STOKASTİK U-TİPİ MONTAJ HATTI DENGELEME PROBLEMLERİ İÇİN PARÇACIK SÜRÜ OPTİMİZASYONU

Özkan BALİ

DOKTORA TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAZİRAN 2009
ANKARA

Özkan BALİ tarafından hazırlanan STOKASTİK U-TİPİ MONTAJ HATTI DENGELEME PROBLEMLERİ İÇİN PARÇACIK SÜRÜ OPTİMİZASYONU adlı bu tezin Doktora tezi olarak uygun olduğunu onaylarım.

Prof. Dr. Cevriye GENCER

.....

Tez Danışmanı, Endüstri Müh. Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği / oy çokluğu ile Endüstri Mühendisliği Anabilim Dalında Doktora tezi olarak kabul edilmiştir.

Prof. Dr. Bilal TOKLU

.....

Endüstri Müh. Anabilim Dalı, Gazi Üniversitesi

Prof. Dr. Cevriye GENCER

.....

Endüstri Müh. Anabilim Dalı, Gazi Üniversitesi

Prof. Dr. Hadi GÖKÇEN

.....

Endüstri Müh. Anabilim Dalı, Gazi Üniversitesi

Doç. Dr. Mehmet Ali AKÇAYOL

.....

Bilgisayar Müh. Anabilim Dalı, Gazi Üniversitesi

Doç. Dr. Burak BİRGÖREN

.....

Endüstri Müh. Anabilim Dalı, Kırıkkale Üniversitesi

Tarih: 04 / 06 / 2009

Bu tez ile G.Ü. Fen Bilimleri Enstitüsü Yönetim Kurulu Doktora derecesini onamıştır.

Prof. Dr. Nail ÜNSAL

Fen Bilimleri Enstitüsü Müdürü

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Özkan BALİ

STOKASTİK U-TİPİ MONTAJ HATTI DENGELEME PROBLEMLERİ İÇİN PARÇACIK SÜRÜ OPTİMİZASYONU

(Doktora Tezi)

Özkan BALİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Haziran 2009

ÖZET

Bu çalışmada, stokastik görev zamanlı U-tipi hat dengeleme problemleri (SUMHDP–1) için bir meta-sezgisel olan parçacık sürü optimizasyonu (PSO) algoritması önerilmektedir. PSO, kuş ve balık sürülerinin yerini bilmedikleri yiyeceği aramaları için gösterdikleri davranışların benzetiminden yararlanılarak ortaya atılmış bir meta-sezgiseldir. Bu yöntemdeki her bir parçacık, bir çözümü ifade eder. Önerilen PSO algoritması, literatürde yer alan 45 göreve kadar standart montaj hattı dengeleme problemlerine uygulanmış ve SUMHDP–1 problemini ele alan çalışmalarla sonuçlar karşılaştırılmıştır. Sonuçlar göstermektedir ki, önerilen algoritma, literatürdeki algoritmalardan, daha kısa CPU zamanlarında ya aynı yada daha iyi sonuçlar vermektedir. Ayrıca, önerilen algoritma büyük boyutlu problemler için de çalıştırılmış ve sonuçlar incelenmiştir.

Bilim Kodu : **906.1.141**
Anahtar Kelimeler : **Stokastik U-tipi hat dengeleme, parçacık sürü optimizasyonu**
Sayfa Adedi : **120**
Tez Yöneticisi : **Prof. Dr. Cevriye GENCER**

PARTICLE SWARM OPTIMIZATION FOR
STOCHASTIC U-TYPE ASSEMBLY LINE BALANCING PROBLEMS
(Ph D. Thesis)

Özkan BALİ

GAZI UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
June 2009

ABSTRACT

In this study, a particle swarm optimization (PSO) algorithm is proposed for the U-type line balancing problem with stochastic task times (SUALBP–1). PSO is a meta-heuristic inspired by simulating the behavior of bird flocks and fish schools when searching for food. In this method, each particle represents a solution. The proposed PSO algorithm has been applied to standard assembly line balancing problems up to 45 tasks from the literature and compared with the results of similar SUALBP–1 studies. Comparisons show that the solution of the proposed algorithm is as good as or better than those of other algorithms in the literature; also they are obtained in less CPU time. Furthermore, the proposed algorithm has been also run for large size problems and the results have been analyzed.

Science Code : **906.1.141**
Key Words : **Stochastic U-type line balancing, particle swarm optimization**
Page Number : **120**
Adviser : **Prof. Dr. Cevriye GENCER**

TEŞEKKÜR

Çalışmalarım süresince değerli yardım ve katkılarından dolayı tez hocam Sayın Prof.Dr. Cevriye GENCER'e, tezim süresince bilgi birikimi ve tecrübelerinden faydalandığım Sayın Prof.Dr. Hadi GÖKÇEN'e, manevi desteklerini her zaman yanımda hissettiğim annem Zehra BALİ'ye ve babam Ahmet BALİ'ye, sevgili eşim Hatice BALİ'ye ve biricik oğlum Erkan'a teşekkürü bir borç bilirim.

İÇİNDEKİLER

	Sayfa
ÖZET…….....	iv
ABSTRACT……..	v
TEŞEKKÜR……..	vi
İÇİNDEKİLER……..	vii
ÇİZELGELERİN LİSTESİ……...	ix
ŞEKİLLERİN LİSTESİ……...	x
SİMGELER VE KISALTMALAR…….....	xii
1. GİRİŞ……	1
2. MONTAJ HATTI DENGELEME ..…….....	4
2.1. Üretim Sistemleri ve Çeşitleri …….…….....	4
2.2. Montaj Hattı Dengeleme Problemi……..……....	7
2.2.1. MHD ile ilgili terimler ...…….....	9
2.2.2. Geleneksel düz hat MHD ……...	12
2.2.3. U-tipi MHD ……	15
2.2.4. Paralel MHD.....	19
2.3. MHD Literatür Araştırması…….....	20
2.3.1. Geleneksel düz hat MHD ile ilgili literatür.....	20
2.3.2. U-tipi MHD ile ilgili literatür.....	21
2.3.3. Paralel MHD ile ilgili literatür.....	23
3. PARÇACIK SÜRÜ OPTİMİZASYONU …….…….....	24
3.1. PSO'nun Tanımı .....……....	24
3.2. PSO'nun Genel Yapısı……...	25
3.3. PSO Terimleri……	26
3.4. PSO'nun Genel Mekanızması……...	30
3.4.1. PSO ile sürekli optimizasyon.....	31
3.4.2. PSO ile kesikli optimizasyon.....	32
3.5. PSO İçin Literatür Araştırması……..	33

Sayfa

4. PSO İLE STOKASTİK U-TİPİ HAT DENGELEME ……..... 38
4.1. Stokastik U-tipi Hat Dengeleme Problemi ……... 38
4.2. SUALBP-1 İçin PSO Algoritması..... 40
4.2.1. Tanımlamalar……... 42
4.2.2. Başlangıç çözümünün bulunması…… 42
4.2.3. PSO'nun işletilmesi……. 45
4.3. Sayısal Örnek……. 49
4.4. Hesaplama Sonuçları……. 53
4.4.1. Karşılaştırmalı sonuçlar……... 54
4.4.2. Genel hesaplama sonuçları…….. 61
4.5. PSO Parametrelerinin Optimal Çözüm Üzerindeki Etkisinin İncelenmesi…… 69
4.5.1. PSO parametrelerinin optimal çözüm üzerindeki ana etkileri……. 70
4.5.2. PSO parametrelerinin optimal çözüm üzerindeki ikili etkileri…… 73
5. SONUÇ……. 79
KAYNAKLAR…….. 82
EKLER…….. 91
EK-1 SUMHDP-1 için PSO algoritması delphi kodu……... 92
EK-2 Düşük varyans için test problemleri sonuçları…… 112
EK-3 Yüksek varyans için test problemleri sonuçları……... 116
ÖZGEÇMİŞ……... 120

ÇİZELGELERİN LİSTESİ

Çizelge	Sayfa
Çizelge 2.1. Öncelik matrisi……..	11
Çizelge 2.2. Basit montaj hattı dengeleme problem tipleri……...	14
Çizelge 4.1. Bowman problemi için örnek görev süreleri...…….	49
Çizelge 4.2. Bowman problemi için örnek bir atama……	50
Çizelge 4.3. Başlangıç çözümü pozisyon vektörü ve yerel en iyi…….	50
Çizelge 4.4. Başlangıç çözümü global en iyi……	51
Çizelge 4.5. Başlangıç hız değerleri……..	51
Çizelge 4.6. Birinci iterasyon pozisyon vektörü……...	51
Çizelge 4.7. Birinci iterasyon yerel en iyi…….	52
Çizelge 4.8. Birinci iterasyon global en iyi……...	52
Çizelge 4.9. Güncellenmiş hızlar……..	52
Çizelge 4.10. İkinci iterasyon pozisyon vektörü……...	53
Çizelge 4.11. İkinci iterasyon yerel en iyi…….	53
Çizelge 4.12. İkinci iterasyon global en iyi……...	53
Çizelge 4.13. Test problemleri çözümleri: düşük varyans…….....	56
Çizelge 4.14. Test problemleri çözümleri: yüksek varyans…….....	58
Çizelge 4.15. Düşük varyans için alt sınıra yakın sonuçlar ....……	62
Çizelge 4.16. Yüksek varyans için sonuçlar .....…….	63
Çizelge 4.17. Genel sonuçlar ve frekansları .....……...	65
Çizelge 4.18. Problemlere göre sonuçların dağılımı ..……..	67
Çizelge 4.19. Dikkate alınan problemler.....……..	69
Çizelge 4.20. Faktör düzeyleri ….....	69

ŞEKİLLERİN LİSTESİ

Şekil	Sayfa
Şekil 2.1. Üretim sistemleri sınıflandırması……..	4
Şekil 2.2. MHDP'nin sınıflandırılması.....	9
Şekil 2.3. (8) Görevli bir öncelik diyagramı (Bowman problemi)……	11
Şekil 2.4. Geleneksel MHD probleminin sınıflandırılması……..	13
Şekil 2.5. Geleneksel düz hat ve U-tipi hat……..	15
Şekil 2.6. U tipi MHDP sınıflandırılması …….....	18
Şekil 2.7. Paralel MHDP sınıflandırılması.....	19
Şekil 3.1. PSO sahte kodu..…….....	26
Şekil 4.1. SUMHD-1 için PSO başlangıç çözümü sahte kodu…….....	42
Şekil 4.2. SUMHDP-1 için PSO başlangıç çözümü akış şeması.....	44
Şekil 4.3. SUMHDP-1 için PSO algoritması sahte kodu.....	45
Şekil 4.4. Hız değerlerinin sınırlandırılması.....	47
Şekil 4.5. SUMHDP-1 için PSO akış şeması.....	48
Şekil 4.6. Bowman problemi öncelik diyagramı.....	49
Şekil 4.7. PSO veri girişi ara yüzü.....	55
Şekil 4.8. Düşük varyans için alt sınıra yakın sonuçlar.....	63
Şekil 4.9. Yüksek varyans için alt sınıra yakın sonuçlar.....	64
Şekil 4.10. Görev sayısı ve alt sınırın göreli frekans ilişkisi.....	65
Şekil 4.11. Problemin ölçeğine göre sonuçlar……..	68
Şekil 4.12. w'nun minimum istasyon sayısına etkisi……	70
Şekil 4.13. c_1'in minimum istasyon sayısına etkisi……..	71
Şekil 4.14. c_2'nin minimum istasyon sayısına etkisi…….	71
Şekil 4.15. V_{max}'ın minimum istasyon sayısına etkisi……..	72
Şekil 4.16. Parçacık sayısının minimum istasyon sayısına etkisi…….	72
Şekil 4.17. w ve c_1'in minimum istasyon sayısına etkisi……..	73
Şekil 4.18. w ve c_2'nin minimum istasyon sayısına etkisi……	73
Şekil 4.19. w ve V_{max}'ın minimum istasyon sayısına etkisi……..	74
Şekil 4.20. w ve parçacık sayısının minimum istasyon sayısına etkisi…….	74

Şekil **Sayfa**

Şekil 4.21. c_1 ve c_2'nin minimum istasyon sayısına etkisi…… 75
Şekil 4.22. c_1 ve V_{max}'ın minimum istasyon sayısına etkisi…….. 76
Şekil 4.23. c_1 ve parçacık sayısının minimum istasyon sayısına etkisi……. 76
Şekil 4.24. c_2 ve V_{max}'ın minimum istasyon sayısına etkisi…….. 77
Şekil 4.25. c_2 ve parçacık sayısının minimum istasyon sayısına etkisi……. 77
Şekil 4.26. V_{max} ve parçacık sayısının minimum istasyon sayısına etkisi……. 78

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte sunulmuştur.

Simgeler	**Açıklama**
C	Çevrim zamanı
c_1	Parçacığın kendi tecrübesi
c_2	Sürünün tecrübesi
c_k	*k*'ıncı istasyonun toplam zamanı
f**(i)**	*i* parçacığının uygunluk fonksiyonu değeri
Gbest	Global en iyi
k	İstasyon numarası
K	Toplam istasyon sayısı
NE_i	*i*'inci çözümdeki istasyon sayısı
Pbest	Yerel en iyi
t_i	*i*'inci görev süresi
X_i^k	Pozisyon vektörü
w	Atalet ağırlığı
V_i^k	Hız vektörü
V_{max}	Maksimum hız
μ_j	*j*'inci görevin ortalama süresi
σ_j	*j*'inci görevin varyansı
$Z_{1-\alpha}$	1- α olasılığı için normal dağılım değeri

Kısaltmalar	**Açıklama**
BMHD-1	Basit montaj hattı dengeleme
ÇMD	Deterministik çoklu/karışık modelli
ÇMS	Stokastik çoklu/karışık modelli
LB	Lower bound (alt sınır)
MHDP	Montaj hattı dengeleme problemi

Kısaltmalar	Açıklama
PSO	Parçacık sürü optimizasyonu
SUMHDP-1	Stokastik U-tipi montaj hattı dengeleme problemi
TMD	Deterministik tek modelli
TMS	Stokastik tek modelli
ULINO	U line optimizer
UMHD	U-tipi montaj hattı dengeleme

1. GİRİŞ

Montaj hattı; Henry Ford'un 1900'lerin başlarında bir mezbahadaki işlemlerin sırayla yapılmasından esinlenerek ve bir kıyaslama örneği olarak ilk defa uyguladığı bir üretim şeklidir. Bir üretim sisteminde, üretilmesi amaçlanan ürün ve/veya yarı ürünlerin belirlenen bir üretim hattında bir araya getirilerek birleştirilmesi ve son aşamada amaçlanan ürün ve/veya yarı ürünlerin elde edilmesi için kullanılan üretim hattına montaj hattı denir.

Seri üretimin artmasıyla montaj hatları zamanla yaygınlaşmış ve 1954'te ilk kez Bryton'ın yüksek lisans tezinde montaj hatlarının dengelenebileceği belirtilmiştir. Salveson [1955] ise montaj hattı dengeleme problemini tanımlamıştır. Geleneksel düz hat konfigürasyonu için tanımlanan basit montaj hattı dengeleme probleminde (BMHDP-1) amaç, verilen bir çevrim süresi için istasyon sayısını minimize etmektir. Bu problemde görev zamanları deterministik veya stokastik olarak ele alınabilmektedir. Literatüre bakıldığında bu konuda bugüne kadar birçok çalışmanın yapıldığı görülmektedir.

Günümüzde rekabetin artmasıyla birlikte ürün çeşitliğinin artması, yüksek miktarlarda talebin azalması ve ürünün müşteriye hızlı ulaşması gibi sebeplerden dolayı daha esnek üretim sistemlerine ihtiyaç doğmuştur. Tam zamanında üretim felsefesinin çıkışıyla geleneksel düz hat montaj hatlarının U-tipine dönüştüğü görülür. Son yıllarda kullanımı artan U-tipi montaj hatlarının geleneksel düz hatlara göre birçok avantajı vardır. Geleneksel hat dengeleme problemi ile U hattı dengeleme problemi arasındaki ana fark; geleneksel hat dengeleme probleminde bir istasyona atanabilir görevler kümesi, öncülleri atanmış görevlerden oluşurken, U hattı dengeleme probleminde bir istasyona atanabilir görevler kümesinin öncülleri atanmış görevler kümesi ile ardılları atanmış görevler kümesinin birleşiminden oluşmasıdır. Böylece U-tipi montaj hattı dengeleme problemi doğmuş ve ilk olarak Miltenburg ve Wijngaard [1994] tarafından tanımlanmıştır. U tipi yerleşimde hattın giriş ve çıkışı aynı pozisyondadır. Bu yerleşimin en dikkat çekici ve önemli avantajı,

üretim miktarındaki değişimlere uyum sağlamada gerekli işçi sayısını arttırabilme ya da azaltabilme esnekliğinin sağlanmasıdır.

Bir montaj hattında görev zamanlarının çoğu zaman deterministik olduğu varsayımı dikkate alınmıştır. Oysaki görev zamanlarının çeşitli sebeplerden dolayı değişkenlik gösterebildiği, başka bir deyişle görev zamanlarının stokastik olabileceği bilinmektedir. Bu durumda görev zamanları için yapılan veri analizleri sonucunda görev zamanlarının normal dağılıma uyan bağımsız değişkenler olduğu varsayımı ele alınmıştır. Görev zamanlarının farklı olasılık dağılımlarına uyması mümkün görünse de literatürde daha çok görev zamanlarının normal dağılıma uyduğunu varsayan çalışmalar mevcuttur. Bu tip problemlerde verilen bir olasılığa göre, her bir istasyonun toplam zamanı, çevrim zamanını aşmayacak şekilde görevler istasyonlara atanır.

Görev zamanlarını stokastik olarak ele alan U-tipi montaj hattı dengeleme problemleri için yapılan çalışmalar oldukça azdır. Bu çalışmada, görev zamanlarının normal dağılımdan gelen bağımsız değişkenler olduğu varsayılarak stokastik U-tipi montaj hattı dengeleme problemi ele alınmaktadır.

Montaj hattı dengeleme problemlerinin çözümünde optimal çözümler sunan doğrusal programlama, dinamik programlama gibi yöntemler kullanılmıştır. Son yıllarda meta sezgisel yöntemlerin de montaj hattı dengeleme problemlerine çözüm bulmada kullanıldığı görülmektedir.

Bu çalışmada ise oldukça yeni bir meta sezgisel yöntem olan parçacık sürü optimizasyonu (PSO), stokastik U-tipi montaj hattı dengeleme probleminin çözümünde önerilmektedir. PSO, Kennedy ve Eberhart [1995] tarafından kuş ve balık sürülerinin yiyecek aramada, bireysel ve sürü halindeki davranışlarından esinlenilerek geliştirilmiş bir tekniktir. PSO'da her birey yada parçacık bir çözümü ifade etmede kullanılır. Sürüdeki parçacık sayısı kadar çözüm aynı anda oluşturulabilmektedir. Parçacıklar, optimuma doğru ilerlemede birbirleriyle iletişim kurarlar. Optimum sonucun bulunması için hem parçacığın hem de sürünün bir

çabası vardır. PSO'nun kullanıldığı Endüstri Mühendisliği alanında yapılmış çalışmalara rastlamak mümkündür.

Çalışmanın ikinci bölümünde, üretim sistemleri ve çeşitleri, montaj hattı dengeleme (MHD), MHD ile ilgili terimler, geleneksel, U-tipi ve paralel MHD ile ilgili literatür araştırmasından bahsedilmektedir.

Çalışmanın üçüncü bölümünde, PSO'nun tanımı, PSO'nun genel yapısı, PSO terimleri, PSO'nun işleyişi ve PSO'nun özellikle endüstri mühendisliği alanında yer alan literatür araştırması yer almaktadır.

Dördüncü bölümde, stokastik U-tipi MHD problemi (SUMHDP-1), SUMHDP-1 için önerilen PSO algoritması ve açıklaması, önerilen PSO algoritmasının uygulandığı sayısal bir örnek, önerilen PSO algoritmasının etkinliğini gösterebilmek amacıyla aynı problemi ele alan iki çalışmanın sonuçlarıyla karşılaştırma, daha büyük problemlerde önerilen algoritma sonuçlarının değerlendirilmesi yer almaktadır. Ayrıca, hangi PSO parametrelerinin optimal çözümün bulunmasında daha etkili olduğunu bulabilmek için parametrelerin ana ve ikili etkileri incelenmektedir.

Çalışmanın son bölümünde, bulunan sonuçların değerlendirilmesi yapılmakta ve gelecekte yapılabilecek çalışmalar için önerilerde bulunulmaktadır.

2. MONTAJ HATTI DENGELEME

2.1. Üretim Sistemleri ve Çeşitleri

Üretim, mühendisler tarafından, fiziksel bir varlık üzerinde onun değerini artırıcı bir değişiklik yapmak veya hammadde ve yarı mamulleri mamule dönüştürmek olarak tanımlanmaktadır [Acar, 1996]. Fiziksel bir varlığın değerini arttırıcı bir değişiklik yapan tasarım süreci ise üretim sistemi olarak tanımlanabilir. Üretim sistemlerini bu çalışmanın amacına uygun olarak Şekil 2.1'deki gibi sınıflandırmak mümkündür:

Şekil 2.1. Üretim sistemleri sınıflandırması

Siparişe göre yada atölye tipi üretim, tüketicinin veya müşteri firmanın, zaman, tasarım, miktar ve kalite bakımından özel olarak belirlediği bir mamulün üretilmesidir [Üreten, 1999]. Bu tip üretimde, küçük miktarlarda fakat çok çeşitli ürün üretilir. Bu sistemlerde, çok yönlü tezgâhlar kullanılmakta ve farklı tezgahlarda çalışabilecek esnek işçi kullanımı gerekmektedir. Bu tip üretime, yerinde alınan ölçüye göre mobilya üreten sistemler örnek olarak verilebilir.

Parti tipi üretim, özel bir siparişi veya sürekli bir talebi karşılamak amacıyla belli bir mamul grubunun belirli miktarlardan oluşan partiler halinde üretilmesidir. Bu tip üretimde oldukça çeşitli ürünlerin, değişik miktarlarda ve genellikle orta büyüklükteki partiler halinde üretildiği söyleyenebilir. Bu sistemlerin en büyük özelliği bir parti bitmeden diğerinin üretimine geçilmemesidir. Talep süreklidir ve sipariş tipi üretimde olduğu kadar değişken değildir. Metal işleme atölyeleri ile bir traktör fabrikasının motor atölyesinde gerçekleştirilen üretimi bu tip üretime örnek olarak verilebilir.

Seri üretim, yüksek miktarlarda fakat düşük seviyede çeşitlilik gösteren ürünler için uygulanan üretim şeklidir. Seri üretimin ana özelliği, ürün akışının olması ve tesislerin üretilecek ürüne göre tasarlanmasıdır. Seri üretimde, sistemin başlangıç noktasından başlayan hammadde, yarı mamul ve parça gibi girdiler, işlem birimlerinden geçmekte ve son ürün haline dönüşerek sistemden çıkmaktadırlar.

Seri sistemlerin ayırt edici özellikleri aşağıdaki gibi sıralanabilir [Üreten, 1999]:

- Düzenli ve yüksek talep,
- Az çeşit, yüksek miktar üretim,
- Yüksek sermaye yatırımı,
- Tüm ürünlerin aynı makineler üzerinde aynı sırayla işlem görmesi,
- Özel amaçlı makineler kullanılması,
- Kalifiye olmayan işgücü kullanımı,
- Yüksek hammadde ve mamul stokları, buna karşılık düşük ara stoklar,
- Taşımacılıkta konveyör, raylı araba gibi özel araçlar kullanılmasıdır.

Seri üretim sistemleri dört ana kural üzerine kurulmuştur [Gökçen, 1994]:

i. İş Akışı Prensibi: Bu sistemlerde iş, malzeme ve ürünlerin düzgün ve düzenli bir şekilde üretim tesisleri boyunca akmaları gerekir.

ii. İkame Edilebilen Parça Kullanımı: Montaj hatları büyük ölçüde ikame edilebilen parçaların kullanımına bağlıdır. Başka bir deyişle, üretimde kullanılan yarı mamul ve parçaların yeterli doğruluk standartlarında olmaları gerekmektedir. Bu şekilde ana ürünü oluşturan parçaların (bileşenlerin) birbirinin yerine kullanılabilmeleri mümkündür.

iii. Minimum Uzaklık Prensibi: İş akışının sürekliliğinin sağlanması ve kullanım alanının en iyi şekilde değerlendirilmesi, büyük ölçüde kullanılan akış modelinin akılcı ve etken olmasına bağlıdır. Bu durumda üretim merkezlerinin birbirine yakın olmaları, hareketli bantlar ve benzeri malzeme taşıma sistemleri ile birbirine bağlanmış olmaları gereklidir.

iv. İş Yükünün Bölünmesi: İş yükünün bölünmesi ve rasyonelleştirilmesi veya montaj hatlarında işgücünün bölünmesi, seri üretim sistemlerinin önemli özelliklerinden biridir.

Sürekli seri üretimde, ürün akışı süreklidir. Demir-çelik ürünler, çimento ve gıda maddeleri bu tip üretime örnek olarak gösterilebilir.

Kesikli seri üretim de, sürekli seri üretim gibi aynı temel prensip üzerine gelişmiştir. Karmaşık birimler akış kuralı kullanılarak büyük miktarlarda özel akış sistemlerinin tasarımı ile üretilmektedirler. Otomobil, beyaz eşya, televizyon vb. ürünler kesikli seri üretime örnektir. Kesikli seri üretim, transfer hatları ve montaj hatları olarak ikiye ayrılmaktadır.

Transfer hatları, büyük ve karmaşık makinelerden oluşan üretim sistemleridir. Bu hatlar düz yada devirli tiplerde olabilir. Ürün ya da malzemelerin bir yerden başka bir yere nakli insan gücü ile değil otomatik hareket ve transfer yöntemiyle sağlanmaktadır. Böylece transfer hatları, transfer aletleri ile bağlanmış otomatik imalat makine serilerinin oluşturduğu üretim hatlarıdır denilebilir [Gökçen, 1994].

Montaj hatları günümüz endüstrisinde önemli bir yeri olan kitle üretim yöntemlerinden birisidir. Bir üretim sisteminde üretilmesi amaçlanan ürün ve/veya yarı ürünlerin belirlenen bir üretim hattında bir araya getirilerek birleştirilmesi ve son aşamada amaçlanan ürün ve/veya yarı ürünlerin elde edilmesine montaj, bu üretim hattına da montaj hattı denir.

Çeşitli tipte montaj hatları vardır ve bu hat tipleri arasındaki bazı farklılıklar şöyle sıralanabilir:

- malzeme taşıma aygıtları (kayış veya rulo konveyör, baş-üstü vinç),
- hat konfigürasyonu (U-şekilli, düz, kollara ayrılan),
- yürüyüş (mekanik, insan),
- ürün karışımı (tek ürün veya çok ürün),
- istasyon karakteristiği (işçiler oturabilir, ayakta durabilir, hat ile yürüyebilir veya hattı yürütebilir),
- hattın uzunluğu (birkaç işçi veya birçok işçi) [Chase ve ark., 2001].

2.2. Montaj Hattı Dengeleme Problemi

Montaj Hattı Dengeleme Problemi (MHDP), klasik bir Endüstri Mühendisliği problemidir. Problemin temel yapısı, üretim sistemlerinin tasarımı ile ilgilidir. Modern fabrikanın yapısı değişmektedir. Yeni fabrikaların tasarımında en önemli konulardan biri esnekliktir. Dolayısıyla, hat dengeleme bir defalık bir yerleşim planından ziyade dinamik bir planlama problemi haline gelmektedir [Nahmias, 2005].

