$\text{Hata} = \frac{C_{max} - UB}{UB} x100 \tag{5.8}$$

UB: Taillard tablolarındaki problemlerin optimum çözüm süresi.

Tablo 5.17: Önerilen KKS & FRB3 Algoritmasının farklı algoritmalarla karşılaştırılması

	KKS & FRB3	KKS	Palmer	GA	TB	KDS	KKA	MMAS	PACO
n/m	Hata Oranı	Hata Oranı	Hata Oranı	Hata Oranı	Hata Oranı	Hata Oranı	Hata Oranı	Hata Oranı	Hata Oranı
20/5	**0,10**	1,19	10,81	1,61	1,27	1,46	0,18	0,41	0,70
20/10	**0,47**	1,70	15,27	2,29	1,71	2,02	0,79	0,59	0,84
20/20	0,16	1,60	16,34	1,95	0,86	**1,10**	0,85	0,41	0,72
50/5	0,12	0,43	5,23	0,45	0,78	0,79	**0,06**	0,14	0,09
50/10	1,78	1,89	13,48	2,28	1,98	3,21	1,81	2,19	**0,75**
50/20	**1,55**	2,71	15,46	3,44	2,86	3,90	2,90	2,47	1,85
100/5	0,15	0,22	2,38	0,23	0,56	0,76	**0,05**	0,19	0,07
100/10	1,05	2,22	9,08	1,25	1,33	2,69	1,35	0,93	**0,40**
100/20	2,31	**0,64**	13,24	2,91	2,32	3,98	2,68	2,23	0,99
200/10	1,04	1,30	4,75	**0,50**	0,83	3,81	------	----	-----
200/20	2,12	1,68	6,72	**1,35**	1,74	6,07	------	----	-----
Ortalama	0,99	1,42	10,25	1,66	1,48	2,71	1,19	1,06	**0,71**

Koyu renkle gösterilmiş alanlar hata oranı en düşük değerleri göstermektedir.

Palmer: 1965 yılında Palmer tarafından geliştirilmiş ve iş akış problemlerine uygulanmış metot.

GA: Genetik algoritma [2].

TB: Tavlama benzetimi [2].

KDS: Komşu Düğüm Seçimi [3].

KKA: Karınca Kolonileri Algoritması [3].

MMAS: Min-Max karınca sistemi [1].

PACO: Paralel Karınca Sistemi [1].

Sonuçlar incelendiğinde KKS algoritmasının FRB3 algoritması ile iyileştirilmesi sonucunda klasik KKS algoritması ve KKA' ya göre genel olarak daha yüksek başarı elde ettiği gözlenmiştir. Programın çalışma süreleri iş ve makine sayılarına bağlı olarak arttığından karınca ve tur sayıları düşük tutulmaya çalışılmıştır. Popülasyon sayısı ve tur sayıları arttırılarak hata oranı daha düşük olan sonuçlar elde edilebileceği düşünülmektedir. Ayrıca KKS algoritmasının genel yapısını değiştirmeden ilkleme için ve bulunan sıralama sonuçlarını iyileştirmek için daha yüksek başarı oranına sahip metotlar da denenebilir.

Sonuç olarak birden fazla bilimsel yöntemin birleştirilerek, her birinin olumlu yönleri ön plana çıkartılarak, daha yüksek başarı elde edilebileceği gözlenmiştir. Bu tezde KKS ve FRB3 algoritmalarının birleştirilmesi ile elde edilen yöntemin eksik yönlerinin, ileride başka yöntemlerin kullanılması ile iyileştirilebileceği düşünülmektedir.

6. SONUÇLAR ve ÖNERİLER

Yapılan yüksek lisans tez çalışmasında, bir çok kombinasyonel optimizasyon problemlerinde iyi sonuçlar veren bir meta-sezgisel teknik olan karınca kolonileri algoritmaları incelenmiştir. Karınca kolonileri algoritmaları, yapay sinir ağları ve genetik algoritmalara benzer şekilde, bazı doğal sistemlerin özet modelleridir ve bir çok alanda uygulaması bulunmaktadır.