MHDP, montaj işleminin yapılabilmesi için gerekli işler, bu işlerin süreleri ve işler arasındaki öncelik ilişkileri verildiğinde bir performans ölçütünü optimum şekilde sıralı iş istasyonlarına atanması şeklinde tanımlanabilir. Bu tanıma göre hat dengeleme problemi öncelik ilişkilerini bozmayacak şekilde görevlerin, bazı verimlilik ölçütlerini (denge gecikmesi minimizasyonu, istasyon sayısı

minimizasyonu, üretim oranının maksimizasyonu vb.) sağlayacak şekilde sıralı istasyonlara atanmasıdır.

MHDP, hesaplama karmaşıklığı sebebiyle NP-hard tipi problem olarak adlandırılır. Eğer bir problemde *m* görev ve *r* öncelik ilişkisi varsa, bu durumda $\frac{m!}{2^r}$ olası görev sıralaması vardır [Ajenblit ve Wainwright, 1998]. Örneğin; küçük boyutlu 8 görevli bir problemde 8 de öncelik ilişkisi varsa $\frac{8!}{2^8} = 19490.625$ olası görev sıralaması yapılabilir. Buna göre optimal sonucun bulunabilmesi için toplam 19490 deneme yapılması gerekebilir. Eğer görev sayısı 30 ve öncelik ilişkisi 60 alınırsa bu durumda gerekli deneme sayısı 230070181493096.4 olacaktır. Daha büyük boyutlu problemlerde gereken deneme sayısı çok daha fazla artmaktadır. MHDP'ni zor yapan budur.

MHD'nin amaçları şöyle sıralanabilir [Meyers ve Stephens, 2005]:

- İşçiler arasında iş yükünü dengelemek,
- Darboğaz işlemlerini tanımlamak,
- Montaj hattının hızını belirlemek,
- İstasyon sayısını tespit etmek,
- İşçilik maliyetini belirlemek,
- Fabrika yerleşimine yardım etmek,
- Üretim maliyetini azaltmak.

Montaj hattı dengeleme problemini Şekil 2.2'deki gibi Geleneksel Düz Hat, U-tipi Hat ve Paralel Hat olmak üzere üç kategoride incelemek mümkündür [Benzer, 2005]:

Şekil 2.2. MHDP'nin sınıflandırılması

2.2.1. MHD ile ilgili terimler

MHDP'de sık kullanılan kavramlar aşağıda verilmiştir[Benzer, 2005].

Görev: Toplam iş kapsamının mantıksal olarak bölünmüş yada daha fazla ayrıştırılamayan en küçük iş öğesidir. Görev *i* indisiyle ifade edilir, problemdeki görev sayısının toplamı ise *N* ile gösterilir.

Görev Zamanı: Bir görevi tamamlamak için gerekli süredir ve *i* görevine ait görev zamanı t_i ile gösterilir. Montaj hattı dengeleme problemlerinde görev zamanları deterministik veya stokastik olabilmektedir. Bir görev zamanı μ_i ortalama ve σ^2 varyansa sahip normal dağılımdan gelirse 1-α olasılıkla görev zamanı Eş.2.1'deki gibi bulunur.

$$t_i = \mu_i + Z_{1-\alpha}\sqrt{\sigma_i^2} \tag{2.1}$$

İş istasyonu: Üretim hattında yapılması öngörülen toplam iş miktarının bir kısmının yerine getirildiği yerdir. Her bir istasyonda ürünlerin üretilmesi için gereken *N* operasyonun bir alt kümesi gerçekleştirilir ve *k* ile gösterilir. Toplam istasyon sayısı ise *K* ile ifade edilir.

Çevrim Zamanı: Hattın sabit veya aralıklı hareketleriyle her ürünün her bir istasyonda işlem gördüğü zamandır ve *C* ile gösterilir. Başka bir ifadeyle, standart bir

akış temposuyla bir üretim hattında tamamlanıp çıkan, ardı ardına iki ürün arasındaki fark olarak tanımlanır ve aşağıdaki gibi hesaplanır:

$$C = \frac{\sum t_i}{N} \quad \text{ya da} \quad C = \frac{T}{Q} \tag{2.2}$$

Burada Q, birim zamanda üretilecek miktar; T, ise üretimin gerçekleştirileceği zaman dilimini ifade etmektedir.

İstasyon Zamanı (İş Yükü): Aynı istasyonda yapılan görevlerin zamanları toplamıdır. k istasyonuna ait zaman c_k ile gösterilir ve montaj hattındaki bir istasyon zamanı hattın çevrim zamanından (C) büyük olmamalıdır (Eş. 2.3).

$$\text{Max } t_i \leq \text{Max } c_k \leq C \tag{2.3}$$

İstasyon Gecikme Zamanı (Boş Zaman): Çevrim zamanı ile istasyon zamanı arasındaki farka eşittir ve aynı zamanda istasyon boş zamanı olarak da adlandırılır.

$$\text{Boş zaman} = C - c_k \tag{2.4}$$

Toplam Boş Zaman: Bir hattın tasarım etkinliğinin bir ölçüsü olarak bütün istasyonların boş zamanlarının toplamıdır (Eş.2.5).

$$TBZ = \sum_{k=1}^{K} (C - c_k) \tag{2.5}$$

Öncelik Diyagramı: Bir ürünün montajında yapılması gereken görevlerin işlem sıralarının grafiksel olarak gösterimine imkân veren bir araçtır. Şekil 2.3'te Bowman problemine ait 8 görevli bir öncelik diyagramı verilmiştir. Dairelerin içindeki numaralar görevleri, dışındakiler ise görev zamanlarını ifade etmektedir. Diyagram soldan sağa doğru çizilen yönlü bir diyagramdır.

Şekil 2.3. (8) Görevli bir öncelik diyagramı (Bowman problemi)

Öncelik ilişkilerini gösteren diğer bir teknik *öncelik matrisidir.* Öncelik diyagramındaki *i* görevini *j* görevi takip ediyorsa matriste *i.* satır *j.* sütun 1, aksi halde 0 değerini alır. Çizelge 2.1'de, Şekil 2.3'te verilen Bowman problemine ait öncelik matrisi verilmektedir.

Çizelge 2.1. Öncelik matrisi

	1	2	3	4	5	6	7	8
1	-	1	1	1	1	1	1	1
2		-	1	1	1	1	1	1
3			-	0	1	1	1	1
4				-	0	1	0	1
5					-	0	1	0
6						-	0	1
7							-	0
8								-

Denge Gecikmesi: Etkinliğin bir ölçüsü olan denge gecikmesi, ürünün, hattın başından sonuna hareketi sırasında toplam boş zamanın tüm istasyonlara kötü bir şekilde dağıtılması sonucunda, bazı istasyonlardaki operatörlerin boş kaldığı süreye denge gecikmesi (*d*) denir. Bu süre, montaj hattının dengesizliğinin de bir ölçüsü olarak kullanılır. Mükemmel dengelenmiş hatlarda denge gecikmesi sıfırdır.

$$d = \frac{\left(K.C - \sum_{i=1}^{N} t_i \right)}{K.C} \qquad (2.6)$$

Hattın Etkinliği: Verimlilik ölçütü olarak kullanılır ve aşağıdaki şekilde hesaplanır (Eş. 2.7). Bu oran her zaman 0 ile 1 arasında değer almak durumundadır.

$$e = \frac{\sum_{k=1}^{K} c_k}{K.C} \qquad \text{yada} \qquad e=1 - d \tag{2.7}$$

Esneklik Oranı: Esneklik oranı (F-ratio) montaj işlemindeki görevlerin öncelik ilişkisine göre bulunabilecek uygun sıralamaların nispi belirleyicisidir. Bir öncelik diyagramının esnekliği öncelik matrisinden faydalanılarak tespit edilir. Esneklik, problem çözümünü kolaylaştıran ya da zorlaştıran bir faktördür. Eş. 2.8'deki gibi hesaplanır.

$$E - Oranı = \frac{2.Y}{N(N-1)} \tag{2.8}$$

Eşitlikteki Y değişkeni öncelik matrisindeki sıfır sayısını, N ise görev sayısını temsil etmektedir.

2.2.2. Geleneksel düz hat MHD

Geleneksel MHD, montaj işleminin yapılabilmesi için gerekli görevler, bu görevlerin aldıkları süreler ve aralarındaki öncelik ilişkileri verildiğinde, bir performans ölçüsü optimum yapılacak şekilde, görevlerin istasyonlara atanmasıdır. Geleneksel MHD problemi Şekil 2.4'teki gibi sınıflandırılabilir [Ağpak, 2001]. Deterministik Tek Modelli (TMD) montaj hattı dengeleme problemlerinde hatta yalnız bir modelin üretildiği ve görev zamanlarının bir birimden diğerine herhangi bir değişiklik göstermediği yani deterministik olduğu varsayılır. TMD problemler geleneksel montaj hattı dengeleme problemlerinin en basit halidir. Paralel istasyon veya bölgeleme kısıtları gibi faktör veya kısıtlar eklendiğinde TMD genel durum olarak adlandırılırlar.

Stokastik Tek Modelli (TMS) MHDP, görev zamanlarındaki değişkenlik nedeniyle oluşmuştur. Buna göre problem, öncelik ilişkileri dikkate alınarak ve belirli bir performans ölçüsü optimum yapılacak şekilde görevlerin uygun iş istasyonlarına atanmasıdır. Elle montaj hatları için daha gerçekçi bir yaklaşımdır. Hatta tamamlanamayan görevler, ya en kısa sürede tamamlanmak üzere hattın dışına alınırlar ve orada tamamlanırlar yada bitmemiş ürün olarak ayrılırlar.

Deterministik Çoklu/Karışık Modelli (ÇMD) MHDP'de görev zamanları belirlidir. Bu tip hatlarda, farklı ürünlerin montajı yapılır. İki veya daha fazla ürün partiler halinde üretilirler. Partiler birbirine karıştırılmadan sırayla üretilir. Modeller farklı ürünler yada aynı ürünün farklı modelleri olabilir. Her iki durumda da ürünler aynı olmayan fakat benzer üretim ihtiyaçlarını gösterir.

Şekil 2.4. Geleneksel MHD probleminin sınıflandırılması

Karışık modellide ise aynı anda birden fazla benzer tipteki modelin karışık olarak üretimi söz konusudur. Bu tip üretimin en önemli faydası, müşteri isteğini karşılamak üzere değişik modellerin sürekli olarak üretilmesi ve büyük mamul stoklarına gidilmemesidir. Modellerin değişik işlem zamanlarından doğan dezavantajı ise iş

akışının düzenli olmaması, dolayısıyla da daha fazla istasyon boş zamanı ve yarı mamulden oluşan yığınlardır.

Stokastik Çoklu/Karışık Modelli (ÇMS) MHDP'de görev zamanlarının stokastik olmasına izin verilir. Görev zamanları TMS'de olduğu gibi parametreleri bilinen dağılımlarla ifade edilirler. Öğrenme etkileri, işçi beceri seviyesi, iş tasarımı ve işçinin iş zamanı değişikliği gibi faktörler problemi zorlaştırır. Bunlardan dolayı, birimlerin tamamlanmadan hat boyunca ilerlemeleri ve dolayısıyla da tamamlanmayan birimlerin hattın dışında tamamlanması, bir takım maliyetlere sebep olmaktadır [Ghosh ve Gagnon, 1989].

Becker ve Scholl [2006] basit montaj hattı dengeleme problem (BMHDP) tiplerini Çizelge 2.2'deki gibi sınıflandırmışlardır. Bu problem tiplerinde görev zamanları deterministik veya stokastik olabilmektedir.

Çizelge 2.2. Basit montaj hattı dengeleme problem tipleri

		Çevrim zamanı	
		verilir	*minimum yapılır*
İstasyon sayısı	*verilir*	BMHDP -F	BMHDP -2
	minimum yapılır	BMHDP -1	BMHDP -E

BMHDP-1: Çevrim zamanı verildiğinde istasyon sayısı minimum yapılır.

BMHDP-2: İstasyon sayısı verildiğinde çevrim zamanı minimum yapılır.

BMHDP-E: İstasyon sayısı ve çevrim zamanı aynı anda minimum yapılırken hat etkinliği maksimum yapılır.

BMHDP-F: İstasyon sayısı ve çevrim zamanı verildiğinde mümkün bir atama bulunur.

2.2.3. U-tipi MHD

Geleneksel düz hat konfigürasyonundan sonra U-tipi hatlar önerilmiştir. U tipi yerleşimde hattın giriş ve çıkışı aynı pozisyondadır (Şekil 2.5). Bu yerleşimin en dikkat çekici ve önemli avantajı, üretim miktarındaki değişimlere uyum sağlamada gerekli işçi sayısını arttırabilme yada azaltabilme esnekliğinin sağlanmasıdır.

U tipi montaj hatlarının ilk uygulaması Toyota Üretim sisteminde olmuştur. U tipi hatlarda çok fonksiyonlu işçi kullanır. Eğer giriş ve çıkış şekildeki gibi yakın olursa bu yerleşim şekli U tipi yerleşim olarak, eğer bu yerleşim içerisinde aynı işçi birden fazla makinede çalışıyorsa, bu durumda da U tipi üretim hattı olarak adlandırılır.

a. Geleneksel düz hat

b. U-tipi hat dengeleme

Şekil 2.5. Geleneksel düz hat ve U-tipi hat

Miltenburg ve Wijngaard [1994] deterministik U-tipi hat dengeleme problemini (UMHDP) ortaya atmıştır. Geleneksel hat dengeleme probleminde istasyona atanacak görevler belirlenirken öncülleri atanmış görevler arasından seçim yapılır.

UMHDP'de ise istasyona atanacak görevler hem öncülleri atanmış hem de ardılları atanmış görevlerin oluşturduğu kümeden seçilir. Bu açıdan U-tipi hatlarda geleneksel hatlara eşit yada daha az istasyon olması beklenir.

Miltenburg ve Wijngaard [1994] deterministik UMHDP'nin doğrusal programlama modelini de sunmaktadır.

F : Görevlerin kümesi $F = \{k | k = 1,2,...,n\}$,

P : Öncelik kısıtlarının kümesi

$P = \{ (x, y) |$ görev x görev y den önce tamamlanmalıdır$\}$,

T : Görev zamanlarının kümesi $T = \{t(k) | k = 1,2,...,n\}$,

C : Çevrim zamanı,

F'nin alt kümelerinin birleşimi bulunur.

S_i : i. istasyon $S_i = \{ k |$ görev k istasyon i'de yapılır$\}$ olmak üzere F, istasyonların $(S_1, S_2, ..., S_N)$ birleşimidir.

Aşağıdaki kısıtların sağlanması gerekir:

$$\bigcup_{i=1}^{N} S_i = F \,, \tag{2.9}$$

$$S_i \bigcap_{i \neq j} S_j = \phi \,, \tag{2.10}$$

$$\sum_{k \in S_i} t(k) \leq C, \qquad i = 1,2,...,N \tag{2.11}$$

her bir y görevi için:

eğer $(x, y) \in P, x \in S_i, y \in S_j$, $i \leq j$ olduğunda, tüm x'ler için veya

eğer $(y,z) \in P, y \in S_j, z \in S_k,\ k \leq j$ olduğunda, tüm z'ler için (2.12)

$$\left[NC - \sum_{i=1}^{N} \sum_{k \in S_i} t(k) \right] \text{ minimize edilir} \quad (2.13)$$

Eş.2.9 tüm görevlerin bir istasyona atanmasını sağlar. Eş.2.10'un sonucunda her bir görev sadece bir kez atanır. Eş.2.11 istasyon toplam zamanının çevrim zamanını aşmasını engeller. Eş.2.12 U-tipi hatta öncelik ilişkilerinin ihlal edilmemesini sağlar. Son olarak Eş.2.13 istasyon sayısının minimize edilmesi amaç fonksiyonunu ifade etmektedir.

U tipi hatlarda çok fonksiyonlu işçi, birden çok makine ile ilgilenir ve bir çevrim zamanı içerisinde her birinde işlem yapar. İşçi bu makinelerden birine vardığı zaman eğer parçanın işlemi devam ediyorsa işlemin bitmesini bekler ve daha sonra parçayı bir sonraki makineye gönderir. Yeni parçayı boşalan makineye yerleştirir, operasyonu başlatır ve sonraki makineye gider. Burada işçinin çevrim zamanı onun ilk makineye ard arda varışları arasındaki zaman aralığıdır. Bu zaman aralığı bekleme zamanı, operasyon zamanı ve yürüme zamanını içerir [Ohno ve Nakade, 1997]. Bu durumda U tipi hat dengeleme problemi genel durum özelliği kazanır. Bekleme zamanı, operasyon zamanı ve yürüme zamanının dahil olmadığı U tipi hat dengeleme problemleri basit durum olarak ifade edilebilir.

Şekil 2.6'da, UMHDP için yapılan sınıflandırma şematik olarak verilmektedir [Ağpak, 2001].

Şekil 2.6. U tipi MHDP sınıflandırılması

Şekil 2.6'daki U-tipi MHDP ile ilgili tanımlar, daha önce geleneksel düz hat MHDP için yapılan tanımlarla aynıdır.

Geleneksel düz hatlar için geçerli problem tipleri U-tipi hatlar için de geçerlidir. Çizelge 2.2'de belirtildiği gibi aşağıdaki tanımlar yapılır:

UMHDP-1: Çevrim zamanı verildiğinde istasyon sayısı minimum yapılır.

UMHDP-2: İstasyon sayısı verildiğinde çevrim zamanı minimum yapılır.

UMHDP-E: İstasyon sayısı ve çevrim zamanı aynı anda minimum yapılırken hat etkinliği maksimum yapılır.

UMHDP-F: İstasyon sayısı ve çevrim zamanı verildiğinde mümkün bir atama bulunur.

Eğer U-tipi hatlarda görev zamanları stokastik olursa problem tipleri; SUMHDP -1, SUMHDP -2, SUMHDP -E, SUMHDP -F olur.

2.2.4. Paralel MHD

Paralel MHD birçok paralel montaj hattını içeren bir üretim sistemini dikkate alır. Paralel MHD, aynı ya da benzer ürünlerin üretildiği birden fazla paralel hatta, istasyonlara görev atamalarının eş zamanlı yapılması ve dolayısıyla kaynak enküçüklemesini hedefleyen yeni bir kavramdır [Benzer, 2005]. Paralel MHDP'de birden fazla montaj hattının birlikte dengelenmesiyle, gerekli operatör sayısının enazlanması amaçlanmaktadır.

Paralel MHDP için sınıflandırma Şekil 2.7'de verilmektedir. Burada "aynı modelli" ifadesi birden fazla hatta, aynı ürünlerin/modellerin üretilmesini, "farklı modelli" ifadesi ise birden fazla hatta farklı modellerin üretilmesi anlamındadır. Aktif durum birden fazla hattın aynı anda ortak kaynaklarla dengelenmesini, Pasif durum ise, birden fazla hattın (genellikle 2) ayrı ayrı dengelenerek, daha sonra ortak kaynakla dengenin iyileştirilmesi anlamındadır.

Şekil 2.7. Paralel MHDP sınıflandırılması

2.3. MHD Literatür Araştırması

MHDP literatür araştırması, MHDP sınıflandırmasına göre geleneksel düz hat, U-tipi hat ve paralel hatlar için yapılmıştır.

2.3.1. Geleneksel düz hat MHD ile ilgili literatür

Montaj hatlarının dengelenmesi düşüncesi ilk kez Bryton'ın yüksek lisans tezinde yer almıştır. Montaj hattı dengeleme problemi ise ilk defa Salveson tarafından 1955 yılında ortaya atılmıştır. Bugüne kadar BMHD–1 için bir çok optimal çözüm yöntemi ileri sürülmüştür. Bu yöntemlere literatür araştırması yapan çalışmalardan ulaşılabilir: Baybars [1986], Ghosh ve Gagnon [1989], Erel ve Sarin [1998], Scholl [1999], Scholl ve Becker [2006], Becker ve Scholl [2006].

MHD probleminin TMD sınıfı için optimal ve sezgisel sonuçlar üreten çok sayıda metot bulunmaktadır. Optimal çözümü veren metotlardan tamsayılı programlama çalışmaları: Bowman [1960], Klein [1963], Patterson ve Albracht [1975]; dinamik programalama çalışmaları: Jackson [1956], Held ve arkadaşları [1963], Kao ve Queyranne [1982], dal ve sınır algoritması; Johnson [1981] ve amaç programlama ile ilgili olarak; Gunther ve arkadaşları [1983], McMullen ve Frazier [1998], Ağpak ve Gökçen [2005] çalışmalarına rastlanabilir.

BMHD–1 problemi için meta-sezgiseller de kullanılmıştır: MHDP'nin çözümünde genetik algoritmayla ilgili Suresh ve arkadaşları [1996], Kim ve arkadaşları [2000], Sabuncuoğlu ve arkadaşları [2000] çalışmaları bulunmaktadır. Karınca algoritmasıyla ilgili McMullen ve Tarasewich [2003], Bautista ve Pereira [2007], tabu arama ile Lapierre ve arkadaşları [2006] çalışmışlardır. Suresh ve Sahu [1994] BMHD–1 için tavlama benzetimi yaklaşımı önermektedir.

Problemin karmaşıklığı nedeniyle birçok da klasik sezgisel metot geliştirilmiştir. Bunlardan bazı önemlileri şunlardır: Tonge [1960], Kilbridge ve Wester [1961], Dar-EI [1973], Dar-el ve Rubinovitch [1991].

Problemin TMD sınıfının çözümü için tasarlanmış özel bilgisayar yazılımları da şunlardır: COMSOAL [Arcus, 1966], SALOME [Scholl ve Klein, 1997].

Stokastik montaj hattı dengeleme ise ilk defa Moodie ve Young [1965] tarafından ortaya koyulmuştur. Stokastik görev zamanlarının genellikle normal dağılıma uyan bağımsız rassal değişkenlerden oluştuğu kabul edilmektedir. Verilen bir olasılığa göre, her bir istasyonun toplam zamanı, çevrim zamanını aşmayacak şekilde görevler istasyonlara atanır. Görev zamanlarının farklı olasılık dağılımlara uyduğunu kabul eden çalışmalar da vardır: Kao [1976, 1979], Sniedovich [1981], Nkasu ve Leung [1995].

TMS problem kategorisinde Kottas ve Lau [1973, 1976], Carraway [1989], Sarin ve Erel [1990], Sarin ve arkadaşlarının [1999], Gökçen ve Baykoç [1999], Liu ve arkadaşları [2005], Ağpak ve Gökçen [2007] çalışmalarına rastlamak mümkündür.

Karışık modelli montaj hatları için sezgisel metotlarla ilgili Macaskill [1972], Chakravarty ve Shtub [1985] çalışmaları bulunmaktadır. Buna karşılık probleme optimal çözüm bulan tam sayılı programlamanın uygulandığı Gökçen ve Erel [1998], Erel ve Gökçen [1999], Gökçen ve Erel [1997] çalışmalarına rastlamak mümkündür.

2.3.2. U-tipi MHD ile ilgili literatür

U-tipi hat dengeleme ile ilgili çalışmalar, geleneksel düz hat ile ilgili çalışmalardan çok daha azdır. Determistik U tipi hat dengeleme konusunda literatürde yer alan çalışmalar şöyledir: 20 göreve kadar çözüm sağlayabilen dinamik programlama algoritması Miltenburg ve Wijngaard [1994] tarafından ortaya konmuştur. Aynı çalışmasında Miltenburg klasik hatlar için uygulanan RPWT tekniğini U tipi hat dengeleme problemi için yeniden düzenlemiş ve uygulamıştır.

Sparling ve Miltenburg [1998], karışık modelli U tipi hat dengeleme problemi için bir çözüm yaklaşımında bulunmuşlardır. Yine Miltenburg [1998] birden fazla U-tipi hat olduğunda, bu hatlar arasında optimal dengeyi sağlayan bir dinamik programlama algoritması sunmaktadır.

Urban [1998] optimal dengenin tespiti için tamsayılı programlama formülasyonu önermektedir. Scholl ve Klein [1999] deterministik U-tipi hat dengeleme için dal sınır algoritmasıyla çalışan ve belki de bilinen en iyi sonuçları üretebilen ULINO (U Line Optimizer) önermektedir. ULINO, literatürde 45 görevden az olan problemler yanında 45 görevden büyük problemlerde de etkin çözümler verebilmektedir. Bunların haricinde bu problem tipini ele alan çalışmalar şunlardır: Ohno ve Nakade [1997], Ohno ve arkadaşları [1997], Nakade and Ohno [1999], Miltenburg [2001], Gökçen ve arkadaşları [2005], Gökçen ve Ağpak [2006], Toksarı ve ark. [2008].

Meta-sezgiseller de deterministik U-tipi hat dengelemede kullanılmıştır. Ajenblit ve Wainwright [1998] genetik algoritmayı, Erel ve arkadaşları [2001] tavlama benzetimini deterministik U-tipi hat dengeleme problemlerinde kullanmıştır.

Stokastik U-tipi hat dengeleme problemleri için literatürde yer alan çalışmalar çok az sayıdadır: Nakade ve arkadaşları [1997], stokastik U tipi hat dengeleme probleminin genel durumu için hattın beklenen çevrim zamanının alt ve üst sınırlarının tespiti üzerinde çalışmışlardır. Chand ve Zeng [2001] U-tipi ve geleneksel düz hat yerleşimin göreli performansında stokastik görev zamanlarının etkisini araştırmaktadır. Ağpak ve arkadaşları [2002] stokastik görev zamanlı tek modelli U-tipi hat dengeleme problemi için COMSOAL metodunu baz alan yeni bir sezgisel prosedür kullanmaktadır. Guerriero ve Miltenburg [2003] U-tipi hat dengeleme problemi için tamamlanma zamanları herhangi bir dağılım fonksiyonuna uyduğunda optimal çözümün bulunması için özyinelemeli (tekrarlayan) bir algoritma sunmaktadır. Urban ve Chiang [2006] stokastik görev zamanlı U-tipi hat dengeleme problemi için şans kısıtlı parçalı (piecewise) doğrusal tamsayılı programlama modeli

formüle etmektedir. Chiang ve Urban [2006] bir başlangıç mümkün çözüm modülü ve çözüm geliştirme modülünü içeren melez bir sezgisel ileri sürmektedir. Baykasoğlu ve Özbakır [2007] ise bu problem için çoklu kural bazlı (multiple-rule-based) yeni bir genetik algoritma önermektedir.

2.3.3. Paralel MHD ile ilgili literatür

Paralel MHD ile ilgili henüz çok az sayıda çalışma olduğu görülmektedir. Paralel MHDP problemini, ilk kez Gökçen ve arkadaşları [2006] tanımlamıştır. Gökçen ve arkadaşları [2006] çalışmasında paralel hatların dengelenmesi probleminde, pasif ve aktif durumu ayrı ayrı ele aldıktan sonra, farklı çevrim zamanları için de problemi irdelemiştir. Ayrıca Paralel MHDP'yi ilk kez ele alan bu çalışmada 0-1 tamsayılı programlama modeli de önerilmektedir. Daha sonra Benzer ve arkadaşları [2007] Paralel MHDP için en kısa yol formülasyonunu kullanan bir network modeli önermektedir. Scholl ve Boysen [2009] Paralel MHDP'nin tanımını ve 0-1 doğrusal programlama modelini daha ayrıntılı olarak ele almaktadır. Kesin çözümler bulduğu bilinen dal-sınır prosedürlü SALOME, bu probleme uyarlanmaktadır. Önerilen prosedür küçük ve orta büyüklükteki problemlerde optimal sonuçlar verirken, büyük ölçekli problemler için iyi derecede sezgisel çözümler sağlamaktadır.

3. PARÇACIK SÜRÜ OPTİMİZASYONU

Bu bölümde, çalışmada ele alınan stokastik U-tipi hat dengeleme problemini çözmek için kullanılan parçacık sürü optimizasyonu anlatılmaktadır.

3.1. PSO'nun Tanımı

PSO, kuş ve balık sürülerinin iki boyutlu davranışlarının bir benzetimidir. Kuş ve balık sürülerinin yerini bilmedikleri bir yiyeceği aramaları yada belli bir yere giderken birbirleriyle iletişim kurarak o yere ulaşmalarından esinlenilmiştir. PSO'nun ana fikri, popülasyondaki bireyler arasındaki sosyal bilgi paylaşımını (social sharing of information) geliştirmektir. Her bir bireye parçacık denir ve parçacıklardan oluşan popülasyona da sürü (swarm) ismi verilir. PSO'da arama işlemi popülasyondaki bireyler tarafından ve belirlenen parçacık sayısınca yapılır.