Çalışmada öncelikle, KKA' nın nasıl ortaya çıktığı ve doğal karıncaların hangi özelliklerinden faydalanıldığı belirtilmiştir. Daha sonra KKA' nın bazı eksikliklerinin giderilmesi amacıyla oluşturulan diğer KKA çeşitleri incelenmiştir. İncelenen algoritmalar arasında, KKA' nın bir açılımı olan KKS algoritması incelenmiş ve konunun daha iyi anlaşılması amacıyla KKS algoritmasının GSP' nin çözümüne uyarlanması ele alınmıştır.

KKS' nin bazı eksiklikleri analiz edilerek, performansını arttırmak amacı ile KKS algoritması, FRB3 algoritması ile iyileştirilmiş ve yeni bir yöntem önerilmiştir. Önerilen bu yönteme KKS & FRB3 adı verilmiştir.

Uygulama alanı olarak PTİA problemleri incelenmiş, problemin çözümü için, geliştirilen yeni algoritma kullanılmıştır. Çalışılan problemler NP-zor tipte problemlerdir. Bu problemleri optimal şekilde çözmek için geliştirilmiş bir yöntem bulunmamakta, daha çok sezgisel yöntemler kullanılmaktadır. Yapılan tez çalışmasında önerilen KKS&FRB3 de tüm çözüm uzayı içinde iyi çözümler arama mantığıyla çalışan sezgisel bir yöntemdir.

Geliştirilen yeni algoritma ve karınca sistemi algoritması ile literatürde kullanılan kıyaslama problemleri incelenmiştir. Daha öncesinde, algoritmaların çözüm performanslarının geliştirilmesi amacıyla parametre optimizasyonu çalışması yapılmıştır. Algoritma; her problem için bulunan en iyi parametrelerle çalıştırılmıştır. Çözülen PTİA problemlerinde, geliştirilen yeni algoritmanın, klasik KKS algoritmasından daha iyi sonuçlar verdiği görülmüştür.

Daha geniş bir ifade ile, literatürde test problemleri olarak kullanılan değişik boyutlardaki toplam 100 adet Taillard Problemi 10' ar kere çözülmüştür. Yapılan çözümlerde parametre optimizasyonu sonucu elde edilen parametre setleri kullanılmıştır.

Önerilen KKS&FRB3 algoritması ile en iyi sonuç 200 iş 10 makine örnek problemlerinden Tai10 probleminin çözümünde elde edilmiştir. Buradaki yüzde fark değeri -1' dir. En düşük başarı ise 20 iş 20 makine örnek problemlerinden Tai03 probleminin çözümünde elde edilmiştir. Buradaki yüzde fark oranı 26.46' dır.

Çözülen tüm problemler için geliştirilen yeni algoritma klasik KKS algoritmasına göre %43, Palmer algoritmasına göre %9.26, GA' ya göre %67, TB' ye göre %49, KDS' ye göre %172, KKA' ya göre %20, MMAS' ye göre %7 oranında başarı göstermiştir. PACO algoritması ise önerilen KKS&FRB3 algoritmasına göre %28 oranında daha iyi sonuçlar üretmiştir. Burada problemleri aynı iş ve aynı makine adedine sahip olanlar şeklinde gruplandırdığımızda, elimizde 10 adet ortalama değeri hesaplanmış sonuç bulunmaktadır. Bu ortalama değerler arasında önerilen KKS&FRB3 algoritması, örneklerin 3 tanesi için alt sınır değerini verirken; KKS algoritması, GA, KDS ve PACO sırasıyla 1, 2, 2, 2 örnek için alt sınır değerini vermiştir. Palmer, TB, MMAS ise hiç bir örnek için alt sınır değeri verememiştir. Bu sonuçlara göre geliştirilen yeni algoritmanın PTİA problemi için çözüm performansının kıyaslanan diğer yöntemlerden daha iyi olduğu; fakat optimal değer bulma konusunda sıkıntıları olduğu sonuçlarına varılmıştır.

Çalışmada gerçek hayatta karşılaşılabilecek bir problem türü olan PTİA problemi ele alınmıştır. PTİA uygulamaları sıkça görülmektedir. Gerçek hayattaki PTİA sistemlerinde genellikle kademelerdeki makineler eş değildir, aynı işlemleri yapabilen fakat farklı işlem sürelerine sahip makinelerdir. Bundan sonraki çalışma olarak, gerçek fabrika ortamındaki bir PTİA problemi çözmek için algoritma değiştirilebilir.

Geliştirilen yeni algoritmada olasılık belirleme ve seçim süreçlerinin iyileştirilmesi veya başka tekniklerle birleştirilmesi ile daha kaliteli çözüm veren algoritmalar elde edilebilir.

Sonuçta bu çalışma ile, klasik karınca kolonileri algoritmasının, FRB3 ile birleştirilerek daha başarılı sonuçlar elde ettiği gözlenmiştir.

KAYNAKLAR

[1] Rajendran, C., Ziegler, A., "Ant-Colony Algorithms for Permutation Flowshop Scheduling to Minimize Makespan/ Total Flow Time of Jobs", ***European Journal of Operationl Research,*** 155, 433, (2004).

[2] Ying, K.C., Liao, C.J., "An Ant Colony Sysytem for Permutation Flowshop Sequencing*",* ***Computers and Operations Research***, 31, 791, (2004).

[3] Ahmadizar, F., Barzinpour, F., Yolat, J., "Solving Permutation Flow Shop Sequencing Using Ant Colony Optimization", ***IEEE Trans.on Evolu. Comp.***, 1448, 753, (2008).

[4] Biroğul, S. "Genetik Algoritma Yaklaşımıyla Atölye Çizelgeleme", Yüksek Lisans Tezi, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 1, 2, 5, 6, 7, 9, 11, 12,21 (2005).

[5] Eren, T., Güner, E., "Tek ve Paralel Makineli Problemlerde Çok Ölçütlü Çizelgeleme Problemleri İçin Bir Literatür Taraması", ***Gazi Üniversitesi Mühendislik Fakültesi Dergisi***, 17(4), 37-69, (2002).

[6] Baskak, M., Erol, V., "Sipariş Tipi Atölyelerde İş Akış Problemi İçin Bir Genetik Algoritma Uygulaması", ***Yöneylem Araştırması Endüstri Mühendisliği XXIV. Ulusal Kongresi,*** 12 Gaziantep, 15-18 Haziran (2004).

[7] Alaykıran, K., "Kombinatoriyal Optimizasyon Problemlerinin Karınca Algoritmaları İle Çözümü", Yüksek Lisans Tezi, ***Selçuk Üniversitesi Fen Bilimleri Enstitüsü***, Konya, 5-10, (2004).

[8] Bircan, H., Kartal, Z., "Doğrusal Programlama Tekniği ile Kapasite Planlaması Yaklaşımı ve Çimento İşletmesinde Bir Uygulaması", ***Cumhuriyet Ünversitesi İktisadi ve İdari Bilimler Dergisi***, 5, 133-137, (2002).

[9] Engin, O., Fığlalı, A., "Akış Tipi Çizelgeleme Problemlerinin Genetik Algoritma Yardımı İle Çözümünde Uygun Çaprazlama Operatörünün Belirlenmesi", ***Doğuş Üniversitesi Dergisi***, 8 (1), 27-35, (2002).

[10] Türkbey, O., Alabaş, Ç. "Tesis Yerleşim Problemi İçin Bir Bulanık-Tabu Arama Yaklaşımı", ***Anadolu Üniversitesi Dergisi***, 3(1), 77-88, (2002).

[11] Şahin, R., "Dinamik Tesis Düzenleme Problemi için Bir Tavlama Benzetimi Sezgiseli", ***Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi***, 23, 863-870, (2008).