Kuşların uzayda, yerini bilmedikleri yiyeceği aramaları, bir problem için optimum çözüm aramaya benzetilmiştir. Kuşlar yiyecek ararken aralarında iletişim kurarak yiyeceğe en yakın olan kuşu takip ederler. Her tekil çözüm, arama uzayındaki bir kuştur. Parçacık her birim hareketinin sonunda bulunduğu koordinatlarını bir uygunluk fonksiyonuna gönderir ve böylece parçacığın uygunluk fonksiyonu değeri belirlenir. Bir parçacık, pozisyonunu (koordinatlarını), hızını (çözüm uzayındaki her boyutta ne kadar hızla ilerlediği), şimdiye kadar elde ettiği en iyi uygunluk değerini ve bu değeri elde ettiği pozisyonu hafızasında tutar.

Parçacığın, her iterasyonda çözüm uzayında hangi yönde ve hangi hızda hareket edeceği diğer parçacıkların (komşular) en iyi koordinatları ve kendi kişisel en iyi koordinatlarının bir birleşimi olacaktır. Parçacık optimuma ulaşırken bir sonraki adımını hem kendi en iyi pozisyonuna (optimuma en yakın) hem de sürüdeki en iyi uygunluk değerine sahip parçacığın pozisyonuna göre ayarlar.

Sürüdeki her bir parçacık çok boyutlu arama uzayında bir çözümü yani pozisyonu temsil eder. Parçacıklar, ilk pozisyonlarını rasgele belirler, daha sonra arama uzayı içinde adım adım hareket eder ve verilen amaç fonksiyonu için çözümleri arar.

PSO, parçacıklar arasındaki bilgi paylaşımını esas almaktadır. Her bir parçacık optimuma ulaşmak isterken yeni pozisyonunu, kendi önceki tecrübelerinden ve bir önceki sürüdeki (yada iterasyondaki) en iyi pozisyonu dikkate alarak bulur. Global komşular, her bir iterasyonda (optimum noktaya doğru yapılan her bir hareket) sürü içerisindeki en iyi uygunluk fonksiyonu değerini veren parçacığın değerleridir. Her bir iterasyonda bir global komşu değeri vardır. Yerel komşular ise ilk iterasyonda her bir parçacığın aynı zamanda pozisyon değerleridir ve her iterasyonda parçacık sayısı kadar yerel en iyi ortaya çıkar. Her bir parçacık, global ve yerel komşu değerlerini kullanarak bir önceki iterasyonda en iyi değere sahip parçacığın pozisyonuna doğru hareket eder. Durdurma kriteri genellikle parçacıklardan en az birinin optimuma ulaşması yada iterasyon sayısıdır. PSO'da parçacıklardan en az birinin optimuma ulaşması beklenir. Başka bir deyişle, optimum sonuca her parçacığın ulaşması beklenmez, önemli olan sürü içerisinden bir parçacığın optimuma ulaşmasıdır. Kısacası, önemli olan sürünün başarısıdır.

3.2. PSO'nun Genel Yapısı

Genel PSO algoritmasının sahte (pseudo) kodu Şekil 3.1'de gösterilmektedir. PSO kodundan da anlaşıldığı gibi, algoritmada ilk olarak kullanılacak parametreler belirlenir. İkinci olarak popülasyon değerleri rasgele belirlendikten sonra bu değerlere karşılık gelen amaç fonksiyon değeri hesaplanır. Başlangıç çözümündeki her bireye ait çözüm aynı zamanda o parçacığın yerel en iyi değeridir. Amaç fonksiyon değeri, en iyi olan parçacık global en iyi olarak atanır ve bu parçacık değerleri global komşular olarak saklanır. Aynı şekilde başlangıçtaki her bir parçacığın amaç fonksiyon değerleri yerel en iyiler olarak saklanıp parçacık değerleri de yerel komşular olarak saklanır. Genellikle başlangıç çözümünde başlangıç hızları rasgele belirlenmektedir.

```
Parametreleri Belirle
Başlangıç Popülasyonunu Oluştur
Herbir Parçaçık İçin{
   Amaç Fonksiyon Değerlerini Bul
   Yerel En İyiyi Bul
   }
Global En İyiyi Bul

Yap{
   Herbir Parçaçık İçin{
     Hız Vektörünü Güncelle
     Pozisyon Vektörünü Güncelle
     Amaç Fonksiyon Değerlerlerini Bul
   Yerel En İyiyi Güncelle
   }
  Global En İyiyi Güncelle
Durdurma Kriteri
```

Şekil 3.1. PSO sahte kodu

Başlangıç çözümünden sonra PSO algoritması tam anlamıyla başlar. Başlangıç çözümünde elde edilen hız değerlerinden yararlanarak yeni pozisyonlar belirlenir. Yeni bulunan pozisyonlar için amaç fonksiyonu değerleri hesaplanır ve hızlar güncellenir. Daha sonra her parçacık için o iterasyona kadar bulunan en iyi amaç fonksiyonu değerini veren çözüm yerel en iyi olarak atanır. Tüm parçacıklar için yerel en iyiler bulunarak yerel en iyi komşular kümesi oluşturulur. Her iterasyonda bulunan global en iyiler karşılaştırılarak en iyi amaç fonksiyonuna sahip parçacığın çözümü global en iyi olarak atanır. Bu işlem bir durdurma kriterine kadar devam eder.

3.3. PSO Terimleri

PSO'da kullanılan terimler aşağıdaki gibi açıklanmaktadır.

Pozisyon Vektörü (Particle): Pozisyon vektörü X_i^k ile gösterilir. Diyelim ki problem boyutu (*j*) kadar parçacıktan meydana gelsin. Her parçacık bir çözümü ifade etmek üzere;

$X_i^k = [x_{i1}^k, x_{i2}^k, ..., x_{ij}^k]$ pozisyon vektörü olarak tanımlanır.

Burada yer alan x_{i1}^k *k*. iterasyonda *i*. parçacığının pozisyon vektöründeki birinci elemanını gösterir.

Hız Vektörü (Velocity): Hız, parçacığın bir sonraki pozisyonunu başka bir ifadeyle yeni çözümü bulmada etkili olan en önemli faktördür. Hız vektörü V_i^k ile gösterilir. Parçacığın hız vektörü;

$V_i^k = [v_{i1}^k, v_{i2}^k, ..., v_{ij}^k]$ şeklinde ifade edilir.

Burada yer alan v_{i1}^k *k*. iterasyonda *i*. parçacığın hız vektöründeki birinci elemanını göstermektedir.

Yerel En İyi Değer (Local Best): Yerel en iyi, bir parçacığın o ana kadar elde edilmiş en iyi amaç fonksiyon değerine sahip pozisyonudur yani çözümüdür. Sürüde bulunan parçacık sayısı kadar her iterasyonda yerel en iyi bulunur. Amaç fonksiyonu yerine uygunluk fonksiyonları da kullanılabilir.

Yerel En İyi Komşular (Local Best Neighbors): Her iterasyonda parçacık sayısı kadar elemandan oluşan bir vektördür. Bu vektör, yerel en iyi değerlere sahip pozisyon vektörü olarak tanımlanır. Bir sürüdeki *i.* parçacığın yerel en iyi komşuları;

$P_i = [p_{i1}, p_{i2}, ..., p_{ij}]$ şeklinde ifade edilir.

Buradaki p_{i1} *i*. yerel en iyi vektörünün 1. değerini göstermektedir.

Global En İyi Değer (Global Best): Elde edilen en iyi amaç fonksiyonu yada uygunluk fonksiyonu değeri ve bu değere sahip parçacığın çözümüdür.

Global En İyi Komşular (Global Best Neighbors): Her iterasyonda bulunan global en iyi değerlere sahip pozisyon değerleridir. Bir çözüm uzayında en fazla iterasyon sayısı kadar global en iyi hesaplanır.

$$G = [g_1, g_2, ..., g_j]$$

Buradaki g_1 global en iyi vektörünün 1. değerini göstermektedir.

Parçacığın tecrübesi (c_1): Sosyal parametre olarak da adlandırılır. Parçacığın önceki iterasyonlarda elde ettiği en iyi sonuçların, bir sonraki iterasyona etkisidir.

Sürünün tecrübesi (c_2): Kavramsal parametre olarak da adlandırılır. Sürüde önceki iterasyonlarda en iyi sonuca sahip parçacığın sonuçlarının bir sonraki iterasyona etkisidir.

Atalet Ağırlığı (Inertia Weight): Bir önceki hız vektörünün mevcut hız vektörü üzerindeki etkisini kontrol eden parametreye atalet ağırlığı denir ve w şeklinde gösterilir. Atalet ağırlığının parçacıkta global ve yerel araştırma kabiliyeti arasında değişim etkisi vardır. Eğer atalet ağırlığı yüksek bir değer alınırsa parçacık global bir şekilde arama gerçekleştirir, bu değerin küçük alınması ise yerel bir arama gerçekleştirir [Kennedy ve ark., 2001]. Hızın optimal çözümün bulunmasında, parçacıkların optimalin bulunduğu pozisyonu aşmalarını yada optimale yaklaşmada geç kalmayı önlemek amacıyla bazı problemlerde önemli bir rolü olabilmektedir.

PSO'nun gelişmiş özellikleri [Merkle ve Middendorf, 2005]:

- Uyarlamalı (adaptive) PSO. Yürütme süresi içinde otomatik olarak uyarlanabilen PSO algoritmasının parametrelerinin en büyük değerlerinde öneride bulunulmuştur.

- En iyi komşuluk hız değerlerinin güncellenmesi. Farklı PSO algoritmaları parçacıklar arasında komşuluk ilişkileri kurar. Diğer bir ifade ile, her parçacığın hız

güncellemesi için global en iyiyi kullanmak yerine, onun komşularındaki en iyi pozisyon kullanılır. Kennedy ve Mendes [2003] parçacığın komşularının onun hızında etkili olduğunu göstermişlerdir. Aşağıdaki formül i parçacığının $d \in [1,...,D]$ boyutta N_i komşulu hız güncellemesini açıklar:

$$v_{id} = w.v_{id} + \sum_{j \in N_i} \frac{c.r_j.(p_{jd} - x_{id})}{|N_i|} \tag{3.1}$$

- Maksimum hız. Hız değerlerinin $[-V_{max};V_{max}]$ arasında sınırlandırılması mümkündür.

- Kraliçe (queen) parçacık. Clerc [1999] kraliçe parçacığı her zaman sürünün gerçek ağırlık merkezine yerleşen bir parçacık olarak tanımlar. Sürünün ağırlık merkezi mümkün optimuma yakın olabileceğinden amaç fonksiyonunu bu konumda değerlendirmek makuldür. Deneysel sonuçlar kraliçe parçacığın fonksiyonun tipine göre avantaj sağlayabileceğini göstermektedir.

- Yakınsama zorlaması. PSO'nun yakınsama oranının geliştirilmesini ele alan çeşitli çalışmalar vardır. Bunlardan en sık kullanılanı daralma faktörüdür. Daralma faktörü K, Clerc ve Kennedy [2002] tarafından ortaya konmuştur. Aşırı geribildirim etkisinin azaltılması amaçlanmaktadır. Eş.3.2'de görüldüğü gibi yeni hız değeri bulunurken güncellenen hız değeri K ile çarpılarak belirli bir değere yakınsaması sağlanır. Daralma faktörünün (K) bulunması için önerilen formül Eş.3.3'te verilmektedir.

$$v_{id} = K(v_{id} + c_1 r_1(p_{id} - x_{id}) + c_2 r_2(p_{gd} - x_{id})) \tag{3.2}$$

$$K = \frac{2}{\left|2 - c - \sqrt{c^2 - 4c}\right|} \qquad c = c_1 + c_2 \qquad c>4 \tag{3.3}$$

3.4. PSO'nun Genel Mekanizması

Parçacık sayısı, atalet ağırlığı, c_1, c_2 başlangıç değeri gibi parametreleri belirlenen algoritma, başlangıç iterasyonunda rastgele oluşturulur. Pozisyon ve hız vektörünün belirlenen aralıklar çerçevesinde değer alması sağlanır. Örneğin başlangıç hız vektörü değerleri [$-V_{max}$, V_{max}] arasında, pozisyon vektörü değerleri de [$-d_{max}$, d_{max}] arasında rassal bir şekilde oluşturulur. Oluşturulan pozisyon vektör değerlerine karşılık gelen amaç fonksiyon değerleri bulunur. Daha sonra bu amaç fonksiyon değerlerine göre yerel en iyiler ile global en iyi değerleri bulunur. Bundan sonraki adım ise hız vektörünün güncelleştirilmesidir. Hız vektörünün güncelleştirilmesi şu şekilde yapılır.

Öncelikle atalet ağırlığı belli bir sabit değerle (β) çarpılarak, her bir iterasyonda azaltılır. Azaltma fonksiyonu Eş.3.4'teki gibidir:

$$w^{k+1} = w^k \times \beta \tag{3.4}$$

Her bir parçacığın yeni hız vektörü değerini bulmak için Eş.3.5.'ten faydalanılır:

$$v_{ij}^{k+1} = (w^{k+1} \times v_{ij}^k) + c_1 \times r_1 \times (p_{ij} - x_{ij}^k) + c_2 \times r_2 \times (g_j - x_{ij}^k) \tag{3.5}$$

Öncelikle azaltılan atalet ağırlığı değeri (w), parçacığın mevcut hız vektörü değeri ile çarpılır. Daha sonra eşitliğin ikinci bölümünde, yerel komşu değeri ve parçacığın pozisyon değeri arasındaki fark c_1 ve r_1 ile çarpılarak parçacığın kendi tecrübesinden yararlanılır. Eşitliğin üçüncü bölümünde ise global komşu değeri ve pozisyon değeri arasındaki fark c_2 ve r_2 ile çarpılarak sürünün tecrübesinden yararlanılır. Nihayetinde bulunan değerler toplanarak yeni hız vektörü değeri bulunur. Burada yer alan r_1 ve r_2 0 ile 1 arasında düzgün dağılmış rassal değerleri, c_1 ve c_2 ise sosyal ve kavramsal parametre değerlerini ifade eder.

Yeni oluşan hız vektör değeri, başlangıçta belirtilen aralıklar arasında kalması için *h* fonksiyonu ile kısıtlanır. Bu fonksiyon Eş.3.6'da verilmektedir:

$$h(v_{ij}^k) = \begin{cases} V_{\max}, & if \quad v_{ij}^k > V_{\max} \\ v_{ij}^k, & if \quad \left|v_{ij}^k\right| \le V_{\max} \\ -V_{\max}, & if \quad v_{ij}^k < -V_{\max} \end{cases} \tag{3.6}$$

Daha sonra yeni pozisyon vektörü, Eş.3.7'de ifade edildiği gibi, elde edilen hız vektör değerleri ile bir önceki pozisyon vektörünün toplamıyla güncelleştirilir. Böylece parçacığın yeni pozisyonu belirlenir.

$$x_{ij}^{k+1} = x_{ij}^k + v_{ij}^{k+1} \tag{3.7}$$

Yeni oluşan bu pozisyon vektörü yardımıyla amaç fonksiyon değerleri, global en iyi ve komşuları ile yerel en iyi ve komşuları güncelleştirilir. Bu işlemler önceden belirlenen iterasyon sayısına ya da maksimum işlem zamanına ulaşılıncaya kadar devam eder.

3.4.1. PSO ile sürekli optimizasyon

Genel olarak bakıldığında, PSO ile ilgili ilk çalışmalar ve daha sonra yapılan çalışmalarda sürekli fonksiyonların optimizasyonu amaçlandığı görülmektedir. Yukarıda anlatılan PSO'nun genel işleyişi sürekli optimizasyona yöneliktir.

PSO'yu test etmek için kullanılan test problemleri de sürekli fonksiyonlardan oluşmaktadır. Örneğin i=1,...,D boyutlu Rastrigrin test problemi aşağıdaki gibidir [Shi ve Eberhart, 1999]:

$$f(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$$

3.4.2. PSO ile kesikli optimizasyon

Gerçek dünya problemlerinin çoğunun kesikli olaylardan oluştuğu düşünüldüğünde başlangıçta sürekli fonksiyonların optimizasyonu için geliştirilen PSO'nun kesikli 0-1 versiyonu geliştirilmiştir [Kennedy ve Eberhart, 1997]. Önerilen versiyonun genel işleyişi aşağıdaki gibidir:

Başlangıç popülasyonunun oluşturulması

Başlangıç pozisyonu değerleri rassal olarak tespit edilir.

Eğer $U(0,1) > 0.5$ ise $x_{id}^0 = 1$ değilse $x_{id}^0 = 0$

Hız değerleri yine belirlenen [$-V_{max}$,V_{max}] aralığında rassal olarak belirlenmektedir.

$$v_{id}^0 = V_{min} + (V_{max} - V_{min}) * rand$$

Daha sonra yerel en iyi ve global en iyi bulunur.

Yeni Çözümlerin Bulunması

Burada elde edilen hız değerlerinden, 0 ve 1 arasında değerler elde etmek için 3 tip transfer fonksiyonu tanımlanmaktadır:

1. Exp fonksiyonu: $$exp(v_{id}^k) = \frac{1}{exp(-v_{id}^k)}$$

2. sigmoid fonksiyon: $$sigmoid(v_{id}^k) = \frac{1}{1 + e^{-v_{id}^k}}$$

3. tanh fonksiyonu $$tanh(v_{id}^k) = \frac{1 - exp(-v_{id}^k)}{1 + exp(-v_{id}^k)}$$

Yapılan testler sonucunda sigmoid fonksiyonun diğerlerine nazaran daha etkili olduğu bulunmuştur [Xu ve ark., 2006]. Eğer sigmoid fonksiyondan elde edilen 0-1 arasındaki değer, rassal değerden küçükse pozisyon 1'dir, diğer durumda 0'dır.

$$x_{id}^{k} = \begin{cases} 1, & if \quad U(0,1) < sigmoid(v_{id}^{k}) \\ 0, & dd. \end{cases} \tag{3.8}$$

Daha sonra hızdaki değişim hesaplanır.

$$\Delta v_{id}^{k-1} = c_1 * r_1 * (pb_{id}^{k-1} - x_{id}^{k-1}) + c_2 * r_2 * (gb_{id}^{k-1} - x_{id}^{k-1}) \tag{3.9}$$

Hızdaki değişim mevcut hız değeri ile toplanarak sınırlandırılmış fonksiyondan geçirilir. Hız değerlerini sınırlandırmak için Eş.3.6'dan faydalanılır. Böylece yeni hız değeri bulunur.

$$v_{id}^{k} = h(v_{id}^{k-1} + \Delta v_{id}^{k-1}) \tag{3.10}$$

Daha sonra yerel en iyiler ve global en iyi bulunur. Bu süreç durdurma kriteri sağlanana kadar sürer.

3.5. PSO İçin Literatür Araştırması

PSO, kuş ve balık sürülerinin iki boyutlu hareketlerinden esinlenerek ilk olarak 1995'te [Kennedy ve Eberhart, 1995] ve 1997 yıllarında [Kennedy ve Eberhart, 1997] geliştirilmiştir. Daha sonra 1998 ve 1999'da Shi ve Eberhart, 2001' de Kennedy ve ark. tarafından incelenmiştir.

Kennedy ve Eberhart'ın [1995] çalışmasında PSO metodolojisi kullanılarak doğrusal olmayan fonksiyonların optimizasyonu için bir konsept tanıtılmakta ve birkaç paradigmanın değerlendirilmesi yapılmaktadır. Paradigmaların karşılaştırma testleri

tanımlanmakta ve doğrusal olmayan fonksiyon optimizasyonu ve network eğitimi içeren uygulamalar sunulmaktadır. PSO ile hem yapay zeka hem de genetik algoritma arasındaki ilişkiler açıklanmaktadır.

Eberhart ve Kennedy'nin [1995] çalışmasında yeni bir optimize edici olan PSO metodoloji kullanılarak doğrusal olmayan fonksiyonların optimizasyonu açıklanmaktadır. Global ve yerel en iyi kavramları tartışılmaktadır. Her iki kavram, test problemleriyle, sinir ağları eğitimi ve robot görev öğrenmesini içeren uygulamalarda sunulmaktadır. PSO ile yapay zeka ve evrimsel hesaplama arasındaki ilişkiler gözden geçirilmektedir.

Kennedy ve Eberhart [1997] PSO'yu ortaya atan yazarlar olarak, kesikli 0-1 versiyonu ele almaktadır. Sürekli uzayda etkili olan PSO'nun kesikli 0–1 değişkenler için tekrar tartışıldığı görülmektedir.

Shi ve Eberhart'ın [1998] çalışmasında, orijinal parçacık sürü optimize ediciye atalet ağırlığı olarak adlandırılan yeni bir parametre tanıtılmaktadır. Simülasyonlar bu yeni parametrenin PSO üzerinde anlamlı ve etkili olduğunu göstermektir.

Shi ve Eberhart'ın [1999] çalışmasında, PSO'nun performansı deneysel olarak denenmektedir. Asimetrik başlangıç aralık ayarları ile dört farklı karşılaştırma fonksiyonu test fonksiyonu olarak seçilmiştir. Deney sonuçları PSO'nun avantaj ve dezavantajlarını göstermektedir. Tüm test durumlarında, PSO her zaman çok hızlı bir şekilde optimal pozisyonlara doğru yakınsamaktadır fakat minimuma yaklaştıkça yakınsama hızı yavaşlamaktadır. Buna rağmen, deney sonuçları uygun bir atalet ağırlığı ile PSO'nun iyi bir performans verdiğini göstermektedir.

Özcan ve Mohan'ın [1999] çalışmasında PSO'nun bir optimizasyon metodu olarak doğrusal olmayan fonksiyonlarda ve sinir ağı eğitiminde başarıyla kullanıldığı anlatılmaktadır. Basit tek boyutlu PSO sistemindeki bir parça, sinüzoidal bir dalga ile tanımlanan bir yol takip eder. Çalışmada, çok boyutlu arama uzayında parçacıkların yörüngesi için yakın form eşitlikleri elde etmek amaçlanmaktadır.

Salman ve arkadaşlarının [2002] çalışmasında PSO prensiplerini esas alan yeni bir görev atama algoritması sunulmaktadır. Görev atama problemlerine PSO arama stratejisinin uyumu ve uygulaması tartışılmaktadır. Atama problemi için önerilen PSO algoritması Genetik Algoritma ile karşılaştırılmaktadır. Simülasyon sonuçlarına göre PSO'yu esas alan algoritma görev atama problemleri için daha geçerli bir yaklaşımdır.

Laskari ve arkadaşlarının [2002] çalışmasında tamsayılı programlama problemlerinde PSO metodunun performansı araştırılmaktadır. Üç farklı PSO, dal sınır tekniği kullanılarak karşılaştırılmaktadır.

Taşgetiren ve Liang'ın [2003] çalışmasında parti büyüklüğü problemi için 0-1 PSO algoritması sunulmaktadır. Ele alınan problem toplam sipariş verme ve elde tutma maliyetlerini minimize edecek sipariş miktarının bulunmasıdır. Test problemleri Wagner-Whitin algoritması ile çözülerek optimal sonuçlar bulunduktan sonra problem PSO ve geleneksel Genetik Algoritma ile kodlanarak karşılaştırılmıştır. Deneysel sonuçlar, 0-1 PSO algoritmasının hemen bütün sonuçlarda optimal çözüme ulaştığını göstermektedir.

Wang ve arkadaşlarının [2003] çalışmasında TSP için PSO'nun yeni bir uygulaması önerilmektedir. Değiş-tokuş (Swap) operatörü ve değiş-tokuş sıralama kavramları sunulmaktadır. Değiş-tokuş sıralama birden fazla değiş-tokuş operatörü kullanılması demektir.

Shi ve arkadaşlarının [2004] çalışmasında genelleştirilmiş gezgin satıcı problemi için yeni bir kesikli PSO metodu sunulmaktadır.

Jerald ve arkadaşlarının [2005] çalışmasında optimum çizelgeleme üretmek için farklı çizelgeleme mekanizmaları tasarlanmıştır. Bu mekanizmalar, Genetik Algoritma, Tavlama Benzetimi, Memetik ve PSO algoritmaları gibi geleneksel

olmayan yaklaşımlar içermektedir. Makine boş zamanı ve toplam ceza maliyetini minimize etmek amaçlanmaktadır. Kullanılan algoritmalar karşılaştırılmıştır.

Andres ve Lozano'nun [2006] çalışmasında PSO algoritması ilk kez hücre oluşturma problemi için kullanılmaktadır. Hücredeki makineler gruplanırken hücreler arası hareketlerin minimizasyonu esas alınmıştır. Örnek problemler üzerindeki deneyler PSO algoritmasının optimal sonucu bulmada etkili olduğunu göstermektedir.

Zhu ve arkadaşlarının [2006] çalışmasında PSO prensipleriyle çalışan zaman pencereli araç rotalama problemi (VRPTW) algoritması sunulmaktadır. VRPTW'ye PSO arama stratejisinin uyumu ve uygulaması tartışılmaktadır. Deneysel sonuçlar PSO'nun VRPTW'ye etkili optimal çözümler bulduğunu göstermektedir.

Şevkli ve Yenisey'in [2006] çalışmasında, atölye tipi çizelgeleme problemlerinin çözümü için bir PSO modeli değişken komşuluk arama yöntemi ile birlikte geliştirilmiştir. Oluşturulan bu model, tamamlanma zamanı performans ölçütüne göre literatürde yer alan bazı zor test problemleri üzerindeki sonuçları incelenmiş ve iyi sonuçlar veren diğer sezgisel yöntemlerin sonuçlarıyla karşılaştırılmıştır. Sonuçta genel olarak önerilen modelin diğer yöntemlere göre daha iyi veya eşdeğer seviyede olduğu görülmüştür.

Zeng ve arkadaşlarının [2006] çalışmasında silah-hedef atama problemi ele alınmaktadır. Bu problemi çözmek için kesikli PSO önerilmektedir. Modelde daha iyi aday çözümler bulmak için Açgözlü (Greedy) Arama Stratejisi kullanılmaktadır.

Xu ve arkadaşlarının [2006] çalışmasında 3 alt-sürü kesikli parçacık sürü optimizasyon algoritması (THSDPSO) önerilmektedir. Bütün parçacıklar 3 alt-sürüye bölünmektedir. Bir alt-sürü global en iyi pozisyona doğru uçmaktadır. İkinci alt-sürü ters yöne uçmaktadır. Son alt-sürü global en iyi pozisyon etrafında rassal olarak uçar. THSDPSO algoritmasında parçacıkların pozisyonunu elde etmek için iki yol kullanılır. Bir yol, transfer fonksiyonu ile bir olasılık ölçüsü olarak uygun hızı

kullanmakta ve buna BTHSDPSO denmektedir. Diğeri, direk olarak hard limit fonksiyonunu kullanmakta ve buna da HTHSDPSO denmektedir.

Taşgetiren ve arkadaşlarının [2007] çalışmasında, çevrim zamanı ve işlerin maksimum gecikmelerini minimize etmek amacıyla permütasyon akış tipi sıralama problemini çözmek için PSO algoritması sunulmaktadır. Ayrıca PSO algoritmasına, değişken komşuluk arama (variable neighborhood search) metodunu esas alan basit fakat çok etkili bir yerel arama eklenmiştir. Önerilen PSO algoritması çevrim zamanı için 40 adet test problemine, maksimum gecikme kriteri için 160 adet örnek probleme uygulanmıştır.