[12] Dorigo, M., Gambardella, L.M., "Ant Colonies For The Traveling Salesman Problem" ***BioSystems***, 43(2), 73–81, (1997).

[13] Özel, S., "Üretim Çizelgeleme Algoritmalarının Programlanması", Yüksek Lisans Tezi, ***Uludağ Üniversitesi Fen Bilimleri Enstitüsü***, Bursa, 5-9, 55-56, (1999).

[14] Selam, Ü., "Bilgisayar Destekli Atelye Tipi Üretim Çizelgeleme ve Bir Kalıp Atelyesinde Uygulanması", Yüksek Lisans Tezi, ***İstanbul Üniversitesi Sosyal Bilimler Enstitüsü***, İstanbul, 9-18, (1996).

[15] Dornorf, U., Pesh, E., " Evolution Based Learning in a Job Shop Scheduling Environment", ***Computers and Operations Research***, 22, 25-40, (1995).

[16] Jain, N., Bagchi, T. P., "Flowshop Scheduling by Hybridized GA: Some New Results", ***Int'l Journal of Industrial Engineering***, 7 (3), 213-223, (2000).

[17] Engin, O., "Akış Tipi Çizelgeleme Problemlerinin Genetik Algoritma İle Çözüm Performansının Artırılmasında Parametre Optimizasyonu", Doktora Tezi, ***İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, 4, 9-36, (2001).

[18] Conway, R., Maxwell, W., McClain, J.O., Thomas, L.J. "The Role of Work-in-Process Inventory in Serial Production Lines", ***Operations Research***, 36 (2), 229-241, (1988).

[19] Gen, M., Tsujımura, Y., Kubota, E., "Solving Job-Shop Scheduling Problems by Genetic Algorithm", ***IEEE International Conferance on, 199411***, 1577-1582, (1994).

[20] Temiz, İ., "Seri İş Akışlı Çizelgeleme İçin Bulanık Dal-Sınır Algoritması", Yüksek Lisans Tezi, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü,*** Ankara, 21, (1998).

[21] Cheng, T.C.E., Lin, B.M.T., "Johnson's Rule, Composite Jobs and the Relocation Problem", ***European Journal of Operational Research***, 192 (3), 1008-1013, (2009).

[22] Erdem, A., "Bakım Kısıtı Altında Genetik Algoritmalarla Üretim Çizelgeleme" Yüksek Lisans Tezi, ***Sakarya Üniversitesi Fen Bilimleri Enstitüsü***, 30, (2008).

[23] Düğenci, M., "Genetik Algoritmalarla Permütasyon Tipi İş Akış", Yüksek Lisans Tezi, ***Sakarya Üniversitesi Fen Bilimleri Enstitüsü***, Sakarya, 3-4, 21, 35-39, (1996).

[24] Campbell, H.G., Dudek, R.A., Smith, B.L.," A Heuristic Algorithm for the n-Job, m-Machine Sequencing Problem", ***Management Science,*** 16, 16, (1970).

[25] Chen, C.L., Vempati, V.S., Aljaber, N., "An Application of Genetic Algorithms for Flow Shop Problems", ***European Journal of Operation Research,*** 80, 389-396, (1995).

[26] Varela, M.L.R., Aparício, J.N., Silva, S.C., "A Web-based Application for Manufacturing Scheduling", ***A Scientific and Technical Publishing Company Acta Press***, 38, 100, (2003).

[27] Glover, F., Laguna, M., "Tabu Search", ***Kluwer Academic Publishers***, 48 (4), (1997).

[28] Glover, F., "Tabu Search - Part II", ***ORSA Journal on Computing***, 4, 32, (1990).

[29] Kirkpatrick, S., Gelatt, C.D., Vecchi, M. P., "Optimization by Simulated Annealing", ***Science***, 220, 671–680, (1983).

[30] Goldberg, D.E., "Genetic Algorithms in Search Optimization and Machine Learning", ***Addison-Wesley Publishing Company***, 15(3), 621-638, (1989).