PSO algoritmasının uygulandığı diğer başarılı çalışmalar ise şunlardır: Araç rotalama problemi [Ai ve Kachitvichyanukul, 2009], sinir ağları eğitimi [Bergh, ve Engelbecht, 2000], akış tipi üretim planlama problemi [Taşgetiren ve ark., 2004; Liao ve ark., 2007; Lian ve ark., 2008; Liu ve ark., 2008; Pan ver ark., 2008; Zobolas ve ark., 2009], en kısa yol problemi [Mohemmed ve ark., 2008], atölye tipi planlama problemi [Xia ve Wu, 2005; Sha ve Hsu, 2006; Lian ve ark., 2006; Gao ve ark., 2008], depo yerleşim tasarımı [Önüt ve ark., 2008].

Yapılan literatür araştırmasında PSO algoritmasının ne geleneksel düz MHDP'ne nede U tipi MHDP'ne uygulamasına rastlanmamıştır. Bu nedenle bu çalışmada, SUMHDP-1'in çözümünde PSO algoritması önerilmiştir.

4. PSO İLE STOKASTİK U-TİPİ HAT DENGELEME

Bu çalışmada, literatürde çok fazla çalışma olmaması sebebiyle, U-tipi hatlarda görev zamanlarının stokastik olması problemi ele alınmaktadır. Bu problemi çözmek için ise bir meta-sezgisel olan PSO kullanılmaktadır.

Bu bölümde öncelikle SUMHDP-1 ele alındıktan sonra, bu problemi çözmek için uyarlanmış PSO algoritması anlatılacak ve algoritmanın çalışması sayısal örnek bir problem üzerinde açıklanacaktır. Müteakiben, literatürde yer alan problemler için hesaplama sonuçları ve bu sonuçlara ait yorumlar verilecektir. Son olarak, PSO algoritmasında bulunan parametrelerin, optimal çözümün bulunmasındaki etkileri incelenecektir.

4.1. Stokastik U-tipi Hat Dengeleme Problemi

Bir montaj hattında çeşitli sebeplerden dolayı görev zamanları değişkenlik gösterebilmektedir. Bu sebepleri şöyle sıralamak mümkündür:

- operatörün yorgunluğu,
- dikkatin dağılması,
- düşük beceri seviyesi,
- iş tatminsizliği,
- bakımsız malzeme/makine,
- hatalı hammadde,
- hatalı montajın bir sonraki aşamaya iletilmesi,
- farklı öğrenme kabiliyetleri,
- operatörlerin fiziksel farklılıkları.

U-tipi hat dengeleme probleminde görev zamanları stokastik olduğu zaman stokastik U-tipi hat dengelemeden bahsedilir. Genellikle görev zamanlarının normal dağılıma uyduğu kabul edilir. N adet görevli bir montaj hattında j inci görevin (j=1,…,N)

görev süresi, ortalaması μ_j ve varyansı σ_j^2 ile gösterilen bağımsız rassal bir değişkendir. Genellikle, bir görevin süresine ait varyans o görevin zorluğuyla doğru orantılıdır. Görevler, verilen tamamlanma olasılığı ile çevrim zamanını (C) aşmayacak şekilde, verilen öncelik ilişkilerini dikkate alarak (k=1,...,K) istasyonlara atanır. SUMHDP-1 probleminde amaç, verilen bir çevrim süresi için istasyon sayısını minimum yapmaktır. Bir istasyonun toplam süresi $(1-\alpha)$ olasılıkla C'yi aşamaz (Eş.4.1).

$$P\left\{\sum_{j\in W_k}\mu_j + Z_{1-\alpha}\sqrt{\sum_{j\in W_k}\sigma_j^2} < C\right\} \geq 1-\alpha \qquad \text{for } k=1,...,K \tag{4.1}$$

W_k : k istasyonuna atanan görevlerin kümesi,

$Z_{1-\alpha}$: Birikimli standart normal dağılımın $1-\alpha$ olasılığı için tablo değeridir.

Eğer bir çözümdeki istasyon sayısı teorik alt sınıra (LB: Lower Bound) eşitse teorik olarak en iyi sonuç bulunmuş demektir. Urban ve Chiang [2006] minimum istasyon sayısı için teorik alt sınırı aşağıdaki gibi elde etmiştir:

$$LB = \left\lceil \frac{\sum_{j=1}^{N}\mu_j + Z_{1-\alpha}\sqrt{\sum_{j=1}^{N}\sigma_j^2}}{C} \right\rceil \tag{4.2}$$

burada $\lceil x \rceil$, x'e eşit yada x'den büyük en küçük tamsayıdır.

Geleneksel düz hatlarda, istasyona atanacak görevler, öncelik ilişkilerini gösteren diyagramda ileri doğru başka bir deyişle öncülleri atanmış görevlerden seçilirken; U-tipi hatlarda istasyona atanacak görevler hem ileri, hem geri hem de eşzamanlı olarak her iki yönde başka bir deyişle, hem öncülleri hem de ardılları atanmış görevlerden seçilir.

SUMHDP-1 problemi ile ilgili olarak aşağıda belirtilen kabuller yapılmıştır:

- Her bir görev sadece bir istasyona atanabilir.
- İstasyon toplam zamanı çevrim süresini aşamaz.
- Görev zamanlarının her biri ortalaması ve varyansı bilinen normal dağılıma uyar.
- Öncelik ilişkileri ihlal edilemez.

4.2. SUMHDP-1 İçin PSO Algoritması

Bu çalışmada, SUMHDP-1 problemlerinin çözümü için PSO algoritması önerilmektedir. Önerilen PSO algoritmasında, literatürde kullanılan PSO terimleri SUMHDP-1 problemine uyarlanmıştır:

X_{ij}^{t} : *t*'inci iterasyonda *i*'inci parçacığın *j*'inci görevi hangi istasyona atadığını gösterir. Bir iterasyonda bir parçacık için görev-istasyon atama vektörü kullanılır. Örnek olarak $X_{23}^{1} = 4$ 1. iterasyonda 2. parçacığın 3. görevi 4. istasyona atadığını gösterir.

V_{ij}^{t} : *t*'inci iterasyonda *i*'inci parçacığın *j*'inci görevi bir istasyona atama isteğidir. Hız değeri $V_{\min}$ ve $V_{\max}$ arasında sınırlandırılmıştır.

Pbest(*i*): *i*'inci parçacığın o zamana kadar elde ettiği en iyi uygunluk fonksiyonu değerini veren görev- istasyon atama vektörüdür.

$$Pbest(i) = [p_{i1}, p_{i2}, ..., p_{ij}]$$

$$p_{ij} = [k_{i1}, k_{i2}, ..., k_{iN}]$$

Gbest(t): t'inci iterasyondaki parçacıkların arasında en iyi (minimum) uygunluk fonksiyonu değerini veren görev-istasyon atama vektörüdür.

Gbest : Tüm iterasyonlar arasında en iyi uygunluk fonksiyonu değerini veren Gbest(t) görev-istasyon atama vektörüdür.

PSO algoritmasında kullanılmak üzere Bautista ve ark. [2000] önerdiği uygunluk fonksiyonu kullanılmıştır. i'inci parçacığın uygunluk fonksiyonu $f(i)$ aşağıdaki gibi bulunur.

$$f(i) = (NE_i - LB) + \left(\frac{\sqrt{\sum_{k=1}^{NE_i} (c_k - C)^2}}{C\sqrt{NE_i}} \right) \tag{4.3}$$

NE_i : i'inci çözümdeki istasyon sayısı,
C : çevrim zamanı,
LB : minimum istasyon sayısı için alt sınır,
c_k : k'ıncı istasyonun toplam zamanıdır.

NE_i alt sınıra yaklaştıkça uygunluk fonksiyonu değeri azalacağından, uygunluk fonksiyonu değerinin minimum olması tercih edilir.

Önerilen algoritmanın ana aşamaları aşağıdaki gibidir:

- Tanımlamalar,
- Başlangıç çözümünün oluşturulması,
- PSO'nun işletilmesi.

4.2.1. Tanımlamalar

Bu aşamada, algoritmada yer alan parametre ve değişkenler tanımlanmaktadır. Başka bir deyişle bu parametre ve değişkenlerin algoritmaya dışarıdan girilmesi gerekmektedir. Bunlar;

- iterasyon sayısı,	- parçacık sayısı,	- çevrim zamanı,
- atalet ağırlığı,	- c_1, c_2,	- V_{min}, V_{max},
- varyans tipi,	- Z değeri.	

4.2.2. Başlangıç çözümünün bulunması

PSO algoritması bir başlangıç çözümüne ihtiyaç duyar. SUMHDP-1'in başlangıç çözümünün oluşturulması için Şekil 4.1'de sunulan sahte kod yazılmıştır.

Şekil 4.1. SUMHD-1 için PSO başlangıç çözümü sahte kodu

Başlangıç çözümünün oluşturulması için önerilen kod, öncelikle istasyona atanacak aday görevlerin belirlenmesi ile başlar. Bu maksatla öncülü ve ardılı olmayan görevler bir kümede toplanır ve aralarından biri rasgele seçilerek bir istasyona atanır. Bir parçacık için tüm görevler çevrim zamanını aşmayacak şekilde bir istasyona atanana kadar bu işlem sürer.

Her bir parçacığın uygunluk fonksiyonları hesaplanır. Sonra parçacığın Pbest(*i*) atanır. Sürüdeki en iyi uygunluk fonksiyonuna ait parçacığın çözümü ise o iterasyon için global en iyi olarak atanır. Başlangıç çözümünün sonunda her parçacık için her göreve ait bir başlangıç hız değeri rassal olarak belirlenir.

Başlangıç çözümü için önerilen algortimaya ait akış şeması ise Şekil 4.2'de verilmektedir.

Başlangıç hızlarının bulunması

Bu çalışmada, bir parçacığın hızı, bir görevin bir istasyona atanma isteğini ifade etmektedir. İstasyona atanacak görev belirlenirken öncülü ve ardılı olmayan görevler bir kümede toplandıktan sonra sahip oldukları hız değerlerine göre seçilecektir. Aday görevler arasından seçim yapılırken mutlak değerce en büyük değere sahip görev tercih edilecektir. Bunun sebebi, mutlak değerce en büyük hıza sahip görev yerel ve global en iyiye sahip parçacıklardan en uzakta olmasıdır. Böylece, o görev istasyona atanarak yeni bir çözüm denenmiş olacaktır.

Başlangıç hızları, minimum ve maksimum hız değerleri arasında rasgele belirlenir (Eş.4.4).

$$V_{ij}^{t=0} = V_{\min} + rand() * (V_{\max} - V_{\min}) \tag{4.4}$$

Şekil 4.2. SUMHDP-1 için PSO başlangıç çözümü akış şeması

4.2.3. PSO'nun işletilmesi

Başlangıç çözümler ve başlangıç hızlar bulunduktan sonra SUMHDP-1'in çözümü için PSO algoritması tam anlamıyla Şekil 4.3'te verilen sahte koda göre çalıştırılır.

Şekil 4.3. SUMHDP-1 için PSO algoritması sahte kodu

PSO algoritması uygulanırken istasyona atanacak aday görevler, öncülü ve ardılı olmayan görevler olarak belirlenir. Bu aday görevler arasından mutlak değerce en büyük hız değerine sahip görev, istasyona atama yapmak için seçilir.

Her parçacık için, çevrim zamanını aşmayacak şekilde tüm görevler bir istasyona atanır. Her bir parçacığın uygunluk fonksiyonları hesaplanır. Sonra parçacığın o iterasyona kadar elde ettiği en iyi uygunluk fonksiyonu değeri $f(i)$ bulunur ve Pbest(i) olarak atanır. Sürüdeki en iyi uygunluk fonksiyonuna ait parçacığın çözümü ise o iterasyon için global en iyi olarak atanır. Bulunan Gbest(t) daha önceki iterasyonların global en iyilerine ait uygunluk fonksiyonu değerleriyle karşılaştırılarak Gbest atanır.

Hızlar güncellenir. Bu süreç devam ederken, eğer atama yapılan son istasyon sayısı (NE_i) alt sınıra eşit olursa algoritma durdurulur. Aksi takdirde, algoritma iterasyon sayısınca çalışır.

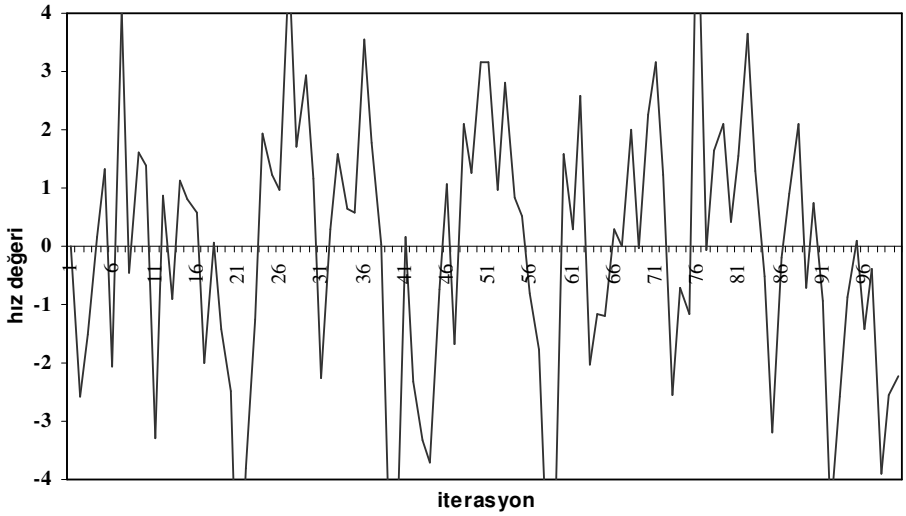
Hızların güncellenmesi

Yeni hızlar, bir parçacığın yeni iterasyondaki çözümlerde bir görevi bir istasyona atama isteğini belirleyecektir. Yeni hızlar belirlenirken daha önceden belirlenmiş hız sınırları içinde olması sağlanır. i'inci parçacığın j'inci görev için yeni hızı bulunurken, o görevin Pbest(i) çözümünde atandığı istasyon P_{ij}, mevcut çözümde atandığı istasyon X_{ij}^{t} ve Gbest çözümünde atandığı istasyon G_{ij} olarak ifade edilir. Eski hız değeri atalet ağırlığı ile çarpılıp, parçacığın kendi tecrübesinden ve sürüdeki en iyi uygunluk fonksiyonuna sahip parçacığın tecrübesinden yararlanarak yeni hız hesaplanır. Ancak burada problemin yapısı gereği hesaplanan bu yeni hız değeri maksimum olması istenen hız değerine bölünerek hızın birikimli olarak pozitif yada negatif olarak artmasının engellenmesi amaçlanmaktadır. Yeni hızlar Eş.4.5'teki gibi hesaplanır.

$$V_{ij}^{t+1} = [w * V_{ij}^{t} + c_1 * r_1 * (P_{ij} - X_{ij}^{t}) + c_2 * r_2 * (G_{ij} - X_{ij}^{t})] / V_{\max} \qquad (4.5)$$

Hızların güncellenmesinde yeni hesaplanan hız, bir $V_{\max}$ değerine bölünmediği takdirde hızın pozitif yada negatif olarak sürekli arttığı görülmüştür. Her ne kadar hız $V_{\min}$ ve $V_{\max}$ arasında sınırlandırılmış ise de hızın yeni seçilecek görevi belirlerken bir

döngüye girmesinin önlenmesi amacıyla V_{max} değerine bölünmesi faydalıdır. V_{max} değerinin özellikle alt sınırın yarısı ve alt sınır arasında olması durumunda hızın aşırı değerlere gitmesini engellediği görülmüştür. Bu durumu Şekil 4.4'te görmek mümkündür. Şekil 4.4'te alt sınırın 8 olduğu bir problem için [-4, 4] aralığında sınırlandırılmış bir parçacığa ait hızın 100 iterasyon boyunca aldığı değerlerin oluşturduğu grafik görülmektedir. Bunun yanı sıra V_{max} en fazla alt sınırın iki katı kadar seçilmelidir. V_{max} daha büyük seçildiğinde hız değerlerinin birbirine çok fazla yaklaştığı belirlenmiştir.



Şekil 4.4. Hız değerlerinin sınırlandırılması

Burada yeni hızın bulunmasında kullanılan V_{max}, Clerc ve Kennedy'nin [2002] sunduğu daralma faktörüne benzese de problemin yapısı gereği burada daralma faktörü direk kullanılamaz. Daralma faktörü c_1 ve c_2'ye göre belirlenirken burada V_{max} atama yapılacak istasyon sayısıyla ilgilidir.

Başlangıç çözümünden sonra SUMHDP-1 için PSO işleyişine ait akış şeması Şekil 4.5'te sunulmaktadır.

Şekil 4.5. SUMHDP-1 için PSO akış şeması

4.3. Sayısal Örnek

SUMHDP-1 için PSO algoritmasının uygulamasını anlatmak için Şekil 4.6'daki Bowman problemi ele alınmıştır.

Şekil 4.6. Bowman problemi öncelik diyagramı

Çizelge 4.1'de Bowman problemi için örnek görev süreleri (μ_j, σ_j^2) verilmektedir. Varyanslar, düşük varyans için rasgele belirlenmiştir.

Çizelge 4.1. Bowman problemi için örnek görev süreleri

	Görev (*j*)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Görev Süreleri	Ortalama(μ_j)	11	17	9	5	8	12	10	3
	Varyans(σ_j^2)	3.849	2.342	0.882	1.516	2.410	1.882	1.610	0.447

Bowman problemi için iterasyon sayısı 5, parçacık sayısı 3, çevrim zamanı 20, w=1, c_1=c_2=2, [V_{min}, V_{max}]=[-4, 4], $Z_{0.90}$=1.28 olarak alındığını varsayalım. Çizelge 4.1'deki değerler için Eş. 4.2 kullanılarak alt sınır 4 olarak hesaplanmıştır.

Başlangıç çözümünde (t=0) birinci parçacık için prosedür şöyledir:

- Öncülü ve ardılı olmayan görevler belirlenir: 1, 7, 8.
- Bunlar arasından rasgele bir görev seçilir: mesela 7 olsun.
- Seçilen görev birinci istasyona atanır.
- İstasyon toplam zamanı bulunur.
- Çevrim zamanını aşıp aşmadığı belirlenir.

- Yeni öncülü ve ardılı olmayan görevler belirlenir: 1, 5, 8.
- Bu şekilde tüm görevler bir istasyona atanıncaya kadar devam edilir.

Çizelge 4.2'de, Bowman problemi için Çizelge 4.1'de verilen görev zamanları dikkate alınarak başlangıç çözümü için örnek bir rasgele atama verilmektedir.

Çizelge 4.2. Bowman problemi için örnek bir atama

Öncülü ve Ardılı Olmayan Görevler	Rasgele Birini Seç	Atanan İstasyon	İstasyon Toplam Zamanı	Çevrim Zamanını Aşıyor mu?	Son Atanan İstasyon	Son İstasyon Toplam Zamanı
1,7,8	7	1	11.624	Hayır	1	11.624
1,5,8	1	1	23.991	Evet	2	13.511
2,5,8	2	2	31.185	Evet	3	13.511
3,4,5,8	8	3	22.138	Evet	4	
3,4,5,6	6	4	16.953	Hayır	4	17.335
3,4,5	5	4	22.652	Evet	5	
3,4	3	5	19.322	Hayır	5	19.652
4	4	5	24.807	Evet	6	6.576

Başlangıç çözümünde birinci parçacığın pozisyon vektörü aynı zamanda birinci parçacığın yerel en iyisidir. Çizelge 4.3'te, 3 parçacık için başlangıç çözümü ve Eş.4.3'e göre hesaplanan uygunluk fonksiyonu değerleri verilmektedir.

Çizelge 4.3. Başlangıç çözümü pozisyon vektörü ve yerel en iyi

Parçacık(i) / Görev(j)	1	2	3	4	5	6	7	8	$f(i)$
X_{1j} = Pbest(1)	2	3	5	6	5	4	1	4	2.377
X_{2j} =Pbest(2)	1	4	5	5	3	6	2	5	2.324
X_{3j} =Pbest(3)	1	4	5	6	5	3	2	1	2.354

Çizelge 4.3'teki sonuçlara göre göre sürü içerisinde ikinci parçacığın uygunluk fonksiyonu değeri (f (i)= 2.324) en küçük değere sahip olduğundan global en iyi ikinci parçacığın çözümü Çizelge 4.4'teki gibi olur.

Çizelge 4.4. Başlangıç çözümü global en iyi

Görev	1	2	3	4	5	6	7	8	$f(2)$
Gbest	1	4	5	5	3	6	2	5	2.324

Başlangıç hızları Eş.4.4'e göre rasgele hesaplanır. Buna göre birinci parçacığın birinci görev için başlangıç hız değeri;

$$V_{11}^{0} = -4 + 0{,}75 * (4 - (-4)) = 2$$

Eş.4.4'e göre hesaplanan diğer tüm başlangıç hız değerleri Çizelge 4.5'te verilmektedir.

Çizelge 4.5. Başlangıç hız değerleri

V_{ij}		**Görev (*j*)**							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Parçacık (*i*)	**1**	2.000	3.350	-1.120	1.870	0.170	3.090	3.210	-0.820
	2	0.650	0.447	-0.486	-1.298	0.571	-1.303	0.262	0.255
	3	-3.647	2.149	-1.998	1.055	-0.308	-3.691	2.015	-3.112

Başlangıç çözümünden sonra, birinci iterasyonda başlangıç hız değerleri dikkate alınarak yeni görev-istasyon ataması yapılır. Başlangıç iterasyonundan farklı olarak, istasyona atanacak görev belirlenirken öncülü ve ardılı olmayan görevler arasından mutlak değerce en büyük olan seçilir. Buna göre, birinci iterasyon sonunda elde edilen pozisyon vektörü (Çizelge 4.6), yerel en iyi (Çizelge 4.7) ve global en iyi vektörleri (Çizelge 4.8) aşağıda sunulmuştur.

Çizelge 4.6. Birinci iterasyon pozisyon vektörü

Parçacık(i) / Görev(j)	1	2	3	4	5	6	7	8	$f(i)$
X_{1j}	2	3	4	4	6	5	1	5	2.315
X_{2j}	1	2	3	3	5	4	6	6	2.306
X_{3j}	1	3	5	5	6	2	4	1	2.373

Çizelge 4.7. Birinci iterasyon yerel en iyi

Parçacık(i) / Görev(j)	1	2	3	4	5	6	7	8	$f(i)$
Pbest(1)	2	3	4	4	6	5	1	5	2.315
Pbest(2)	1	2	3	3	5	4	6	6	2.306
Pbest(3)	1	4	5	6	5	3	2	1	2.354

Çizelge 4.8. Birinci iterasyon global en iyi

Görev	1	2	3	4	5	6	7	8	$f(2)$
Gbest	1	2	3	3	5	4	6	6	2.306

Hızların güncellenmesi Eş. 4.5'e göre yapılır. Çizelge 4.9'da birinci iterasyon sonundaki yeni hızlar verilmektedir. Eğer güncellenen hızlar, daha önceden belirlenen $V_{\min}$ den daha küçük olursa yeni hız $V_{\min}$ olarak, $V_{\max}$ dan daha büyük olursa $V_{\max}$ olarak alınır. Örneğin Çizelge 4.3'te verilen birinci parçacığa ait pozisyon vektörü (aynı zamanda Pbest(1)) dikkate alalım. Buna göre birinci parçacığın birinci görev için başlangıç hızı 2 $(V_{11}^0 = 2)$ birinci iterasyondaki yeni hızı aşağıdaki gibi hesaplanır:

$$V_{11}^1 = (1*2 + 2*0{,}137*(2-2) + 2*0{,}952*(1-2))/4 = 0{,}024$$

Çizelge 4.9. Güncellenmiş hızlar

V_{ij}		**Görev (j)**							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Parçacık (i)	**1**	0.024	2.384	-2.62	1.08	-0.544	2.62	4	-0.728
	2	0.652	0.448	-0.488	-1.3	0.572	-1.304	0.264	0.256
	3	-3.648	1.268	-3.268	-0.132	-1.036	1.688	3.636	-2.428

Birinci iterasyondaki işlemler ikinci iterasyon için yapıldığında pozisyon vektörü (Çizelge 4.10), yerel en iyi (Çizelge 4.11) ve global en iyi vektörleri (Çizelge 4.12) aşağıda gibi bulunur:

Çizelge 4.10. İkinci iterasyon pozisyon vektörü

Parçacık(i) / Görev(j)	1	2	3	4	5	6	7	8	$f(i)$
X_{1j}	5	4	3	2	3	2	1	1	0.188
X_{2j}	1	2	3	3	5	4	6	6	2.306
X_{3j}	1	4	5	6	5	3	2	2	2.347

Çizelge 4.11. İkinci iterasyon yerel en iyi

Parçacık(i) / Görev(j)	1	2	3	4	5	6	7	8	$f(i)$
Pbest(1)	5	4	3	2	3	2	1	1	0.188
Pbest(2)	1	2	3	3	5	4	6	6	2.306
Pbest(3)	1	4	5	6	5	3	2	2	2.347

Çizelge 4.12. İkinci iterasyon global en iyi

Görev	1	2	3	4	5	6	7	8	$f(1)$
Gbest	5	4	3	2	3	2	1	1	0.188

Bu problemde literatürde bilinen en iyi sonuç k=5, ikinci iterasyon sonunda bulunmuştur. Ancak tanımlanan iterasyon sayısı 5 olması nedeniyle algoritma iterasyon sayısınca çalışmaya devam eder.

4.4. Hesaplama Sonuçları

Literatürdeki test problemleri önerilen PSO algoritması ile çalıştırılmış ve sonuçları 45 göreve kadar olan problemler için literatürde bulunan iki çalışma sonuçları ile karşılaştırılmıştır. Ayrıca 45 görevden büyük olanlar dahil 911 test probleminin genel bir değerlendirilmesi yapılmıştır.

4.4.1. Karşılaştırmalı sonuçlar

SUMHDP-1 için önerilen PSO algoritmasını değerlendirebilmek için Urban&Chiang [2006] ve Baykasoğlu&Özbakır [2007] çalışmalarında yer alan test problemleri* kullanılmıştır.

Veri setinden okunan görev zamanları görevlerin ortalaması (μ_j) olarak; görev zamanlarının varyansları, $[0,(\mu_j/4)^2]$ arasında olduğunda düşük varyans ve $[0,(\mu_j/2)^2]$ arasında olduğunda yüksek varyans olarak kabul edilmiştir. Görevlerin tamamlanma olasılığı 0.90, 0.95 ve 0.975 ($Z_{0.90}$=1.28, $Z_{0.95}$=1.645 ve $Z_{0.975}$=1.96) olarak alınmıştır**.

SUMHDP-1 için önerilen PSO algoritması Delphi 7.0 ile kodlanmıştır. Şekil 4.7'de PSO algoritmasının çalıştırılması için gerekli arayüz verilmektedir. Bu arayüzde önce "*data set oku"* dan problem seçilir. Dosya bilgisi bölümünde problem ve hemen altında görev sayısı yer alır. Daha sonra tanımlar bölümünde iterasyon sayısı, parçacık sayısı, çevrim süresi, atalet ağırlığı, parçacığın kendi tecrübesi, sürünün tecrübesi, Z değeri, varyans tipi ve son olarak $V_{\min}$, $V_{\max}$ girilir. Aslında $V_{\min}$, -$V_{\max}$ olarak alınır.