[31] Rad, S.F., Ruben, R., Boroojerdian, N., "New High Performing Heuristics For Minimizing Makespan In Permutation Flowshop", ***OMEGA, The International Journal of Management Science***, 37(2), 3, (2007).

[32] Nawaz, M., Enscore, E.E., Ham, I., "A Heuristic Algorithm For The M-Machine, N-Job Flow-Shop Sequencing Problem", ***OMEGA, The International Journal of Management Science,*** 11(1), 91–5, (1983).

[32] Öztürk, C., "Karınca ve Sürü Optimizasyon Yöntemlerinin İncelenmesi ve Yazılım Uygulamalarının Oluşturulması", Yüksek Lisans Tezi, ***Marmara Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, 5, (2006).

[33] Gaubert, L., Redou, P., Harrauet, F., Tisseau, J., "First Mathematical Model Of Brood Sorting By Ants: Functional Self-Organization Without Swarm-Intelligence", ***Ecological Complexity***, 4 (4), 234-241, (2007).

[34] Cao, Y.U., Fukunaga, A.S., Kahng, A.B., "Cooperative Mobile Robotics: Antecedents and Directions", ***Iros***, 1, 7-27, (1997).

[35] Kennedy, J., Eberhart, R.C.,''Swarm Intelligence'', ***Morgan Kaufmann Publishers***, 47, (2001).

[36] Wordnet Sözlük, ***Princeton Üniversitesi***, (2005).

[37] Dorigo, M., Caro, G., "Ant Algorithms for Discrete Optimization," ***Universite Libre de Bruxelles Brussels***, Belgium, 5(2), 25, (2005).

[38] Shang, Y., "Global Search Methods for Solving Nonlinear Optimization Problems", ***University of Illinois at Urbana Champaign***, Illinois, 37-39, (1997).

[39] Pohlheim, H., 2007, The Genetic and Evolutionary Algorithm Toolbox [online], http://www.software.com/downloads/educationmathematics/review-GEATbx-3.x-Genetic-and-Evolutionary-Algorithm-Toolbox-academic-unive-938622.html, (**Ziyaret Tarihi:** 14 Aralık 2008).

[40] Bakır, M.A., Altunkaynak, B., "Tamsayılı Programlama, Teori Modeller ve Algoritmalar", ***Nobel Yayın Dağıtım***, Ankara, 112, (2003).

[41] Bonabeau, E., Dorigo, M., Theraulaz, G., "Inspiration for Optimization From Social Insect Behaviour", ***Nature Dergisi***, 406, 39-42 (2000).

[42] Dorigo, M., "Optimization, Learning and Natural Algorithms", ***Dipartimento di Elettronica, Milan,*** 38, 85, (1992).

[43] Gambardella, L.M., Dorigo, M., "Solving symmetric and asymmetric TSPs by ant colonies", ***IEEE Conference on Evolutionary Computation, IEEE Press,*** 51, 622-627, (1996).

[44] Dorigo, M., Maniezzo, V., Colorni, A., "The Ant System: An Autocatalytic Optimizing Process", ***Technical Report,*** 91-016, Italy, 17, (1991).

[45] Dorigo, M., Stützle, T. "Ant Colony Optimization", ***MIT Press***, 25, 72, 77, 79, 107, (2004).

[46] Piscataway, N.J., "Intelligent Systems and Signal Processing", ***International Conference on Robotics***, Joint Center for Intelligent Sensing and Systems, 17-18 Changsha, China, 8-13 Ekim, (2003).

[47] Zhang, L.G., Liu, Z., "Robotics, Intelligent Systems and Signal Processing", ***Chinese Journal of Oncology***, 23(3), 193, Beijing, (2003).

[48] Brahimi, N., Dauzere-Peres, S., Najid, N., Nordli, M.A., "Single Item Lot Sizing Problem", ***European Journal of Operational Research,*** 168, 1-16, (2006).