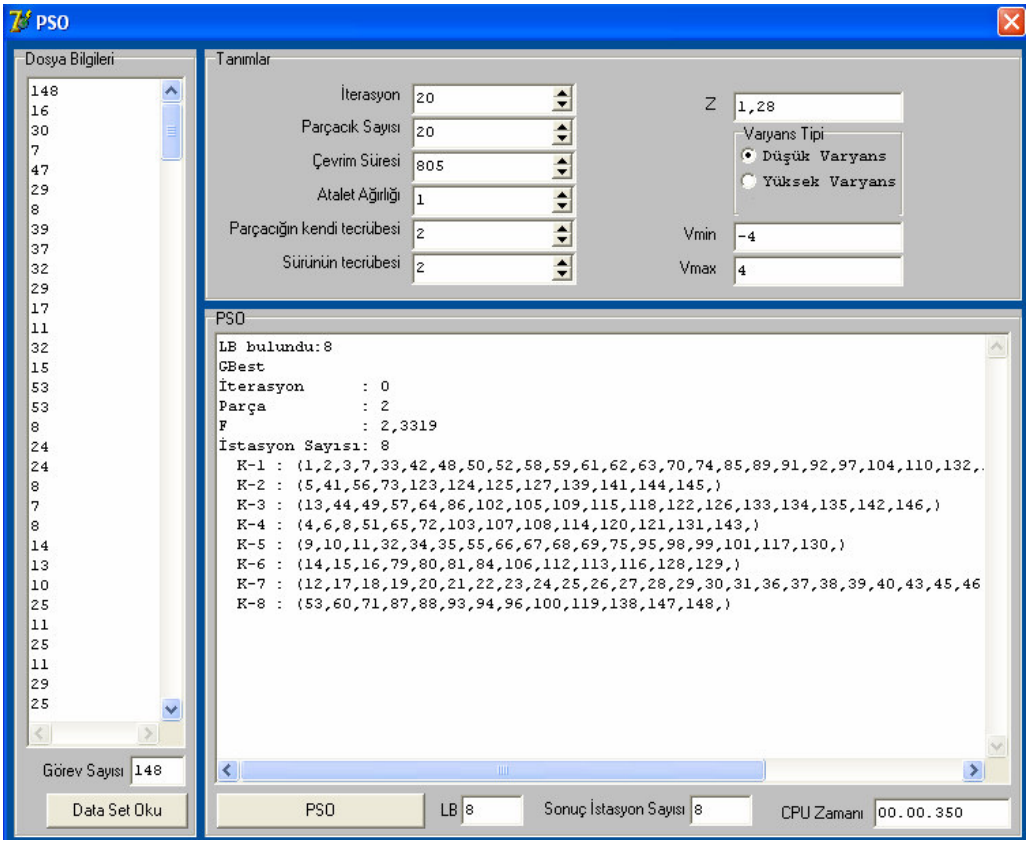
Tüm bu tanımlamalardan sonra *"PSO"* butonuna basılarak kod çalıştırılır. Arayüzün orta bölümünde bulunan PSO bölümünde alt sınır bulundu ise LB'yi veren, bulunamadı ise Gbest'i veren iterasyon, parçacık, uygunluk fonksiyonu değeri ve istasyon-görev atama vektörleri yer alır. Arayüzün alt kısmında hesaplanan LB, bulunan minimum istasyon sayısı ve bu sonuçlara ne kadar CPU zamanında varıldığı yer alır.

Yazılan Delphi kodu EK-1'de sunulmaktadır. Test problemleri hızı 1.80 GHz olan Penthium 4 bir kişisel bilgisayarda çalıştırılmıştır.

---

* Bu test problemlerinin veri setlerine Scholl'un internet sitesinden ulaşılabilir. www.assembly-line-balancing.de

** Test problemlerinin çözümünde dikkate alınan PSO parametreleri yazardan öğrenilebilir.



Şekil 4.7. PSO veri girişi arayüzü

Çizelge 4.13'te düşük varyans, Çizelge 4.14'te ise yüksek varyans için hesaplama sonuçları verilmektedir. Her iki varyans tipi için Z değerleri 0.90, 0.95 ve 0.975 olasılıklarına göre sırasıyla 1.28, 1.645 ve 1.96 alınmıştır.

Şekil 4.7'de 148 görevli Barthold problemi çevrim süresi 805, varyans tipi düşük, $Z_{090} = 1.28$, $[V_{\min}, V_{\max}]=[-4, 4]$, $w=1$, $c_1=c_2=2$, parçacık sayısı ve tekrar sayısı 20 alınarak alt sınır olan 8 istasyon bulunmuştur. CPU zamanı 0.350 saniyede istasyonlara yapılan atamalar orta bölümde verilmektedir. Örneğin K-1, birinci istasyona atanan görevleri gösterir. Alt sınır, 0'ıncı iterasyonda ikinci parçacık tarafından 2,3319 uygunluk fonksiyonu değeriyle bulunmuştur.

Çizelge 4.13. Test problemleri çözümleri: düşük varyans. Baykasoğlu & Özbakır [2007] sonuçları () parantezinde, Urban & Chiang [2006] sonuçları [] parantez içinde verilmiştir.

Problem	*N*	*C*	$Z_{0.975}$=1.96			$Z_{0.95}$=1.645			$Z_{0.90}$=1.28		
			LB	Çözüm	CPU zamanı (s)	LB	Çözüm	CPU zamanı (s)	LB	Çözüm	CPU zamanı (s)
Merten	7	8	5	5(5)[5]	0.02(0.203)[0.22]	4	5(5)[5]	0.06(0.281)[0.06]	4	5(5)[5]	0.063(0.078)[0.20]
		10	4	4(4)[4]	0.04(0.17)[0.06]	4	4(4)[4]	0.06(0.20)[0.07]	4	4(4)[4]	0.016(0.18)[0.08]
		15	3	3(3)[3]	0.001(0.079)[0.06]	3	3(3)[3]	0.01(0.203)[0.05]	3	3(3)[3]	0.01(0.22)[0.05]
		18	2	2(2)[2]	0.001(0.281)[0.04]	2	2(2)[2]	0.01(0.155)[0.04]	2	2(2)[2]	0.01(0.187)[0.05]
Bowman	8	20	5	6(6)[6]	0.016(0.172)[12.64]	5	6(6)[6]	0.016(0.203)[5.71]	4	5(5)[5]	0.047(0.094)[2.32]
Jaeschke	9	6	7	8(8)[8]	0.11(0.172)[1.06]	7	8(8)[8]	0.187(0.203)[0.29]	7	8(8)[8]	0.062(0.156)[0.67]
		7	6	7(7)[7]	0.265(0.157)[0.29]	6	7(7)[7]	0.187(0.172)[0.63]	6	7(7)[7]	0.062(0.23)[0.47]
		8	5	7(7)[7]	0.25(0.172)[6.71]	5	7(7)[7]	0.12(0.09)[6.66]	5	7(7)[7]	0.063(0.171)[5.64]
		10	4	5(5)[5]	0.11(0.141)[0.18]	4	5(5)[5]	0.203(0.141)[0.4]	4	**5(5)[4]**	0.063(0.13)[0.14]
		18	3	3(3)[3]	0.001(0.14)[0.09]	3	3(3)[3]	0.01(0.11)[0.07]	3	3(3)[3]	0.001(0.203)[0.06]
Jackson	11	9	6	7(7)[7]	0.125(0.17)[7.59]	6	7(7)[7]	0.17(0.204)[20.68]	6	7(7)[7]	0.094(0.2)[120.46]
		10	6	7(7)[7]	0.030(0.063)[1,010.97]	6	7(7)[7]	0.17(0.183)[6,299.81]	5	**6(7)[7]**	0.170(0.172)[1,818.93]
		13	4	5(5)[5]	0.029(0.14)[3.29]	4	5(5)[5]	0.16(0.204)[1.28]	4	5(5)[5]	0.093(0.13)[27.26]
		14	4	4(4)[4]	0.031(0.188)[0.92]	4	4(4)[4]	0.371(1.1)[0.34]	4	4(4)[4]	0.01(0.24)[0.25]
		21	3	3(3)[3]	0.016(0.187)[0.11]	3	3(3)[3]	0.01(0.14)[0.09]	3	3(3)[3]	0.01(0.157)[0.50]
Mitchell	21	15	Mümkün çözüm yok			Mümkün çözüm yok			8	9(9)[9]	0.188(0.297)[*]
		21	6	6(6)[6]	0.562(0.5)[245.57]	6	6(6)[6]	0.34(0.843)[0.88]	6	6(6)[6]	0.14(0.26)[4.85]
		26	5	5(5)[5]	0.148(0.34)[1.18]	5	5(5)[5]	0.016(0.344)[8.01]	5	5(5)[5]	0.01(0.234)[2.91]
		35	4	4(4)[4]	0.007(0.28)[0.2]	4	4(4)[4]	0.01(0.21)[0.35]	4	4(4)[4]	0.01(0.281)[0.25]
		39	3	4(4)[4]	0.105(0.21)[0.99]	3	4(4)[4]	0.188(0.235)[203.21]	3	3(3)[3]	0.047(0.156)[1.01]

Çizelge 4.13. (Devam) Test problemleri çözümleri: düşük varyans. Baykasoğlu & Özbakır [2007] sonuçları () parantezinde, Urban & Chiang [2006] sonuçları [] parantez içinde verilmiştir.

Problem	*N*	*C*	$Z_{0.975}$=1.96			$Z_{0.95}$=1.645			$Z_{0.90}$=1.28		
			LB	Çözüm	CPU zamanı (s)	LB	Çözüm	CPU zamanı (s)	LB	Çözüm	CPU zamanı (s)
Heskiaoff	28	205	6	6(6)[6]	2.491(11.031)[164.71]	6	6(6)[6]	0.595(3.297)[85.32]	6	6(6)[6]	0.25(0.343)[86.91]
		216	6	6(6)[6]	0.610(1.97)[524.95]	6	6(6)[6]	0.016(0.1)[1.44]	5	6(6)[6]	0.203(0.25)[61.79]
		256	5	5(5)[5]	0.111(0.405)[1.69]	5	5(5)[5]	0.016(0.36)[4.52]	5	5(5)[5]	0.015(0.328)[0.52]
		324	4	4(4)[4]	0.016(0.453)[87.52]	4	4(4)[4]	0.015(0.25)[0.24]	4	4(4)[4]	0.01(0.454)[0.23]
		342	4	4(4)[4]	0.016(0.328)[0.67]	4	4(4)[4]	0.015(0.328)[0.44]	4	4(4)[4]	0.01(0.406)[0.37]
Killbridge	45	79	8	9(9)	0.288(0.39)	8	9(9)	0.33(0.39)	8	8(8)	0.578(5.203)
		92	7	8(8)	0.265(0.594)	7	**7(8)**	2.032(0.391)	7	7(7)	0.313(1.37)
		110	6	6(6)	0.211(0.4)	6	6(6)	0.14(0.4)	6	6(6)	0.016(0.594)
		138	5	5(5)	0.023(0.578)	5	5(5)	0.016(0.2)	5	5(5)	0.016(0.39)
		184	4	4(4)	0.016(0.391)	4	4(4)	0.016(0.112)	4	4(4)	0.015(0.45)

* 10 CPU saati içinde çözüm bulunamadı.

Çizelge 4.14. Test problemleri çözümleri: yüksek varyans. Baykasoğlu & Özbakır [2007] sonuçları () parantezinde, Urban & Chiang [2006] sonuçları [] parantez içinde verilmiştir.

Problem	*N*	*C*	$Z_{0.975}$=1.96			$Z_{0.95}$=1.645			$Z_{0.90}$=1.28		
			LB	Çözüm	CPU zamanı (s)	LB	Çözüm	CPU time (s)	LB	Çözüm	CPU zamanı (s)
Merten	7	8	Mümkün çözüm yok			Mümkün çözüm yok			5	5(5)[5]	0.313(6.172)[0.07]
		10	4	5(5)[5]	0.05(0.1)[1.48]	4	5(5)[5]	0.032(0.18)[3.90]	4	5(5)[5]	0.047(0.14)[3.83]
		15	3	3(3)[3]	0.01(0.13)[0.06]	3	3(3)[3]	0.01(0.125)[0.13]	3	3(3)[3]	0.01(0.11)[0.04]
		18	2	3(3)[3]	0.031(0.09)[0.10]	2	3(3)[3]	0.031(0.078)[0.18]	2	2(2)[2]	0.016(0.075)[0.05]
Bowman	8	20	5	6(6)[6]	0.046(7.12)[1.89]	5	6(6)[6]	0.062(0.171)[7.12]	5	6(6)[6]	0.063(0.2)[8.43]
Jaeschke	9	8	6	7(7)[7]	0.75(0.922)[3.05]	6	7(7)[7]	0.344(0.531)[4.61]	6	7(7)[7]	0.078(1.47)[12.91]
		10	5	7(7)[7]	0.078(0.125)[101.74]	5	7(7)[7]	0.125(0.219)[55.14]	5	**6(7)[7]**	0.906(0.187)[50.22]
		18	3	3(3)[3]	0.048(0.234)[0.11]	3	3(3)[3]	0.01(0.175)[0.13]	3	3(3)[3]	0.01(0.3)[0.12]
Jackson	11	10	Mümkün çözüm yok			Mümkün çözüm yok			6	7(7)[7]	0.172(0.203)[522.34]
		13	5	5(5)[5]	2.141(2.04)[1.26]	5	5(5)[5]	0.969(0.985)[1.67]	4	5(5)[5]	0.14(1.402)[2.43]
		14	4	5(5)[5]	0.5(0.891)[10.62]	4	5(5)[5]	0.109(0.25)[1.77]	4	5(5)[5]	0.094(0.772)[17.77]
		21	3	3(3)[3]	0.031(0.766)[0.06]	3	3(3)[3]	0.047(0.187)[0.11]	3	3(3)[3]	0.01(0.31)[0.09]
Mitchell	21	21	6	8(8)[8]	0.329(0.344)[32,791.40]	6	7(7)[7]	0.375(0.231)[822.35]	6	7(7)[7]	0.313(0.516)[738.35]
		26	5	6(6)[6]	0.359(0.782)[204.89]	5	6(6)[6]	0.125(0.344)[62.64]	5	5(5)[5]	0.671(1.89)[116.32]
		35	4	4(4)[4]	0.062(5.468)[45.08]	4	4(4)[4]	0.157(0.562)[2.40]	4	4(4)[4]	0.015(0.281)[1.85]
		39	4	4(4)[4]	0.01(0.174)[0.20]	4	4(4)[4]	0.015(0.235)[2.00]	3	4(4)[4]	0.141(0.344)[0.20]

Çizelge 4.14. (Devam) Test problemleri çözümleri: yüksek varyans. Baykasoğlu & Özbakır [2007] sonuçları () parantezinde, Urban & Chiang [2006] sonuçları [] parantez içinde verilmiştir.

Problem	*N*	*C*	$Z_{0.975}$=1.96			$Z_{0.95}$=1.645			$Z_{0.90}$=1.28		
			LB	Çözüm	CPU zamanı (s)	LB	Çözüm	CPU time (s)	LB	Çözüm	CPU zamanı (s)
Heskiaoff	28	205	6	8(8)[8]	0.704(0.547)[*]	6	7(7)[7]	0.922(1.641)[22,587.20]	6	7(7)[7]	0.375(0.437)[290.00]
		216	6	7(7)[7]	0.906(1.976)[536.21]	6	7(7)[7]	0.547(0.563)[494.62]	6	6(6)[6]	1.362(5.593)[434.14]
		256	5	6(6)[6]	0.31(0.48)[36.54]	5	6(6)[6]	0.39(0.53)[44.23]	5	5(5)[5]	0.631(1.453)[29.22]
		324	4	5(5)[5]	0.235(0.691)[33.61]	4	4(4)[4]	0.21(0.328)[5.55]	4	4(4)[4]	0.391(0.531)[2.46]
		342	4	4(4)[4]	0.125(0.531)[1.57]	4	4(4)[4]	0.016(0.657)[1.01]	4	4(4)[4]	0.031(0.18)[1.29]
Killbridge	45	92	7	8(8)	0.922(0.61)	7	8(8)	2.938(5.547)	7	8(8)	0.469(0.594)
		110	6	7(7)	0.438(0.609)	6	7(7)	0.61(0.984)	6	6(6)	3.585(4.14)
		138	5	6(6)	0.437(0.782)	5	**5(6)**	0.734(0.39)	5	5(5)	0.14(0.797)
		184	4	4(4)	0.063(0.593)	4	4(4)	0.016(0.781)	4	4(4)	0.03(0.593)

[*] 10 CPU saati içinde çözüm bulunamadı.

Düşük varyans için Çizelge 4.13'te verilen sonuçlar incelendiğinde önerilen PSO algoritmasının, Urban&Chiang [2006] ve Baykasoğlu&Özbakır'ın [2007] algoritmaları ile aynı sonuçları verdiği görülmektedir. Ayrıca PSO algoritması, Jackson probleminde ($Z_{0.90}$=1.28, C=10) Urban&Chiang [2006] ve Baykasoğlu&Özbakır'ın [2007] sunduğu sonuçlardan daha iyi; Killbridge probleminde ($Z_{0.95}$=1.645, C=92) Baykasoğlu&Özbakır'ın [2007] sunduğu sonuçtan daha iyi (optimal) sonuç vermiştir. Çizelge CPU süresi açısından değerlendirildiğinde, Mitchell problemine Urban&Chiang [2006] ($Z_{0.90}$=1.28, C=15) 10 CPU saatini aşan bir sürede sonuç bulabilmiş iken, önerilen PSO algoritması 0.188 saniyede sonuç vermiştir. Jaeschke probleminde ($Z_{0.90}$=1.28, C=10) Urban ve Chiang [2006] optimal sonucu bulurken, önerilen PSO algoritması optimal sonucu elde edememiştir.

Yüksek varyans için Çizelge 4.14'te verilen sonuçlar incelendiğinde, önerilen PSO algoritmasının Urban&Chiang [2006] ve Baykasoğlu&Özbakır'ın [2007] algoritmaları ile aynı sonuçları verdiği görülmektedir. Ayrıca PSO algoritması, Jaeschke probleminde ($Z_{0.90}$=1.28, C=10) Urban&Chiang [2006] ve Baykasoğlu&Özbakır'ın [2006] bulduğu sonuçtan daha iyi; Killbridge probleminde ($Z_{0.95}$=1.645, C=138) Baykasoğlu&Özbakır'ın [2007] bulduğu sonuçtan daha iyi (optimal) sonuç vermiştir. Çizelge CPU süresi açısından değerlendirildiğinde, Heskiaoff probleminde ($Z_{0.975}$=1.96, C=205) Urban&Chiang [2006] 10 CPU saatini aşan bir sürede ve aynı problemde $Z_{0.95}$=1.645 için Urban&Chiang [2006] 22587.20 saniyede sonuç bulabilmiş iken, önerilen PSO algoritması 1 saniyenin altında aynı sonuçları bulmuştur.

Çizelge 4.13 ve Çizelge 4.14'de problemler görev sayısı açısından değerlendirildiğinde, Urban&Chiang [2006] 28 görevli problemlere kadar çözüm bulabilirlerken; Baykasoğlu&Özbakır'ın [2007] 45 görevli problemlere kadar çözüm bulmuşlardır. Önerilen PSO algoritması 297 görevli problemlere kadar çözüm bulabilmektedir.

Çizelge 4.13 ve Çizelge 4.14'te yer alan problemler için ortalama CPU zamanı, önerilen PSO algoritmasında 0.198, Baykasoğlu&Özbakır'ın [2007] 0.682 ve Urban&Chiang'ın [2006] 414.383'dür.

Önerilen PSO algoritması, 45'ten daha fazla görevden oluşan problemlere de uygulanmıştır. Beklendiği şekilde problemin boyutu büyüdükçe daha fazla parçacık ve iterasyon gerekmektedir. Dolayısıyla çözümün bulunması için de CPU zamanları artmaktadır. Eğer alt sınır yerine herhangi bir çözüm tercih ediliyorsa, daha az parçacık ve iterasyon kullanılarak daha kısa CPU zamanlarında çözüm elde edilebilmektedir.

4.4.2. Genel hesaplama sonuçları

4.4.1.'de, PSO algoritmasının karşılaştırılması Urban&Chiang'de [2006] 28 göreve kadar, Baykasoğlu&Özbakır'da [2007] 45 göreve kadar olan problemlerde yapılmıştır. Oysa önerilen PSO algoritması büyük boytulu problemlerde de rahatlıkla çalışabilmektedir. Bunu göstermek için Scholl'un internet sitesindeki test problemleri kullanılmıştır.

SUMHDP-1 için önerilen PSO algoritması toplam, farklı isimdeki 25 test problemi için çalıştırılmıştır. Bu problemler düşük ve yüksek varyansa göre, olasılıklar ise 0.90, 0.95 ve 0.975 alınarak denenmiştir. Başka bir deyişle, her problem bir çevrim zamanı için eğer mümkün çözümü varsa yani en yüksek görev zamanı çevrim zamanını aşmıyorsa 6 kez çalıştırılmıştır. Sonuçlar varyans tipine göre; EK-2'de düşük varyans için elde edilen sonuçlar, EK-3'te ise yüksek varyans için elde edilen sonuçlar olmak üzere sunulmuştur.

EK-2 ve EK-3'te varyans tipi ve olasılıklara göre toplam 911 problem çalıştırılmış ve sonuçları alınmıştır. Problemler çalıştırılırken 500 saniye (CPU time) kısıtı konulmuştur. Bu süre içerisinde algoritma minimum istasyon sayısını alt sınıra eşitleyebilirse durmaktadır. Eğer bu süre içerisinde algoritma alt sınırı yakalayamazsa bulduğu en iyi sonuç alınmıştır.

Elde edilen sonuçlar literatürde bu büyüklükteki problemlerin sonuçlarının bulunmaması nedeniyle karşılaştırılamamıştır.

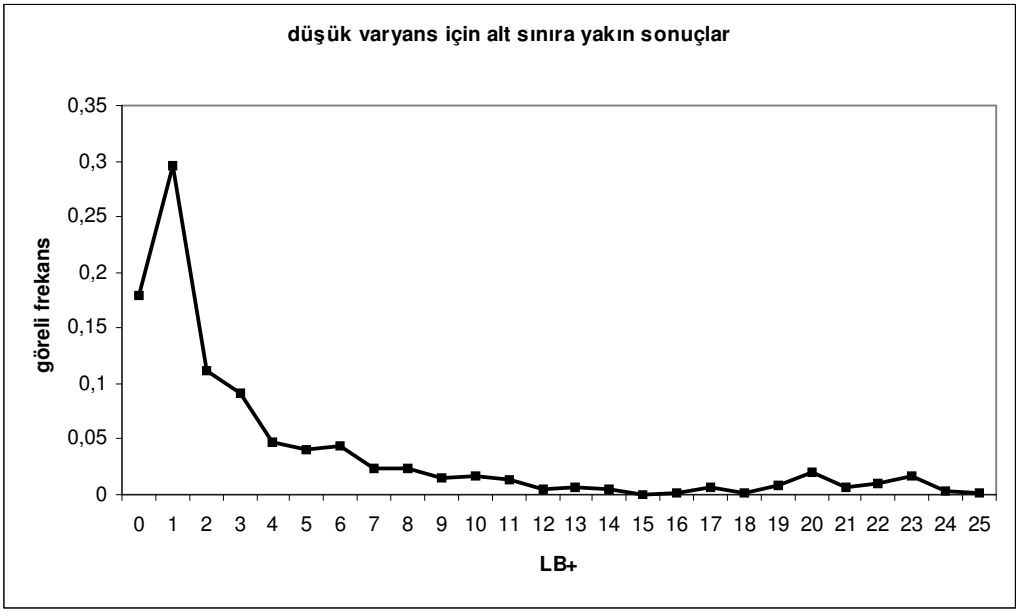
Düşük varyans için değerlendirme

EK-2'de düşük varyans için 547 problemin sonucu yer almaktadır. Düşük varyans için tüm sonuçlara ait özet, Çizelge 4.15'te verilmektedir.

Çizelge 4.15. Düşük varyans için alt sınıra yakın sonuçlar

Alt sınıra yaklaşık sonuç	Frekans	***Alt sınıra yaklaşık sonuç***	Frekans	***Alt sınıra yaklaşık sonuç***	Frekans	***Alt sınıra yaklaşık sonuç***	Frekans
0	98	***7***	13	***14***	3	***21***	4
1	162	***8***	13	***15***	0	***22***	6
2	61	***9***	8	***16***	1	***23***	9
3	50	***10***	9	***17***	4	***24***	2
4	26	***11***	7	***18***	1	***25***	1
5	22	***12***	3	***19***	5		
6	24	***13***	4	***20***	11		

Çizelge 4.15'te (alt sınıra yaklaşık sonuç=0) ifadesi, alt sınırın bulunduğunu; (alt sınıra yaklaşık sonuç=1) ifadesi, alt sınırın 1 fazlasının bulunduğunu; (alt sınıra yaklaşık sonuç=2) ifadesi, alt sınırın 2 fazlasının bulunduğunu tanımlamaktadır. Çözülen 547 problemde elde edilen sonuçlar alt sınıra 25 yaklaşık olana kadar gitmektedir. Bu sonuçlar incelendiğinde 98 problemde alt sınırın bulunduğu, 162 problemde alt sınıra bir yaklaşık, 61 problemde alt sınıra iki yaklaşık ve 50 problemde alt sınıra üç yaklaşık sonucun elde edildiği görülmektedir. Bu sonuçlar tüm sonuçların %67.82'sini oluşturmaktadır. Bulunan sonuçların tüm sonuçlar içindeki göreli frekansı ve alt sınıra yaklaşık sonuç ilişkisini Şekil 4.8'de daha iyi görmek mümkündür.



Şekil 4.8. Düşük varyans için alt sınıra yakın sonuçlar

Yüksek varyans için değerlendirme

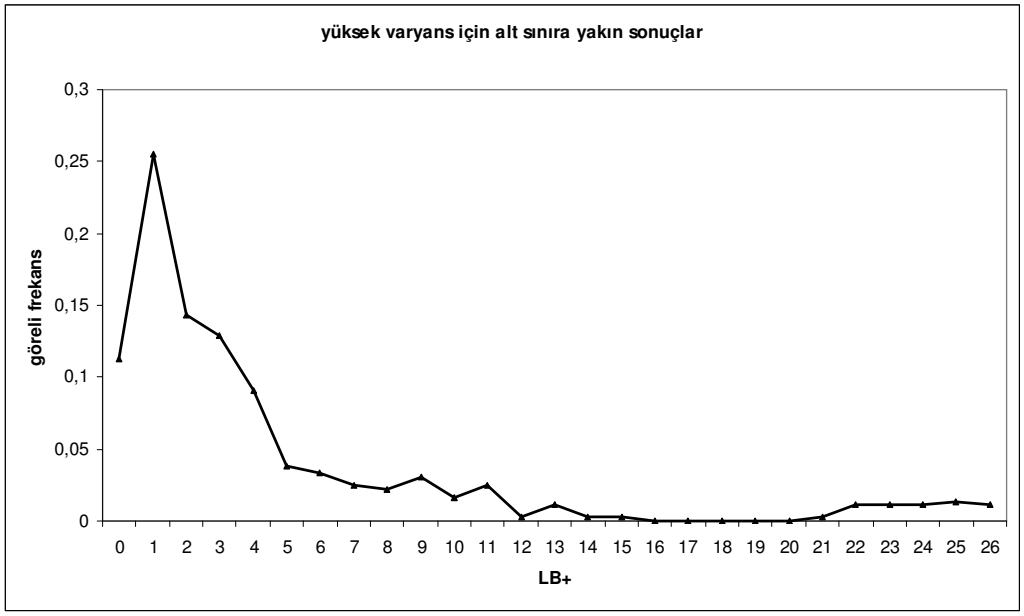
EK-3'te yüksek varyans için 364 problemin sonucu yer almaktadır. Yüksek varyans için tüm sonuçlara ait özet, Çizelge 4.16'da verilmektedir.

Çizelge 4.16. Yüksek varyans için sonuçlar

Alt sınıra yaklaşık sonuç	Frekans	***Alt sınıra yaklaşık sonuç***	Frekans	***Alt sınıra yaklaşık sonuç***	Frekans	***Alt sınıra yaklaşık sonuç***	Frekans
0	41	***7***	9	***14***	1	***21***	1
1	93	***8***	8	***15***	1	***22***	4
2	52	***9***	11	***16***	0	***23***	4
3	47	***10***	6	***17***	0	***24***	4
4	33	***11***	9	***18***	0	***25***	5
5	14	***12***	1	***19***	0	***26***	4
6	12	***13***	4	***20***	0		

Bu sonuçlar incelendiğinde 41 problemde alt sınırın bulunduğu, 93 problemde alt sınıra bir yaklaşık, 52 problemde alt sınıra iki yaklaşık, 47 problemde alt sınıra üç yaklaşık ve 33 problemde alt sınıra dört yaklaşık sonucun elde edildiği görülmektedir. Bu sonuçlar tüm sonuçların %73.07'sini oluşturmaktadır.

Elde edilen sonuçlar en fazla alt sınıra 26 yaklaşık olarak gerçekleşmiştir. Bulunan sonuçların tüm sonuçlar içindeki göreli frekansı ve alt sınıra yaklaşık sonuç ilişkisini Şekil 4.9'da daha iyi görmek mümkündür.



Şekil 4.9. Yüksek varyans için alt sınıra yakın sonuçlar

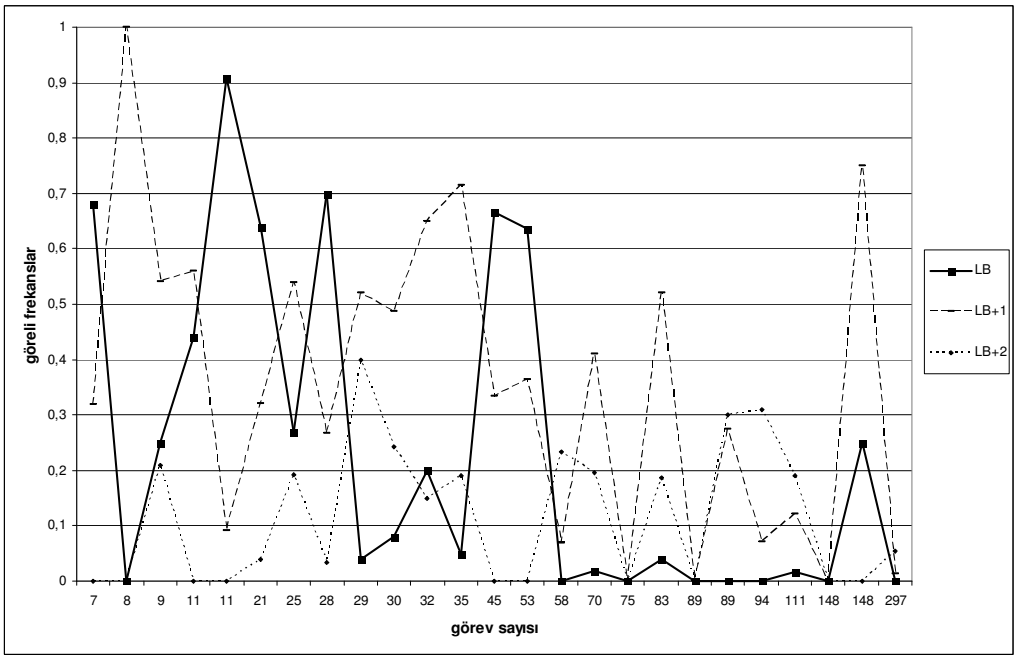
Sonuçların genel değerlendirmesi

Bu çalışmada toplam (düşük ve yüksek varyans) test edilen 911 problemden 139'unda alt sınır yakalanmıştır. Bunun yanı sıra 255 problemde alt sınıra 1 yaklaşık, 113 problemde 2 yaklaşık, 97 problemde 3 yaklaşık, 59 problemde 4 yaklaşık sonuç elde edilmiştir. Bu sonuçların toplam içindeki oranı % 72.78'dir. Tüm sonuçlar aşağıdaki Çizelge 4.17'de özetle sunulmaktadır. Bu sonuçlara göre tüm sonuçlar, düşük ve yüksek varyans için bulunan sonuçlara benzerlik göstermektedir.

Çizelge 4.17. Genel sonuçlar ve frekansları

Alt sınıra yaklaşık sonuç	Frekans	***Alt sınıra yaklaşık sonuç***	Frekans	***Alt sınıra yaklaşık sonuç***	Frekans	***Alt sınıra yaklaşık sonuç***	Frekans
0	139	***7***	22	***14***	4	***21***	5
1	255	***8***	21	***15***	1	***22***	10
2	113	***9***	19	***16***	1	***23***	13
3	97	***10***	15	***17***	4	***24***	6
4	59	***11***	16	***18***	1	***25***	6
5	36	***12***	4	***19***	5	***26***	4
6	36	***13***	8	***20***	11		

Bunun yanında problemdeki görev sayısı arttıkça problem zorlaştığından alt sınıra ulaşmak da zorlaşmaktadır. Bu durumu Şekil 4.10'da incelemek mümkündür. Şekil 4.10'da görev sayısı ile alt sınır, alt sınıra bir yaklaşık ve iki yaklaşık sonuç arasındaki ilişki görülmektedir. Şekil 4.10'a göre görev sayısı arttıkça alt sınırı yakalamak düzenli bir azalma göstermese de nispeten azalmaktadır. Bunun yanında alt sınıra bir ve iki yaklaşık sonuç bulunması ihtimali artmaktadır.



Şekil 4.10. Görev sayısı ve alt sınırın göreli frekans ilişkisi

Alt sınır ve elde edilen sonuçlar arasındaki farkı etkileyen bir diğer unsur ise çevrim sürelerinin deterministik problemler için belirlenmiş olmasıdır. Aslında bu çevrim sürelerinin stokastik görev zamanları için çok uygun olmadığı değerlendirilmektedir. Ancak önerilen PSO algoritmasının etkinliğini kanıtlayabilmek için bu çevrim süreleri bir gösterge olarak kullanılmıştır.

Deterministik görev zamanlı U-tipi hat dengeleme problemleri için bilinen en iyi sonuçları veren çalışma Scholl ve Klein [1999] tarafından sunulan ULINO'dur. Eğer ULINO'nun deterministik zamanlı U-tipi hat dengeleme problemleri için bulduğu sonuçlar incelenirse Wee-Mag probleminde çevrim süresi 40 için alt sınır 38 istasyon olmasına rağmen ULINO'nun bulduğu optimum sonuç 60'tır. Buna göre fark 22 istasyondur. Wee-Mag probleminde diğer çevrim zamanları için de alt sınır ve optimum sonuç arasında önemli farklar oluşabilmektedir.

Görüldüğü gibi deterministik görev zamanları için türetilen problemlerde, deterministik süreli görevlerden oluşan U-tipi hatlarda önemli farklar olabilirken, görev zamanlarının stokastik olduğu durumda önemli farkların oluşabilmesi mümkündür. Başka bir deyişle, stokastik görev zamanlı test edilen bu problemlerde deterministik görev zamanlı problemlere nazaran, alt sınırın daha az sayıda yakalanması mümkündür.

Problemlerin görev sayısı ile elde edilen alt sınır ve alt sınıra yakın sonuçların incelenebilmesi için özet olarak Çizelge 4.18 sunulmaktadır. Bu çizelgeden anlaşılacağı gibi 500 saniye kısıtı altında problemdeki görev sayısı arttıkça elde edilen sonuçlar alt sınırdan uzaklaşabilmektedir. Sadece 148 görevli olmasına rağmen Barthold probleminde önerilen PSO algoritması 28 problemden 7'sinde alt sınırı rahatlıkla bulmuş ve diğer 21 problemde alt sınıra 1 yaklaşık sonuç elde etmiştir.

Çizelge 4.18. Problemlere göre sonuçların dağılımı

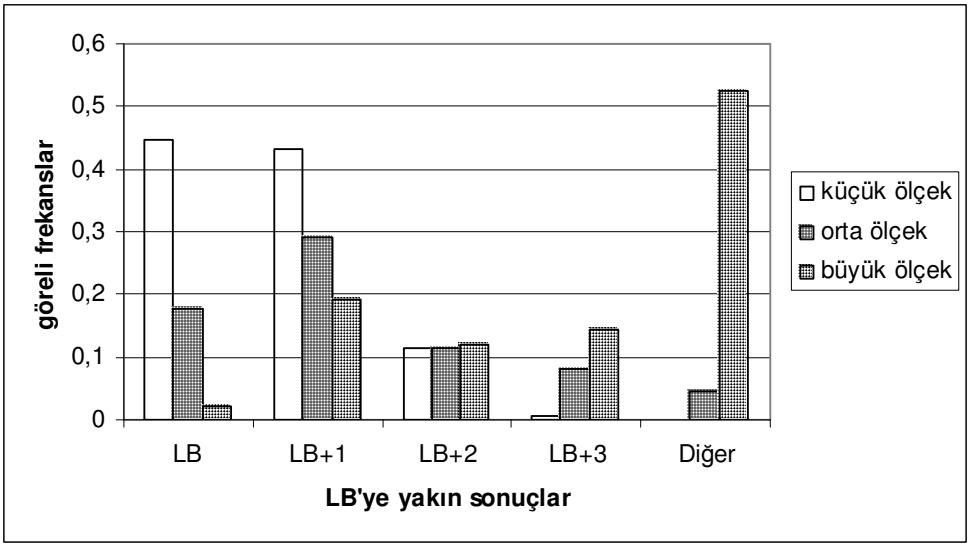
Problemin adı	**görev sayısı**	**Çalıştırılan problem sayısı**	**LB sayısı**	**LB+1 sonuç sayısı**	**LB+2 sonuç sayısı**	**LB+3 sonuç sayısı**	**Diğer sonuç sayısı**
Mertens	7	22	15	7	-	-	-
Bowman	8	6	-	6	-	-	-
Jaeschke	9	24	6	13	5	-	-
Jackson	11	25	11	14	-	-	-
Mansoor	11	11	10	1	-	-	-
Mitchell	21	25	16	8	1	-	-
Roszieg	25	26	7	14	5	-	-
Heskiaoff	28	30	21	8	1	-	-
Buxey	29	25	1	13	10	1	-
Sawyer	30	37	3	18	9	7	-
Lutz1	32	20	4	13	3	-	-
Gunther	35	21	1	15	4	1	-
Kilbridge	45	27	18	9	-	-	-
Hahn	53	22	14	8	-	-	-
Warnecke	58	43	-	3	10	10	10
Tonge	70	56	1	23	11	13	8
Wee-mag	75	71	-	-	-	-	71
Arcus 1	83	75	3	39	14	15	4
Lutz2	89	39	-	-	-	-	39
Lutz3	89	40	-	11	12	10	7
Mukherje	94	42	-	3	13	13	13
Arcus 2	111	58	1	7	11	13	26
Barthold2	148	63	-	-	-	3	60
Barthold	148	28	7	21	-	-	-
Scholl	297	75	-	1	4	11	59

Stokastik problemler varyans değerleri “0” alındığında deterministik problem haline dönüşür. SUMHDP-1 için önerilen PSO algoritması deterministik UMHP-1 için de kullanılabilir.

Bir diğer inceleme, problemlerin görev sayısına göre yapılmıştır. Problemler; küçük, orta ve büyük ölçekte problemler olarak 3 kategoride ele alınmıştır.

- $1 \leq$ Görev sayısı < 30 için küçük ölçekte problemler,
- $30 \leq$ Görev sayısı < 60 için orta ölçekte problemler,
- $60 \leq$ Görev sayısı ≤ 297 için büyük ölçekte problemler.

Problem büyüklüğüne göre küçük, orta ve büyük ölçekli problemlerde alt sınırı bulma ve alt sınıra yakın olan sonuçların incelemesi Şekil 4.11'e göre yapılabilir.



Şekil 4.11. Problemin ölçeğine göre sonuçlar

Şekil 4.11'den açık bir şekilde anlaşılmaktadır ki küçük ölçekli problemlerde alt sınırı yakalama ve alt sınıra bir yaklaşık sonuç oranı orta ve büyük ölçekli olanlara nazaran oldukça fazladır. Alt sınıra iki yaklaşık sonuçlar arasında bir fark görünmezken, alt sınıra üç yaklaşık sonuçtan itibaren orta ve büyük ölçekli problemlerin oranı artmaktadır.

4.5. PSO Parametrelerinin Optimal Çözüm Üzerindeki Etkisinin İncelenmesi

PSO parametrelerinin SUMHDP-1 probleminde optimal sonucun bulunmasındaki etkilerini inceleyebilmek amacıyla Çizelge 4.19'da belirtilen farklı büyüklüklerdeki 5 problem dikkate alınmıştır. Ele alınan problemler ile birlikte görev zamanlarının stokastik olmasından dolayı varyans $[0,(\mu/4)^2]$ aralığında düşük tipte ve $Z_{0.975}$=1.96 alınmıştır. Ayrıca iterasyon sayısı 10 olarak belirlenmiştir.

Çizelge 4.19. Dikkate alınan problemler

Problemin adı	**Görev sayısı**	**Çevrim süresi**	**Alt sınır**
Jackson	11	C=14	LB=4
Lutz1	32	C=2020	LB=8
Mukherje	94	C=263	LB=17
Arcus 2	111	C=11570	LB=14
Barthold2	148	C=170	LB=26

Çizelge 4.20'de ise ele alınan faktörler ve faktör düzeyleri görülmektedir. PSO'da faktör sayısı 5 ve faktör düzeyleri 3 olarak alındığında, $5x3^5$=1215 deneme yapılmıştır.

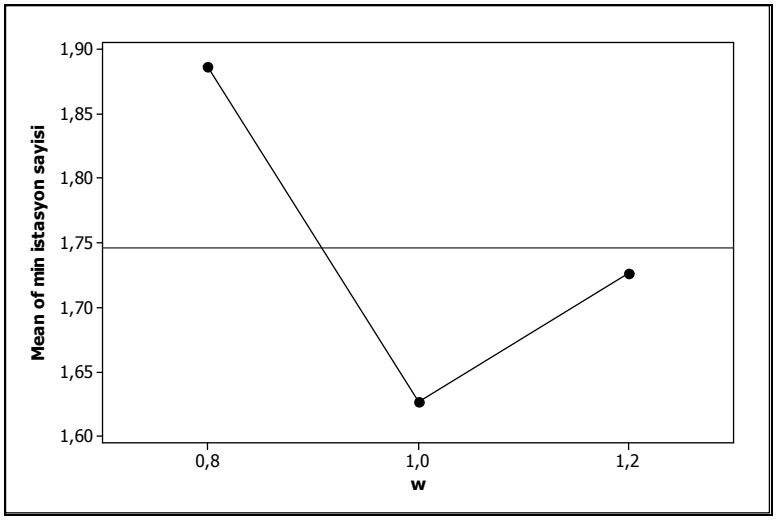
Çizelge 4.20. Faktör düzeyleri

Faktör	**Düzey**		
	1	**2**	**3**
Parçacık Sayısı	Görev Sayısı	Görev Sayısı x 2	Görev Sayısı x 3
Atalet ağırlığı (w)	0.8	1	1.2
c_1	2	4	6
c_2	2	4	6
Maksimum Hız (V_{max})	LB x (1/2)	LB	LB x 2

Yapılan denemelerle, elde edilen sonuçların (minimum istasyon sayıları) normal dağılım ve eşit varyanslar varsayımına uymadığı görülmüştür. Bu sebeple, ANOVA testlerinin, optimal sonuçların bulunmasında PSO parametre etkilerini göstermesi amacıyla kullanılamayacağı değerlendirilmektedir. ANOVA testleri yerine, PSO parametrelerinin optimal çözümün bulunmasındaki ana etkileri ve ikili etkileri şekillerle incelenmiştir. Şekillerin çizimi için Minitab 14 kullanılmıştır. Şekillerle ilgili yorum yapılırken minimum istasyon sayısının en küçük değere yakın olmasına dikkat edilmiştir. Bu kapsamda, minimum sonucu veren PSO parametre değerleri irdelenmiştir.

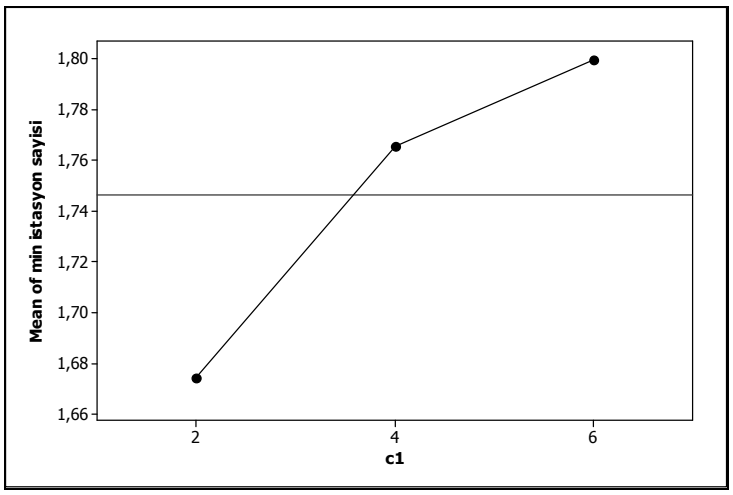
4.5.1. PSO parametrelerinin optimal çözüm üzerindeki ana etkileri

Çizelge 4.20'de belirtilen PSO parametrelerinin, minimum istasyon sayısının bulunmasındaki etkileri 5 ayrı şekille incelenmiştir.



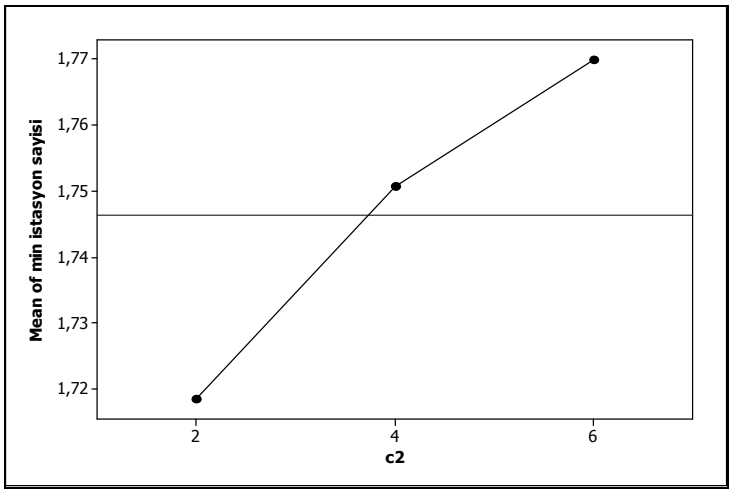
Şekil 4.12. *w*'nun minimum istasyon sayısına etkisi

Şekil 4.12 elde edilen minimum istasyon sayısı ile *w* arasındaki ilişkiyi göstermektedir. Şekil 4.12'ye göre *w* değeri sırasıyla 1 ve 1.2 olduğunda minimum istasyon sayısının elde edilmesi daha mümkündür.



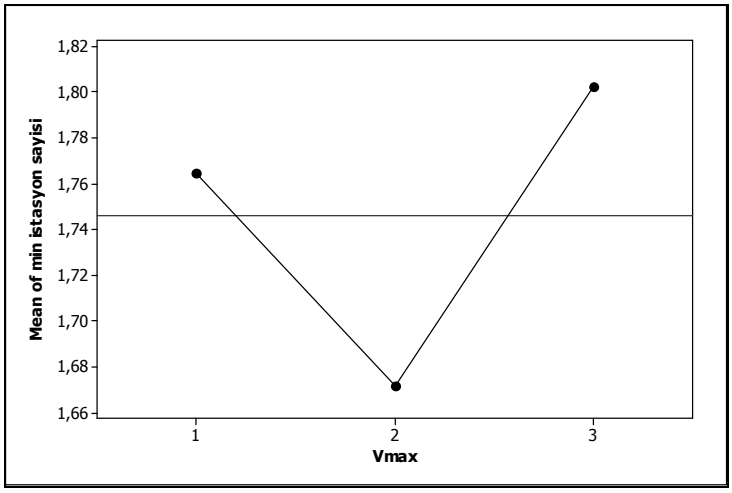
Şekil 4.13. c_1'in minimum istasyon sayısına etkisi

Şekil 4.13 elde edilen minimum istasyon sayısı ile c_1 arasındaki ilişkiyi göstermektedir. Şekil 4.13'e göre, c_1 özellikle 2 değerini aldığında minimum istasyon sayısının elde edilmesi daha mümkündür.



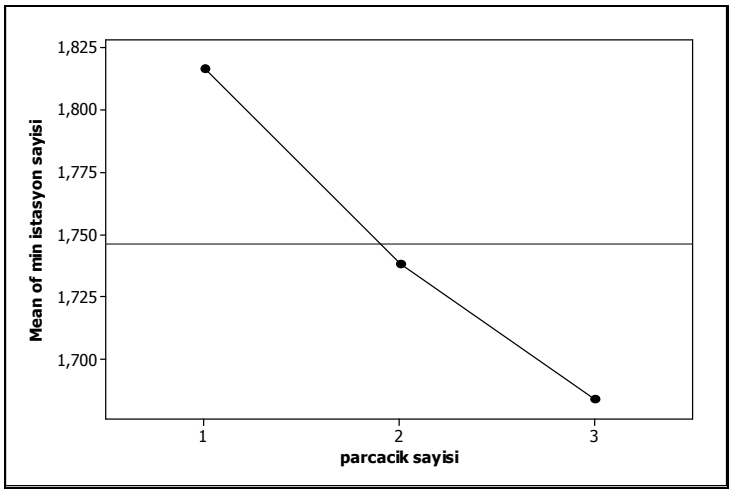
Şekil 4.14. c_2'nin minimum istasyon sayısına etkisi

Şekil 4.14 elde edilen minimum istasyon sayısı ile c_2 arasındaki ilişkiyi göstermektedir. Şekil 4.14'e göre, c_2 özellikle 2 değerini aldığında minimum istasyon sayısının elde edilmesi daha mümkündür.



Şekil 4.15. V_{max}'ın minimum istasyon sayısına etkisi

Şekil 4.15 elde edilen minimum istasyon sayısı ile V_{max} arasındaki ilişkiyi göstermektedir. Şekil 4.15'e göre, V_{max} özellikle alt sınıra eşit değer aldığında minimum istasyon sayısının elde edilmesi daha mümkündür.

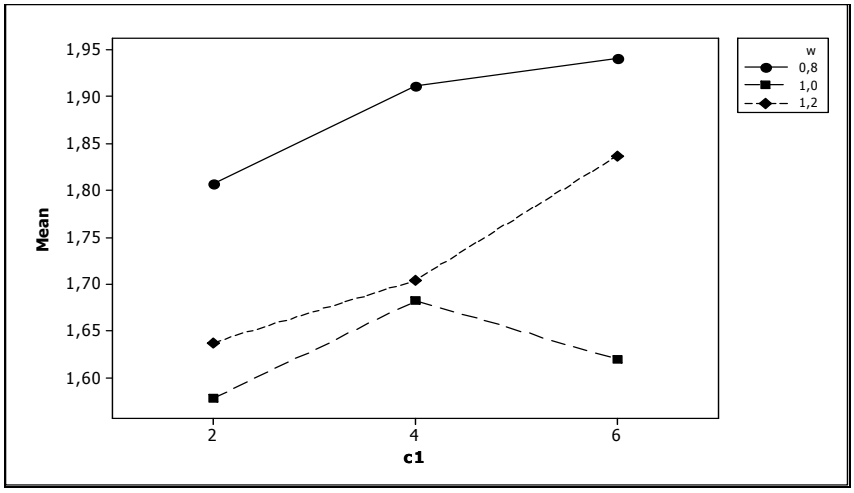


Şekil 4.16. Parçacık sayısının minimum istasyon sayısına etkisi

Şekil 4.16 elde edilen minimum istasyon sayısı ile parçacık sayısı arasındaki ilişkiyi göstermektedir. Şekil 4.16'ya göre, parçacık sayısı görev sayısının 2 ve 3 katı alındığında minimum istasyon sayısının elde edilmesi daha mümkündür.

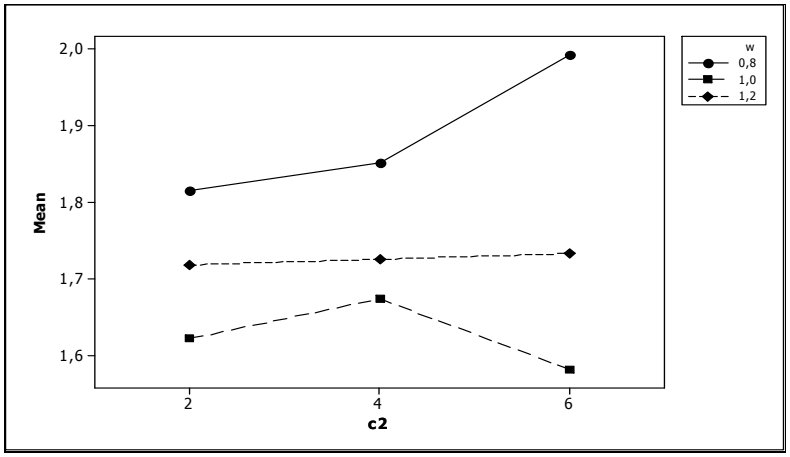
4.5.2. PSO parametrelerinin optimal çözüm üzerindeki ikili etkileri

PSO parametrelerinin, minimum istasyon sayısının bulunmasındaki ikili etkileri 10 ayrı şekille incelenmiştir.



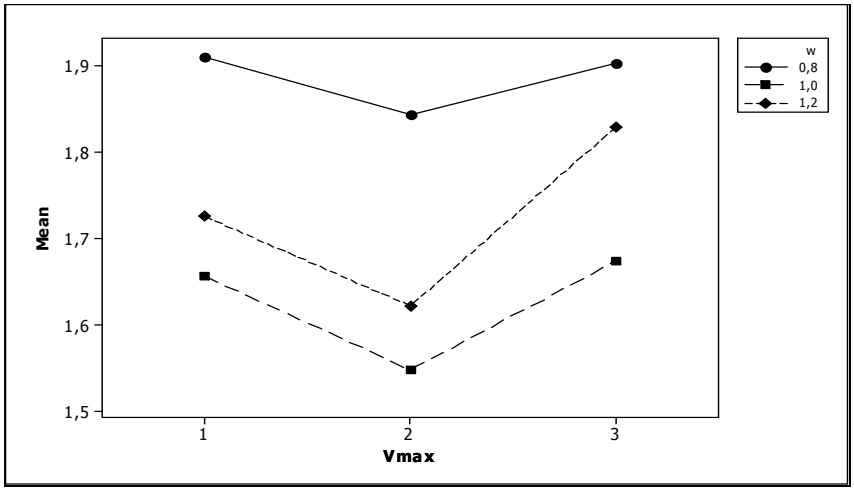
Şekil 4.17. w ve c_1'in minimum istasyon sayısına etkisi

Şekil 4.17 w ve c_1'in elde edilen minimum istasyon sayısı üzerindeki etkisini göstermektedir. Şekil 4.17'ye göre, w 1 ve 1.2 değerlerini alırken c_1 2 değerini alırsa; w 1 değerini alırken c_1 6 değerini alırsa minimum istasyon sayısının elde edilmesi daha mümkündür.



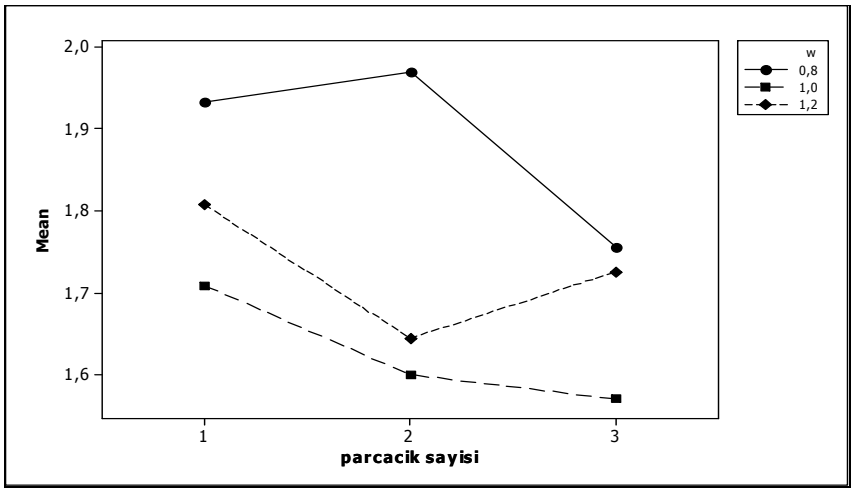
Şekil 4.18. w ve c_2'nin minimum istasyon sayısına etkisi

Şekil 4.18 w ve c_2 elde edilen minimum istasyon sayısı üzerindeki etkisini göstermektedir. Şekil 4.18'e göre, w 1 değerini alırken c_2 2, 4 ve 6 değerini alırsa minimum istasyon sayısının elde edilmesi daha mümkündür.