[49] Kalınlı, A., "Aktif Filtreler İçin Devre Elemanı Değerlerin Karınca Koloni Algoritması Kullanılarak Seçimi", Yüksek Lisans Tezi, ***Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü***, İzmir, 45-46, (2003).

[50] Wodrich, M., "Ant Colony Optimization,", Yüksek Lisans Tezi, ***Cape Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü***, Güney Afrika, 27,42, (1996).

[51] Bilchev, G., Parmee, I., "Constrained Optimization with Ant Colony Search Model", ***Plymouth Mühendislik Merkezi, Plymouth Üniversitesi,*** İngiltere, 18, (1996).

[52] Hiroyasu, T., Miki, M., Ono, Y., Minami Y., "Ant Colony for Continuous Functions", ***Bilim Mühendisliği, Doshisha Üniversitesi***, 127, (2000).

[53] Spaulding, K.A., "Natural Metaphoric Optimization Algorithms", Yüksek Lisans Tezi, ***Texas Üniversitesi,*** Austin, 46-49, (1998).

[54] Kuhn, L., "Ant Colony Optimization for Continuous Spaces", Yüksek Lisans Tezi, ***Queensland Üniversitesi, Yazılım Mühendisliği***, Queensland, 16, 33, (2002).

[55] Kalınlı, A., Karaboğa, D., "Training Recun-ent Neural Networks by Using Parallel Tabu Search Algorithm based on Crossover Operation", ***Engineering Applications of Artificial Inteligence***, 17(5), 529-542, (2004).

[56] Sarıkoç, F., "Paralel Karınca Kolonisi Optimizasyon Algoritması ve Test Problemlerinden Performansının İncelenmesi", Yüksek Lisans Tezi, ***Erciyes Üniversitesi Fen Bilimleri Enstitüsü***, Kayseri, 35-40, (2004).

[57] Timor, M., "Yöneylem Araştırması ve İşletmecilik Uygulamaları", ***İ.Ü.İşletme Fakültesi Yayınları,*** İstanbul, 48, 148-149, (2001).

[58] Tsai C.F., Tsai C.W., Tseng C.C., "A new hybrid heuristic approach for solving large traveling salesman problem", ***Information Sciences***, 48, 1-15, (2003).

[59] Stützle, T., Dorigo, M., "ACO Algorithms for the Traveling Salesman Problem", ***Evolutionary Algorithms in Engineering and Computer Science***, Wiley, 116, (1999).

[60] Dorigo, M., Di Caro, G., Gambardella, L.M., "Ant Algorithms for Discrete Optimization", ***Artificial Life***, 137-172, (1999).

[61] Dorigo, M., L. M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", ***IEEE Transactions on Evolutionary Computation,*** 1(1), 53-66, (1997).

[62] Söyler, H., Keskintürk, T., "Karınca Kolonisi Algoritması ile Gezen Satıcı Probleminin Çözümü", ***8. Türkiye ekonometri ve İstatistik Kongresi,*** İnönü Üniversitesi, 7-10 Malatya, 24-25 Mayıs (2007).

[63] Mark C., Rothwell, E.J., Ross, J.E**.,** "Investigation of Simulated Annealing, Ant-Colony Optimization, And Genetic Algorithms For Self-Structuring Antennas" ***IEEE Transactions On Antennas And Propagation***, 52(4), 1007-1014, (2004).

ÖZGEÇMİŞ

1983 yılında İzmit' te doğdu. İlk ve orta öğrenimini İzmit' te tamamladı. 2001 yılında girdiği Gazi Üniversitesi, Teknik Eğitim Fakültesi, Bilişim Sistemleri Öğretmenliği Bölümü' nden 2006 yılında Bilişim Teknolojileri Öğretmeni olarak mezun oldu. 2006 yılından beri Pendik Endüstri Meslek Lisesi, Endüstriyel Otomasyon Teknolojileri Bölümünde, Bilişim Teknolojileri Öğretmeni olarak görev yapmakta olup evlidir.