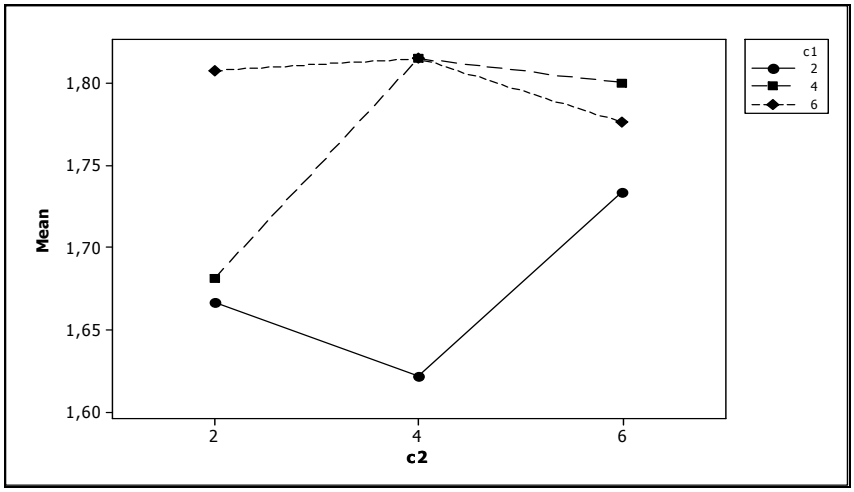
Şekil 4.19. w ve V_{max}'ın minimum istasyon sayısına etkisi

Şekil 4.19 w ve V_{max}'ın elde edilen minimum istasyon sayısı üzerindeki etkisini göstermektedir. Şekil 4.19'a göre, w 1 değerini alırken V_{max} alt sınıra eşit alınırsa minimum istasyon sayısının elde edilmesi daha mümkündür.



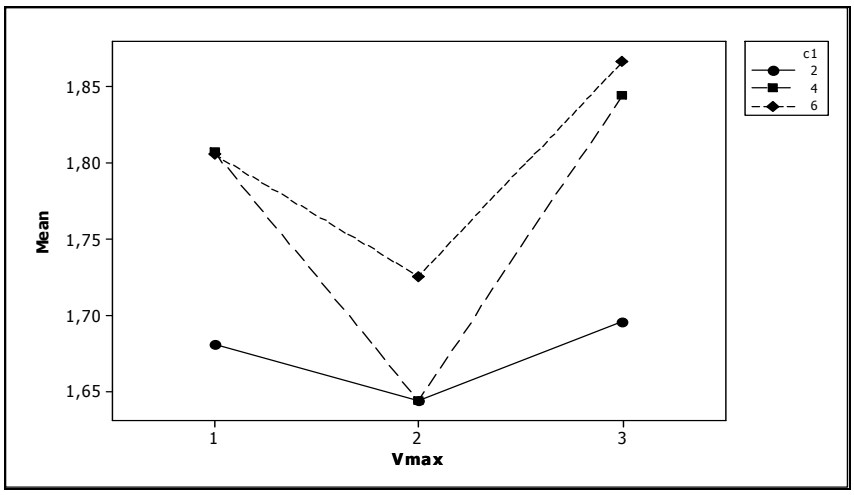
Şekil 4.20. w ve parçacık sayısının minimum istasyon sayısına etkisi

Şekil 4.20 w ve parçacık sayısının elde edilen minimum istasyon sayısı üzerindeki etkisini göstermektedir. Şekil 4.20'ye göre, w 1 değerini alırken parçacık sayısı görev sayısının 2 ve 3 katı alınırsa; w 1.2 değerini alırken parçacık sayısı görev sayısının 2 katı alınırsa minimum istasyon sayısının elde edilmesi daha mümkündür.



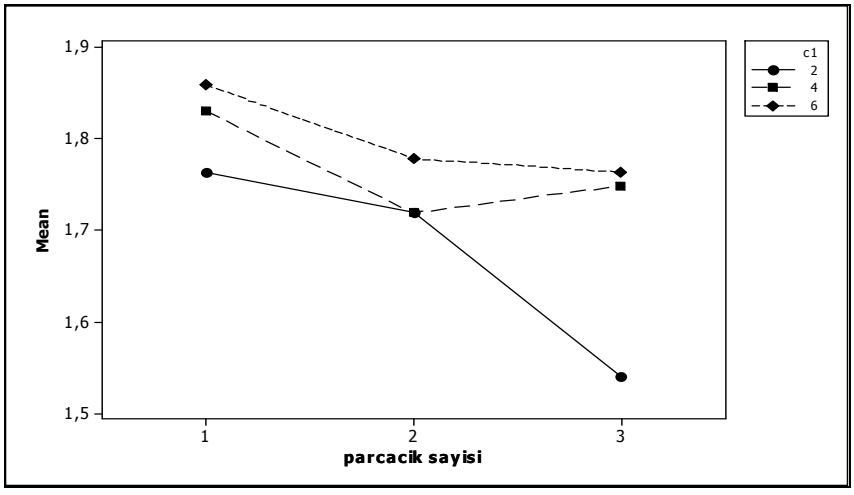
Şekil 4.21. c_1 ve c_2'nin minimum istasyon sayısına etkisi

Şekil 4.21 c_1 ve c_2'nin elde edilen minimum istasyon sayısı üzerindeki etkisini göstermektedir. Şekil 4.21'e göre, c_1 2 değerini alırken c_2 2 ve 4 değerini alırsa; c_1 4 değerini alırken c_2 2 değerini alırsa minimum istasyon sayısının elde edilmesi daha mümkündür.



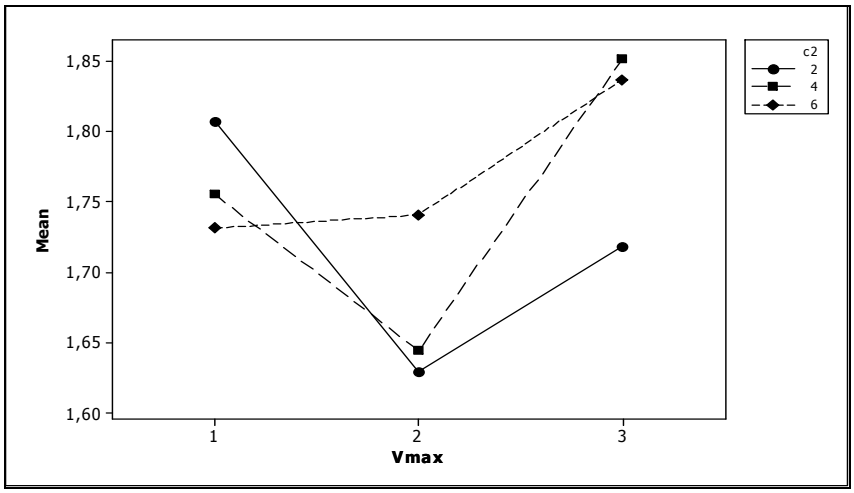
Şekil 4.22. c_1 ve V_{max}'ın minimum istasyon sayısına etkisi

Şekil 4.22 c_1 ve V_{max}'ın elde edilen minimum istasyon sayısı üzerindeki etkisini göstermektedir. Şekil 4.22'ye göre, c_1 2 değerini alırken V_{max} alt sınırın yarısı, alt sınır ve alt sınırın 2 katı alınırsa; c_1 4 değerini alırken V_{max} alt sınır kadar alınırsa minimum istasyon sayısının elde edilmesi daha mümkündür.



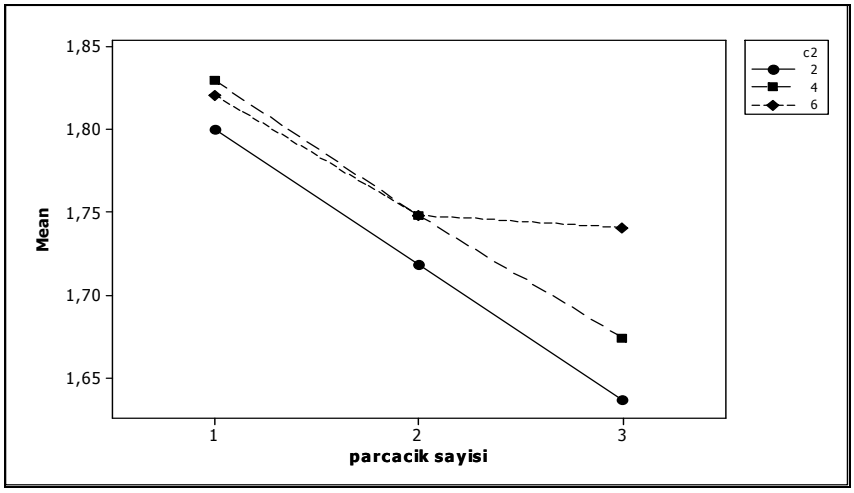
Şekil 4.23. c_1 ve parçacık sayısının minimum istasyon sayısına etkisi

Şekil 4.23 c_1 ve parçacık sayısının elde edilen minimum istasyon sayısı üzerindeki etkisini göstermektedir. Şekil 4.23'e göre, c_1 2 değerini alırken parçacık sayısı görev sayısının 3 katı alınırsa minimum istasyon sayısının elde edilmesi daha mümkündür.



Şekil 4.24. c_2 ve V_{max}'ın minimum istasyon sayısına etkisi

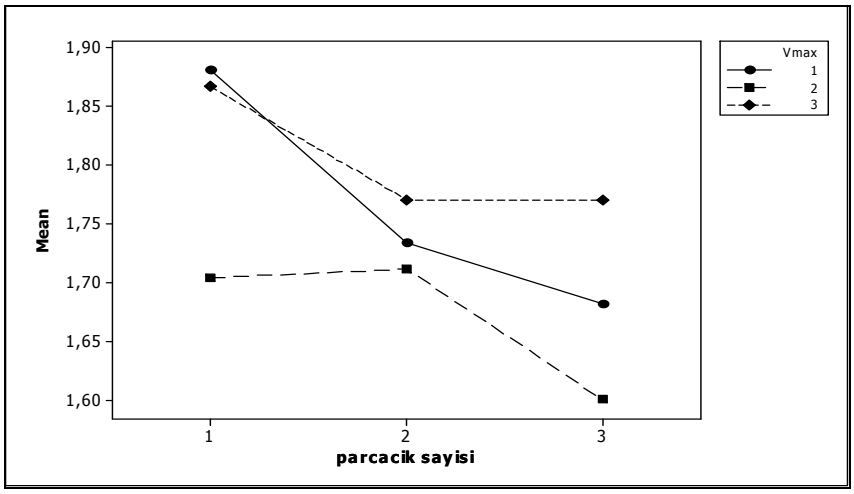
Şekil 4.24 c_2 ve V_{max}'ın elde edilen minimum istasyon sayısı üzerindeki etkisini göstermektedir. Şekil 4.24'e göre, c_2 2 ve 4 değerini alırken V_{max} alt sınır kadar alınırsa minimum istasyon sayısının elde edilmesi daha mümkündür.



Şekil 4.25. c_2 ve parçacık sayısının minimum istasyon sayısına etkisi

Şekil 4.25 c_2 ve parçacık sayısının elde edilen minimum istasyon sayısı üzerindeki etkisini göstermektedir. Şekil 4.25'e göre, c_2 2 ve 4 değerini alırken parçacık sayısı

görev sayısının 3 katı kadar alınırsa minimum istasyon sayısının elde edilmesi daha mümkündür.



Şekil 4.26. V_{max} ve parçacık sayısının minimum istasyon sayısına etkisi

Şekil 4.26 V_{max} ve parçacık sayısının elde edilen minimum istasyon sayısı üzerindeki etkisini göstermektedir. Şekil 4.26'ya göre, V_{max} alt sınır kadar değer alırken parçacık sayısı görev sayısının 3 katı kadar alınırsa minimum istasyon sayısının elde edilmesi daha mümkündür.

5. SONUÇ

Bu çalışmada görev zamanlarının normal dağılımdan gelen bağımsız rassal değişkenler olduğu varsayımıyla stokastik U-tipi montaj hattı dengeleme problemi için bir parçacık sürü optimizasyonu algoritması önerilmiştir.

Normal dağılımın yapısı gereği her görev için bir ortalama değere ve bir de varyansa ihtiyaç vardır. Görevlerin ortalaması (μ_i), literatürdeki deterministik problemlerin süreleri olarak, varyansı ise $[0,(\mu_i/2)^2]$ arasında olduğunda yüksek varyans ve $[0,(\mu_i/4)^2]$ arasında olduğunda düşük varyans olarak alınmıştır. Görevlerin tamamlanma olasılığı ise 0.90, 0.95 ve 0.975 olarak dikkate alınmıştır.

Önerilen algoritmanın etkinliğini görebilmek için öncelikle 45 görevli problemlere kadar olan literatürdeki test problemlerinde önerilen PSO algoritması çalıştırılmıştır. Bulunan sonuçlar daha önce SUMHDP-1 problemini ele alan Urban ve Chiang'ın [2006] ve Baykasoğlu ve Özbakır'ın [2007] sunduğu sonuçlarla karşılaştırılmıştır. Önerilen PSO algoritması düşük varyans için iki problemde ve yüksek varyans için iki problemde diğerlerine nazaran daha iyi sonuç bulabilmiştir. Sadece bir problemde ise Urban ve Chiang'ın [2006] sunduğu sonuçlardan daha iyi bir sonuca erişememiştir. Burada dikkate alınan problemin stokastik olması ve görev zamanlarının varyanslarının değişkenlik göstermesi sebebiyle, bazen farklı sonuçların ortaya çıkması mümkündür. Ancak bu problemler defalarca çalıştırılarak en fazla bulunan sonuçlara yer verilmiştir.

Ayrıca önerilen PSO algoritmasının ortalama CPU süresine bakıldığında diğer iki çalışmanın ortalama sürelerinden daha kısa olduğu görülmektedir.

Daha sonra görev sayısı 45'ten fazla olan test problemleri için de önerilen algoritma çalıştırılmıştır. Toplam 911 problemin %15.26'sında alt sınır bulunmuştur. Ele alınan problemlerin deterministik problemler için ortaya atılmış olması ve normal dağılım

gereği dikkate alınan problemlerde doğrusal olmayan bir yapı ortaya çıkması sebebiyle bu oranın iyi olduğu değerlendirilmektedir.

Bu çalışmada bulunan sonuçların bazıları alt sınırdan çok uzak gibi görünmektedir. Test edilen problemlerde en iyi sonuç alt sınıra eşit olan sonuçtur. Ancak her zaman alt sınırı bulmak mümkün görünmemektedir.

Sonuçlar incelendiğinde görülmektedir ki varyans arttıkça alt sınır ve elde edilen sonuç arasındaki fark artmaktadır. Yine aynı şekilde çevrim süresi azaldıkça alt sınır ve elde edilen sonuç arasındaki fark artabilmektedir.

Önerilen PSO algoritmasında yer alan parametrelerin (w, c_1, c_2, V_{max}, parçacık sayısı) minimum istasyon sayısının (optimal sonucun) bulunmasında etkili olduğunu bulabilmek amacıyla, parametrelerin ana etki ve ikili etkileri incelenmiştir.

Genel olarak değerlendirildiğinde PSO, her iterasyonda parçacık sayısı kadar farklı çözüm üretebildiğinden optimal çözümün bulunmasında çok hızlı bir algoritmadır. Özellikle kesikli optimizasyon problemlerinde optimal çözüm yerine eğer sadece herhangi bir çözüm aranırsa, iterasyon ve parçacık sayısı kadar algoritma çalıştırılarak herhangi bir sonuç elde edilebilmektedir. PSO'nun şimdiye kadar sürekli optimizasyonda daha etkili olduğu ve kesikli optimizasyon için geliştirilmesinde fayda olduğu değerlendirilmektedir.

PSO'nun dezavantajlarından biri ise her problem için PSO parametreleri, atalet ağırlığı (w), parçacığın tecrübesi (c_1), sürü tecrübesi (c_2), maksimum hız (V_{max}), parçacık sayısı ve iterasyon sayısının teker teker girilmek durumunda kalınmasıdır. Problemin büyüklüğüne, karmaşıklığına göre bu parametreler değişebilmektedir.

Bundan sonraki yapılacak çalışmalarda aşağıda belirtilen hususlar değerlendirmeye alınabilir:

- Literatürde montaj hattı dengeleme problemlerini stokastik ele alan çalışmaların çoğunun görev zamanlarının normal dağılıma uyduğu varsayımını dikkate aldığı görülür. Az da olsa görev zamanlarının diğer dağılımlara uyduğunu varsayan çalışmalar da mevcuttur. Bundan sonra yapılacak çalışmalarda özellikle stokastik U-tipi montaj hattı dengeleme problemi için görev zamanlarının diğer dağılımlara uyduğunu varsayan çalışmalar yapılabilir.

- PSO algoritması, istasyon sayısı verildiğinde çevrim zamanının minimum yapılmasını amaçlayan SUMHDP-2 problemleri için uygulanabilir.

- PSO algoritması, hem çevrim zamanını hem de istasyon sayısını minimum yapmayı amaçlayan SUMHDP-E problemleri için uygulanabilir.

- Ayrıca literatürde yeni yer bulan paralel MHD problemlerinde PSO algoritması kullanılabilir.

KAYNAKLAR

Acar, N., "Üretim planlaması yöntem ve uygulamaları", ***MPM Yayınları***, Ankara, 9-15 (1996).

Ağpak, K., "Tek modelli basit U-tipi montaj hattı dengeleme problemleri için sezgisel metot", Y.Lisans Tezi, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 37-41 (2001).

Ağpak, K. and Gökçen, H., "Assembly line balancing: two resource constrained cases, ***International Journal of Production Economics***, 96:129–140 (2005).

Ağpak, K. and Gökçen H., "A chance-constrained approach to stochastic line balancing, ***European Journal Of Operational Research***, 180:1098–1115 (2007).

Ağpak, K., Gökçen, H., Saray, N. ve Özel, S., "Stokastik görev zamanlı tek modelli U tipi montaj hattı dengeleme problemleri için bir sezgisel", ***Journal of The Faculty of Engineering and Architecture of Gazi University***, 17(4):115-124 (2002).

Ai, T.J. and Kachitvichyanukul, V. "A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery", ***Computers&Operations Research***, 36(5):1693-1702 (2009).

Ajenblit, D.A. and Wainwright, R.L., "Applying genetic algorithms to the U-shaped assembly line balancing problem, ***IEEE International Conference on Evolutionary Computation***, 96–101 (1998).

Andres, C. and Lozano, S., "A particle swarm optimization algorithm for part-machine grouping", ***Robotics and Computer-Integrated Manufacturing***, 468-474 (2006).

Arcus, A.L., "COMSOAL A computer method of sequencing operations for assembly lines", ***International Journal of Production Research***, 4(4):559-277 (1966).

Bautista, J. and Pereira, J., "Ant algorithms for a time and space constrained assembly line balancing problem", ***European Journal of Operational Research***, 177:2016-2032 (2007).

Bautista, J., Suárez, R., Mateo, M. and Companys, R., "Local search heuristics for the assembly line balancing problem with incompatibilities between tasks", ***Proceedings of the 2000 IEEE International Conference on Robotics & Automation***, San Francisco, 2404-2409 (2000).

Baybars, I., "A survey of exact algorithms for the simple assembly line balancing problem", ***Management Science***, 32(8):909-932 (1986).

Baykasoğlu, A. and Özbakır, L., "Stochastic U-line balancing using genetic algorithms", ***The International Journal of Advanced Manufacturing Technology***, 32:139–147 (2007).

Becker, C. and Scholl, A., "A survey on problems and methods in generalized", ***European Journal of Operational Research***, 168(3):694-715 (2006).

Benzer, R., "Paralel montaj hattı dengeleme problemi için yeni modeller", Doktora Tezi, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 4-10 (2005).

Benzer, R., Gökçen, H., Çetinyokuş, T. and Çerçioğlu, H., "A network model parallel line balancing problem", ***Mathematical Problems in Engineering***, 10106, 1-12 (2007).

Bergh, F.V. and Engelbecht, A.P., "Cooperative learning in neural networks using particle swarm optimizers", ***South African Computer Journal***, 26:84–90 (2000).

Bowman, E.H., "Assembly line balancing by linear programming", ***Operations Research***, 8(3):385-389 (1960).

Carraway, R.L., "A dynamic programming approach to stochastic assembly line balancing", ***Management Science***, 45(4):459-471 (1989).

Chand, S. and Zeng, T., "A comparison of U-line and straight-line performances under stochastic task times", ***Manufacturing & Service Operations Management***, 3(2):138–150 (2001).

Chase, R.B., Aquilano, N.J. and Jacobs, F.R., "Operations management for competitive advantage", 9th Ed., ***McGraw-Hill Company***, 181-182 (2001).

Chakravarty, A.K. and Shtub, A., "Balancing mixed model lines with in-process inventories", ***Management Science***, 31:1161–1174 (1985).

Chiang, W-C. and Urban, T.L., "The stochastic U-line balancing problem: a heuristic procedure", ***European Journal of Operational Research***, 175:1767–1781 (2006).

Clerc, M., "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization", ***Proc. Congress of Evolutionary Computation,*** IEEE, NJ, 1951-1957 (1999).

Clerc, M. and Kennedy, J., "The particle swarm—explosion, stability, and convergence in a multidimensional complex space", ***IEEE Transaction Evolutionary Computation***, 6:58-73 (2002).

Dar-El, E.M., "MALB-A heuristic technique for balancing large single-model assembly lines", ***AIIE Transactions***, 5(4):343-356 (1973).

Dar-El, E.M and Rubinovitch, Y. "Using learning theory in assembly lines for new products", ***International Journal of Production Economics***, 25:102-109 (1991).

Eberhart, R. and Kennedy, J., "A new optimizer using particle swarm theory", ***Sixth International Symposium on Micro Machine and Human Science***, 39-43 (1995).

Erel, E. and Sarin, S.C., "A survey of the assembly line balancing procedures", ***Production Planning and Control***, 9(5):414-434 (1998).

Erel, E. and Gökçen, H., "Shortest route formulation of mixed-model assembly line balancing problem", ***European Journal of Operational Research***, 116:194–204 (1999).

Erel, E., Sabuncuoğlu, I. and Aksu, B.A., "Balancing of U-type assembly systems using simulated annealing", ***International Journal of Production Research***, 39(13):3003-3015 (2001).

Gao, J., Sun, L. and Gen, M., "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems", ***Computers&Operations Research***, 35(9):2892-2907 (2008).

Ghosh, S. and Gagnon, J., "A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems", ***International of Production Research***, 27(4):637-670 (1989).

Gökçen, H., "Karışık modelli montaj hattı dengeleme problemleri için yeni modeller", Doktora Tezi, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 5-13 (1994).

Gökçen, H. and Ağpak, K., "A goal programming approach to simple U-Line balancing problem", ***European Journal of Operational Research***, 171:577–585 (2006).

Gökçen, H., Ağpak K. and Benzer, R., **"**Balancing of parallel assembly lines", ***International Journal of Production Economics***, 103(2):600-609 (2006).

Gökçen, H. and Baykoç, Ö.F., "A new line remedial policy for the paced lines with stochastic task times", ***International Journal of Production Economics***, 58:191-197 (1999).

Gökçen, H. and Erel, E., "Binary integer formulation for mixed-model assembly line balancing problem", ***Computers and Industrial Engineering***, 34:451–461 (1998).

Gökçen, H. and Erel, E., "A goal programming approach to mixed-model assembly line balancing problem", ***International Journal Of Production Economics***, 48(2):177-185 (1997).

Gökçen, H., Ağpak, K., Gencer, C. and Kizilkaya, E., "A shortest route formulation of simple U-type assembly line balancing problem", ***Applied Mathematical Modeling***, 29:373–380 (2005).

Guerriero, F. and Miltenburg, J., "The stochastic U-line balancing problem", ***Naval Research Logistics***, 50(1):31-57 (2003).

Gunter, R.E., Johnson, G.D. and Peterson, R.S., "Currently practiced formulations for the assembly line balance problem", ***Journal of Operations Management***, 3(4):209-221 (1983).

Held, M., Karp, R.M. and Shareshian, R., "Assembly line balancing: dynamic programming with precedence constraints", ***Operations Research***, 11(3):442-459 (1963).

Jackson, J.R., "A computing procedure for a line balancing problem", ***Management Science***, 2(3):261-271 (1956).

Jerald J., Asokan P., Prabaharan G. and Saravanan R., "Scheduling optimisation of flexible manufacturing systems using particle swarm optimization algorithm", ***International Journal of Advanced Manufacturing Technology***, 25:964-971 (2005).

Johnson, R.V., "Assembly line balancing algorithms: computation comparisons", ***International Journal of Production Research***, 19(3):277-287 (1981).

Kao, E.P.C., "A preference order dynamic program for stochastic assembly line balancing", ***Management Science***, 22:1097–1104 (1976).

Kao, E.P.C., "Computational experience with a stochastic assembly line balancing algorithm", ***Computers and Operations Research***, 6:79–86 (1979).

Kao, E.P.C. and Queyranne, M., "On dynamic programming methods for assembly line balancing", ***Operations Research***, 30(2):375-390 (1982).

Kennedy, J. and Eberhart, R.C., "Particle swarm optimization", ***Proc. of the IEEE International Conference on Neural Network***, 1942-1948 (1995).

Kennedy, J. and Eberhart, R.C., "A discrete version of the particle swarm algorithm", ***Proc. of the Con. on Systems, Man and Cybernetics SMC97***, 4104-4109 (1997).

Kennedy J., Eberhart R.C. and Shi Y. "Swarm intelligence", San Francisco, ***CA: Morgan Kaufmann***, 15-50 (2001).

Kennedy, J. and Mendes, R., "Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms", ***Proc. IEEE Int. Workshop on Soft Computing in Industrial Applications***, 45-50 (2003).

Killbridge, M.D. and Wester, L., "A heuristic method of assembly line balancing", ***Journal of Industrial Engineering***, 12(4):51-58 (1961).

Kim, Y.K., Kim, Y. and Kim, Y.J., "Two-sided assembly line balancing: a genetic algorithm approach", ***Production Planning and Control***, 11:44–53 (2000).

Klein, M., "On assembly line balancing", ***Operations Research***, 11:274-281 (1963).

Kottas, J.F. and Lau, H.S., "A cost-oriented approach to stochastic line balancing", ***AIIE Transactions***, 5:164–171 (1973).

Kottas, J.F. and Lau, H.S., "A total operating cost model for paced lines with stochastic task times", ***AIIE Transactions***, 8:234–240 (1976).

Lapierre, S.D., Ruiz, A. and Soriano, P., "Balancing assembly lines with tabu search", ***European Journal of Operational Research***, 168:826-837 (2006).

Laskari, E.C., Parsopoulos, K.E. and Vrahatis, M.N., "Particle swarm optimization for integer programming", ***Proceedings of the IEEE***, 1582-1587 (2002).

Lian, Z., Gu, X. and Jiao, B., "A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan", ***Chaos, Solitons & Fractals***, 35(5):851-861 (2008).

Lian, Z., Jiao, B. and Gu, X., "A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan", ***Applied Mathematics and Computation***, 183(2):1008-1017 (2006).

Liao, C-J., Tseng, C-T. and Luarn, P., "A discrete version of particle swarm optimization for flowshop scheduling problems", ***Computers & Operations Research***, 34(10):3099-3111 (2007).

Liu, S.B., Ong, H.L. and Huang, H.C., "A bidirectional heuristic for stochastic assembly line balancing type II problem", ***International Journal Advanced Manufacturing Technology***, 25:71–77 (2005).

Liu, B., Wang, L. and Jin Y., "An effective hybrid pso-based algorithm for flow shop scheduling with limited buffers", ***Computers&Operations Research***, 35(9):2791-2806 (2008).

Macaskill, J.L.C., "Production-line balances for mixedmodel lines", ***Management Science***, 19:423–434 (1972).

McMullen, P.R. and Frazier, G.V., "Using simulated annealing to solve a multi-objective assembly line balancing problem with parallel stations", ***International Journal of Production Research***, 36:2717–2741 (1998).

McMullen, P.R. and Tarasewich, P., "Using ant techniques to solve the assembly line balancing problem", ***IIE Transactions***, 35:605–617 (2003).

Merkle, D. and Middendorf, M., "Search methodologies", ***Springer US***, 401-435 (2005).

Meyers, F.E. and Stephens, M.P., "Manufacturing facilities design and material handling", 3rd Ed., ***Pearson Prentice-Hall***, New Jersey, 109-112 (2005).

Miltenburg, J. and Wijngaard, J., "The U-line balancing problem", ***Management Science***, 40(10):1378-1388 (1994).

Miltenburg, J., "Balancing U-lines in a multiple u-line facility", ***European Journal of Operational Research***, 109:1–23 (1998).

Miltenburg, J., "U-shaped production lines: a review of theory and practice", ***International Journal of Production Economics***, 70(3):201–214 (2001).

Mohemmed, A.W., Sahoo, N.C. and Geok, T.K., "Solving shortest path problem using particle swarm optimization", ***Applied Soft Computing***, 8(4):1643-1653 (2008).

Moodie, C.L. and Young, H.H., "A heuristic method of assembly line balancing for assumptions of constant or variable work element times", ***Journal of Industrial Engineering***, 16(1):23–29 (1965).

Nahmias, S., "Production and operations management", 5th Ed., ***McGraw-Hill Irwin***, 439-440 (2005).

Nakade, K. and Ohno, K., "An optimal worker allocation for a u-shaped production line", ***International Journal of Production Economics***, 60–61:353–358 (1999).

Nakade, K., Ohno, K. and Shanthikumar, J.G., "Bounds and approximations for cycle times of a U-shaped production line", ***Operations Research Letters***, 21:191-200 (1997).

Nkasu, M.M. and Leung, K.H., "A stochastic approach to assembly line balancing", ***International Journal of Production Research***, 33:975–991 (1995).

Ohno, K. and Nakade, K., "Analysis and optimization of U-shaped production line", ***Journal of the Operation Research Society of Japan***, 40(1):90-104 (1997).

Ohno, K., Nakade, K. and Shantikumar, J.G., "Bounds and approximations for cycle times of a U-shaped production line", ***Operations Research Letters***, 21:191–200 (1997).

Önüt, S., Tuzkaya, U.R. and Doğaç, B., "A particle swarm optimization algorithm for the multiple-level warehouse layout design problem", ***Computers & Industrial Engineering***, 54:783–799 (2008).

Özcan, E. and Mohan, C.K., "Particle swarm optimization: surfing the waves", ***Proc. Congress on Evolutionary Computation***, 1939–1944 (1999).

Pan, Q., Taşgetiren, M.F. and Liang, Y., "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem", ***Computers & Operations Research***, 35(9):2807-2839 (2008).

Patterson, J.H. and Albracht, J.J., "Assembly line balancing: 0-1 programming with fibonacci search", ***Operations Research***, 23:166-174 (1975).

Sabuncuoğlu, İ., Erel, E. and Tanyer, M., "Assembly line balancing using genetic algorithms", ***Journal of Intelligent Manufacturing***, 11:295-310 (2000).

Salman, A., Ahmad, I. and Al-Madani, S., "Particle swarm optimization for task assignment problem", ***Microprocessors and Microsystems***, 363-371 (2002).

Salveson, M.E., "The assembly line balancing problem", ***Journal of Industrial Engineering***, 6(6):18-25 (1955).

Sarin, S.C. and Erel, E., "Development of cost model for the single-model stochastic assembly line balancing problem", ***International Journal of Production Research***, 28:1305–1316 (1990).

Sarin, S.C., Erel, E. and Dar-El, E.M., "A methodology for solving single-model, stochastic assembly line balancing problem", ***Omega***, 27:525–535 (1999).

Scholl, A., "Balancing and sequencing of assembly lines", Second Ed., ***Physica-Verlag, Heidelberg***, 15-30 (1999).

Scholl, A. and Becker, C., "State-of-the art exact and heuristic solution procedures for simple assembly line balancing", ***European Journal of Operational Research***, 168(3):666-693 (2006).

Scholl, A. and Boysen, N., "Designing parallel assembly lines with split workplaces: Model and optimization procedures", International ***Journal of Production Economics***, 119;90-100 (2009).

Scholl, A. and Klein, R., "SALOME: A bidirectional branch and bound procedure for assembly line balancing", ***INFORMS Journal on Computing***, 9:319–334 (1997).

Scholl A, and Klein R., "ULINO: Optimally balancing U-Shaped JIT assembly lines", ***International Journal of Production Research***, 37(4):721–736 (1999).

İnternet: Scholl'ün internet sitesi, "Standard problems", www.assembly-line-balancing.de (2008)

Sha, D.Y. and Hsu, C., "A hybrid particle swarm optimization for job shop scheduling problem", ***Computers & Industrial Engineering***, 51:791–808 (2006).

Shi, Y. and Eberhart, R., "A modified particle swarm optimizer", ***IEEE International Conference on Evolutionary Computation***, Anchorage, Alaska, 69-73 (1998).

Shi, Y. and Eberhart, R.C., "Empirical study of particle swarm optimization", ***Proceeding of the IEEE Congress on Evolutionary Computation***, 1945-1950 (1999).

Shi, X.H., Xing, X.L., Zhang, L.H., Yang, X.W., Zhou, C.G. and Liang, Y.C., "A discrete PSO method for generalized TSP problem", ***Proceedings of the Third International Conference on Machine Learning and Cybernetics***, Shanghai, 2378-2383 (2004).

Sniedovich, M., "Analysis of a preference order assembly line problem", ***Management Science***, 27:1067–1080 (1981).

Sparling, D. and Miltenburg, J., "Mixed-model U-Line balancing", ***International Journal of Production Research***, 36:485–501 (1998).

Suresh, G. and Sahu, S., "Stochastic assembly line balancing using simulated annealing", ***International Journal of Production Research***, 32:1801–1810 (1994).

Suresh, G., Vinod, V.V. and Sahu, S, "Genetic algorithm for assembly line balancing", ***Production Planning and Control***, 7:38–46 (1996).

Şevkli, M. ve Yenisey, M.M., "Atölye tipi çizelgeleme problemleri için parçacik sürü optimizasyonu yöntemi", ***itü dergisi***, 5(2):58-68 (2006).

Taşgetiren, M.F. and Liang, Y., "A binary particle swarm optimization algortihm for lot sizing problem", ***Journal of Economic and Social Research***, 5(2):1-20 (2003).

Tasgetiren, M.F., Liang, Y-C, Sevkli, M. and Gencyilmaz, G., "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem", ***European Journal of Operational Research***, 177(3):1930-1947 (2007).

Tasgetiren, M.F., Şevkli, M., Liang, Y.C. and Gençyılmaz, G., "Particle swarm optimization algorithm for permutation flowshop sequencing problem", ***Lecture Notes in Computer Science***, Springer-Verlag, 3172, 382–390 (2004).

Toksarı, M.D., İşleyen, S.K., Güner, E. and Baykoç, Ö.F., "Simple and U-type assembly line balancing problems with a learning effect", ***Applied Mathematical Modelling***, 32(12):2954-2961 (2008).

Tonge, F.M., "Summary of a heuristic line balancing procedure", ***Management Science***, 7(1):21-42 (1960).

Xia, W. and Wu, Z., "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems", ***Computers & Industrial Engineering***, 48(2):409-425 (2005).

Xu, Y., Chen, G. and Yu, J., "Three sub-swarm discrete particle swarm optimization algorithm", ***International Conference on Information Acquisition***, China, August 20-23 (2006).

Wang, K.P., Huang, L., Zhou, C.G. and Pang, W., "Particle swarm optimization for traveling salesman problem", ***Proceedings of the Second International Conference on Machine Learning and Cybernetics***, Xian, 1583-1585 (2003).

Urban, T.L., "Note: Optimal balancing of U-shaped assembly lines", ***Management Science***, 44(5):738–741 (1998).

Urban, T.L. and Chiang, W-C., "An optimal piecewise-linear program for the U-line balancing problem with stochastic task times", ***European Journal of Operational Research***, 168:771–782 (2006).

Üreten, S., "Üretim/işlemler yönetimi", ***Başar Ofset***, 2. Baskı, Ankara, 5-15 (1999).

Zeng, X., Zhu, Y., Nan, L., Hu, K., Niu, B. and He, X., "Solving weapon-target assignment problem using discrete particle swarm optimization", ***Proceedings of the 6th World Congress on Intelligent Control and Automation***, China, 3562-3565 (2006).

Zhu, Q., Qian, L., Li, Y. and Zhu, S., "An improved particle swarm optimization algorithm for vehicle routing problem with time windows", ***IEEE Congress on Evolutioniary Computation***, Canada, 1386-1390 (2006).

Zobolas, G.I., Tarantilis, C.D. and Ioannou, G., "Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm", ***Computers & Operations Research***, 36(4):1249-1267 (2009).

EKLER

EK-1 SUMHDP-1 için PSO algoritması delphi kodu

```
unit frm1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, IniFiles, StrUtils, Math, Spin,
Mask;

type
  TfrmTanim = class(TForm)
    GroupBox1: TGroupBox;
    txtZ: TEdit;
    rgVaryans: TRadioGroup;
    rgVaryansDusuk: TRadioButton;
    rgVaryansYuksek: TRadioButton;
    txtVmin: TEdit;
    txtVmax: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    GroupBox2: TGroupBox;
    btnDosyaYukle: TButton;
    GroupBox4: TGroupBox;
    btnPSO: TButton;
    txtDosya: TMemo;
    txtSonCozum: TMemo;
    Label13: TLabel;
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
    txtSonistsayi: TEdit;
    txtIterasyon: TSpinEdit;
    txtParcacikSayi: TSpinEdit;
    txtCevrimSure: TSpinEdit;
    txtW: TSpinEdit;
    txtc1: TSpinEdit;
    txtc2: TSpinEdit;
    Label11: TLabel;
    txtGorevSayi: TEdit;
    Label7: TLabel;
    txtCPUTime2: TEdit;
    Label14: TLabel;
    txtLB: TEdit;
    rgVaryansYok: TRadioButton;
    procedure btnDosyaYukleClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnPSOClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
TCloseAction);
  private
    LKimden : THashedStringList;
    LKime : THashedStringList;
    LKimden1 : THashedStringList;
    LKime1 : THashedStringList;
    LOnculArdil1 : THashedStringList;
    LB : integer;
    Z : double;
    C : double;
    TM : double;
    TS2 : double;
    PS : integer;
    w :double;
    c1,c2 :double;
    Vmin : double;
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
    Vmax : double;
    MaxIter : integer;
    MaxGrvSure : double;
    GS : Integer;
    GB1 : integer;
    GL :Array of double;
    S2 :Array of double;
    CZ1 : Array of array of array of integer;
    PB1 : Array of integer;
    F1 : Array of array of double;
    V1 : Array of array of double;
    time1 :TDateTime;
    time2 :TDateTime;
  public
        { Public declarations }
    procedure p_tanimlari_oku;
    function f_baslangic_cozumu_bul : integer;
  end;

var
  frmTanim: TfrmTanim;
implementation
{$R *.dfm}

procedure TfrmTanim.p_tanimlari_oku;
var
  i,j: integer;
  vr1 : double;
begin
        GS:=0;
        MaxGrvSure := 0;
        C := StrToFloat(txtCevrimSure.Text);
        Z := StrToFloat(txtZ.Text);
        PS := StrToInt(txtParcacikSayi.Text);
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
w := StrtoFloat(txtW.Text);
c1 := StrToFloat(txtc1.Text);
c2 := StrToFloat(txtc2.Text);
LB := 0;
TM := 0;
TS2 := 0;
Vmin := StrToFloat(txtVmin.Text);
Vmax := StrToFloat(txtVmax.Text);
MaxIter := StrToInt(txtIterasyon.Text);
txtGorevSayi.Text :='0';
txtGorevSayi.Text := txtDosya.Lines[0];
GS :=    StrToInt(txtGorevSayi.Text);
if rgVaryansDusuk.Checked then
  vr1 := 4.0;
if rgVaryansYuksek.Checked then
  vr1:= 2.0;
if rgVaryansYOK.Checked then
  vr1:= 0.0;
SetLength(GL, GS+3);
SetLength(S2, GS+3);
SetLength(CZ1, MaxIter+3);
SetLength(PB1, MaxIter+3);
SetLength(F1, MaxIter+3);
SetLength(V1, PS+3);
for i := Low(V1) to High(V1) do
begin
   SetLength(V1[i], GS+3);
end;
for i := Low(CZ1) to High(CZ1) do
begin
   SetLength(CZ1[i], PS+3);
   for j := Low(CZ1[i]) to High(CZ1[i]) do
   begin
      SetLength(CZ1[i][j], GS+3);
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
            end;
            SetLength(F1[i], PS+3);
         end;

         Randomize;
         for i:=1 to  GS do
          begin
            GL[i] := strtofloat(txtDosya.Lines[i]);
            TM := TM+ GL[i];
            if vr1=0.0 then
               S2[i] := 0.0
            else
               S2[i] :=  (GL[i]/vr1)* (GL[i]/vr1)*random;
            TS2 := TS2+ S2[i];
            if MaxGrvSure<(GL[i]+(Z*sqrt(S2[i]))) then
               MaxGrvSure := GL[i]+(Z*sqrt(S2[i]));
          end;
         i:=   GS+1;
         LKimden.Clear;
         LKime.Clear;
         while  txtDosya.Lines[i]<>'-1,-1' do
         begin

LKime.Add(midstr(txtDosya.Lines[i],pos(',',txtDosya.Lines[i])+
1,length(txtDosya.Lines[i])));
           i:=i+1;
         end;
         LB :=    Ceil((TM+(Z*sqrt(TS2)))/C);
         txtLB.Text := FloatToStr(LB);
end;

function TfrmTanim.f_baslangic_cozumu_bul : integer;
var
  p: integer;
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
  g : integer;
  k: integer;
  t : integer;
  t1 :integer;
  g1 : string;
  s1 : string;
  tiz1 : double;
  tm1 :double;
  ts21 :double;
  kc2 : double;
  minf : double;
  sonk : integer;
  iter : integer;
begin

  if MaxGrvSure > C then
  begin
      txtSonCozum.Lines.Add('Çözüm yok.');
      txtSonCozum.Lines.Add('Çevrim süresi ='+floattostr(C));
      txtSonCozum.Lines.Add('Max görev
süresi='+floattostr(MaxGrvSure));
      btnDosyaYukle.Enabled := true;
      btnPSO.Enabled := true;
      Result:=0;
      exit;
  end;

  minf := 10000000;
  GB1:=0;
  for p:=1 to PS+1 do
    for g:=1 to GS+1 do
      CZ1[0][p][g]:=0;

  for iter:=0 to MaxIter do
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
    PB1[iter] :=0;
  for iter:=0 to MaxIter do
    for p :=1 to PS+1 do
      F1[iter][p] :=0;

  p:=1;
  while (p<=PS) and ((CZ1[0][PB1[0]][GS+1]=0) or
(CZ1[0][PB1[0]][GS+1]>LB)) do
  begin
    LKimden1.Clear;
    LKime1.Clear;
    for t:=0 to LKimden.Count-1 do
    begin
        LKimden1.Add(LKimden[t]);
        LKime1.Add(LKime[t]);
    end;
    k:=1;
    sonk := k;

    while LKimden1.Count>0 do
    begin
      LOnculArdil1.Clear;
      for g:=1 to GS do
      begin
        s1:= inttostr(g);
        if LOnculArdil1.IndexOf(s1)=-1 then
        begin
          if (LKimden1.IndexOf(s1)=-1) and
(LKime1.IndexOf(s1)<>-1) then
          begin
            LOnculArdil1.Add(s1);
          end
          else  if (LKime1.IndexOf(s1)=-1) and
(LKimden1.IndexOf(s1)<>-1) then
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
          begin
            LOnculArdil1.Add(s1);
          end;
        end;
      end;
      g1:= LOnculArdil1[random(LOnculArdil1.Count)];
      tm1:=0;
      ts21 := 0;
      for g:=1 to GS do
      begin
        if CZ1[0][p][g]=k then
        begin
          tm1:=tm1+GL[g];
          ts21 := ts21 + S2[g];
        end;
      end;

      tiz1 :=
(tm1+GL[strtoint(g1)])+(Z*sqrt(ts21+S2[strtoint(g1)]));
yeni istasyon aç
      if tiz1>C then
      begin
        k:=k+1;
      end
      else
      begin
        sonk := k;
        kc2:=kc2+sqr((tm1+(Z*sqrt(ts21)))-C);
      end;
      CZ1[0][p][strtoint(g1)] := k;
      t:= LKimden1.IndexOf(g1);
      while t<>-1 do
      begin
        LKimden1.Delete(t);
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
        LKime1.Delete(t);
        t:= LKimden1.IndexOf(g1);
      end;
      t:= LKime1.IndexOf(g1);
      while t<>-1 do
      begin
        LKime1.Delete(t);
        LKimden1.Delete(t);
        t:= LKime1.IndexOf(g1);
      end;

istasyona atanmamışları da kontrol et
      if LKimden1.Count=0 then
      begin
        for g:=1 to GS do
        begin
          if    CZ1[0][p][g]=0 then
          begin
            tm1:=0;
            ts21 := 0;
            for t1:=1 to GS do
            begin
              if CZ1[0][p][t1]=k then
              begin
                tm1:=tm1+GL[t1];
                ts21 := ts21 + S2[t1];
              end;
            end;
            tiz1 := (tm1+GL[g])+(Z*sqrt(ts21+S2[g]));
            if tiz1>C then
            begin
              k:=k+1;
            end
            else
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
            begin
              sonk := k;
              kc2:=kc2+sqr((tm1+(Z*sqrt(ts21)))-C);
            end;
            CZ1[0][p][g] := k;
          end;
        end;
      end;
    end;
    if sonk < k then
    begin
      tm1:=0;
      ts21 := 0;
      for t1:=1 to GS do
      begin
        if CZ1[0][p][t1]=k then
        begin
           tm1:=tm1+GL[t1];
           ts21 := ts21 + S2[t1];
        end;
      end;
       kc2:=kc2+sqr((tm1+(Z*sqrt(ts21)))-C);
    end;
    CZ1[0][p][GS+1] := k;
    F1[0][p] :=  (CZ1[0][p][GS+1]-LB)+sqrt(kc2)/(C*
sqrt(CZ1[0][p][GS+1]));
    if minf> F1[0][p] then
    begin
      minf := F1[0][p];
      PB1[0] := p;
    end;
    p:=p+1;
  end;
  Result:=PB1[0];
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
end;

procedure TfrmTanim.btnDosyaYukleClick(Sender: TObject);
begin
  btnDosyaYukle.Enabled := false;
  btnPSO.Enabled := false;
  with TOpenDialog.Create(nil) do
  begin
    if Execute then
    begin
      txtDosya.Lines.LoadFromFile(FileName);
      btnPSO.Enabled :=  (txtDosya.Lines.Count>1);
      txtGorevSayi.Text := txtDosya.Lines[0];
      GS :=    StrToInt(txtGorevSayi.Text);
    end;
    Free;
  end;
  btnDosyaYukle.Enabled := true;
end;

procedure TfrmTanim.FormCreate(Sender: TObject);
begin
  LKimden := THashedStringList.Create;
  LKime := THashedStringList.Create ;
  LKimden1 := THashedStringList.Create;
  LKime1 := THashedStringList.Create;
  LOnculArdil1 := THashedStringList.Create;
  txtDosya.Clear;
  txtSonCozum.Clear;
end;

procedure TfrmTanim.btnPSOClick(Sender: TObject);
var
  p: integer;
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
  g : integer;
  k: integer;
  t : integer;
  t1 :integer;
  g1 : string;
  s1 : string;
  gv1 : double;
  tiz1 : double;
  tm1 :double;
  ts21 :double;
  kc2 : double;
  minf : double;
  bas1 : DWord;
  bit1 : DWord;
  sonk : integer;
  iter : integer;
  isok : integer;
  vstr : String;
begin
  btnDosyaYukle.Enabled := false;
  btnPSO.Enabled := false;
  txtSonCozum.Clear;
  txtSonistsayi.Text := '';
  txtCPUTime2.Text := '';

  Randomize;

  p_tanimlari_oku;

  time1:=now;

  k:= f_baslangic_cozumu_bul;
  if k=0 then
    exit;
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
  for p:=1 to PS do
    for g:=1 to GS do
        V1[p][g]:= Vmin + random*(Vmax - Vmin);

  iter :=1;
  GB1 :=0;
  while (iter<=MaxIter) and ((CZ1[GB1][PB1[GB1]][GS+1]=0) or
(CZ1[GB1][PB1[GB1]][GS+1]>LB)) do
  begin

   minf := 10000000;

   for p:=1 to PS+1 do
      for g:=1 to GS+1 do
         CZ1[iter][p][g]:=0;

   while (p<=PS) and ((CZ1[GB1][PB1[GB1]][GS+1]=0) or
(CZ1[GB1][PB1[GB1]][GS+1]>LB)) do
   begin
      LKimden1.Clear;
      LKime1.Clear;
      for t:=0 to LKimden.Count-1 do
      begin
            LKimden1.Add(LKimden[t]);
            LKime1.Add(LKime[t]);
      end;
      k:=1;
      sonk := k;
      kc2:=0.0;
      while LKimden1.Count>0 do
      begin
         LOnculArdil1.Clear;
         for g:=1 to GS do
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
         begin
            s1:= inttostr(g);
            if LOnculArdil1.IndexOf(s1)=-1 then
            begin
               if (LKimden1.IndexOf(s1)=-1) and
(LKime1.IndexOf(s1)<>-1) then
               begin
                  LOnculArdil1.Add(s1);
               end
               else   if (LKime1.IndexOf(s1)=-1) and
(LKimden1.IndexOf(s1)<>-1) then
               begin
                  LOnculArdil1.Add(s1);
               end;
            end;
         end;
seç
         gv1:=-99999999999.0;
         g1:='';
         for t:=0 to LOnculArdil1.Count-1 do
         begin
            if V1[p][strtoint(LOnculArdil1[t])]>gv1 then
            begin
              gv1:= V1[p][strtoint(LOnculArdil1[t])];
              g1:= LOnculArdil1[t];
            end;
         end;

         tm1:=0;
         ts21 := 0;
         for g:=1 to GS do
         begin
            if CZ1[iter][p][g]=k then
            begin
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
                tm1:=tm1+GL[g];
                ts21 := ts21 + S2[g];
             end;
          end;
          tiz1 :=
(tm1+GL[strtoint(g1)])+(Z*sqrt(ts21+S2[strtoint(g1)]));
          if tiz1>C then
          begin
             k:=k+1;
          end
          else
          begin
             sonk := k;
             kc2:=kc2+sqr((tm1+(Z*sqrt(ts21)))-C);
          end;

          t:= LKimden1.IndexOf(g1);
          while t<>-1 do
          begin
             LKimden1.Delete(t);
             LKime1.Delete(t);
             t:= LKimden1.IndexOf(g1);
          end;
          t:= LKime1.IndexOf(g1);
          while t<>-1 do
          begin
             LKime1.Delete(t);
             LKimden1.Delete(t);
             t:= LKime1.IndexOf(g1);
          end;

          if LKimden1.Count=0 then
          begin
             LOnculArdil1.Clear;
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
            for g:=1 to GS do
            begin
              if CZ1[iter][p][g]=0 then
              begin
                isok := 0;
                t:=0;
                while (t<LOnculArdil1.Count) and (isOk=0) do
                begin
                  if V1[p][g]>V1[p][strtoint(LOnculArdil1[t])]
then
                  begin
                    isok := 1;
                    LOnculArdil1.Insert(t,inttostr(g));
                  end;
                  t:=t+1;
                end;
                if isok=0 then
                  LOnculArdil1.Add(inttostr(g));
              end;
            end;
            for t:=0 to LOnculArdil1.Count-1 do
            begin
                  g1:= LOnculArdil1[t];
                  tm1:=0;
                  ts21 := 0;
                  for t1:=1 to GS do
                  begin
                     if CZ1[iter][p][t1]=k then
                     begin
                        tm1:=tm1+GL[t1];
                        ts21 := ts21 + S2[t1];
                     end;
                  end;
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
                  tiz1 :=
(tm1+GL[strtoint(g1)])+(Z*sqrt(ts21+S2[strtoint(g1)]));
                  if tiz1>C then
                  begin
                     k:=k+1;
                  end
                  else
                  begin
                     sonk := k;
                     kc2:=kc2+sqr((tm1+(Z*sqrt(ts21)))-C);
                  end;
                  CZ1[iter][p][strtoint(g1)] := k;
            end;
         end;
      end;
      if sonk < k then
      begin
         tm1:=0;
         ts21 := 0;
         for t1:=1 to GS do
         begin
            if CZ1[iter][p][t1]=k then
            begin
                tm1:=tm1+GL[t1];
                ts21 := ts21 + S2[t1];
            end;
         end;
          kc2:=kc2+sqr((tm1+(Z*sqrt(ts21)))-C);
      end;
      CZ1[iter][p][GS+1] := k;
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
      F1[iter][p] := (CZ1[iter][p][GS+1]-LB)+sqrt(kc2)/(C*
sqrt(CZ1[iter][p][GS+1]));

      if minf> F1[iter][p] then
      begin
         PB1[iter] := p;
         minf := F1[iter][p];
      end;
      p:=p+1;
   end;
   if CZ1[iter][PB1[iter]][GS+1]<CZ1[GB1][PB1[GB1]][GS+1] then
     GB1 := iter
   else if

(CZ1[iter][PB1[iter]][GS+1]=CZ1[GB1][PB1[GB1]][GS+1]) and
(F1[iter][PB1[iter]]<F1[GB1][PB1[GB1]]) then
     GB1 := iter;
   for p:=1 to PS do
      for g:=1 to GS do
         begin
            V1[p][g]  :=
(w*V1[p][g]+random*(CZ1[iter][PB1[iter]][g]-
CZ1[iter][p][g])+c2*random* (CZ1[GB1][PB1[GB1]][g]-
CZ1[iter][p][g]))/Vmax;
         end;
   iter := iter+1;
  end;

  time2 := now;
  time1 := time2-time1;
  txtCPUTime2.Text := FormatDateTime('nn.ss.zzz',time1) ;

  if CZ1[GB1][PB1[GB1]][GS+1]<=LB then
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
  begin
    txtSonCozum.Lines.Add('LB
bulundu:'+inttostr(CZ1[GB1][PB1[GB1]][GS+1]));
  end;

  txtSonistsayi.Text := inttostr(CZ1[GB1][PB1[GB1]][GS+1]);
  txtSonCozum.Lines.Add('GBest          ');
  txtSonCozum.Lines.Add('İterasyon      : '+IntToStr(GB1));
  txtSonCozum.Lines.Add('Parça          :
'+IntToStr(PB1[GB1]));
  txtSonCozum.Lines.Add('F              :
'+FloatToStr(round(F1[GB1][PB1[GB1]]*10000)/10000));
  txtSonCozum.Lines.Add('İstasyon Sayısı:
'+IntToStr(CZ1[GB1][PB1[GB1]][GS+1]));
  for k:=1 to CZ1[GB1][PB1[GB1]][GS+1] do
  begin
     vstr := '  K-'+InttoStr(k)+' : (';
     for g:=1 to GS do
       if CZ1[GB1][PB1[GB1]][g] = k then
     txtSonCozum.Lines.Add(vstr+')')
  end;

  btnDosyaYukle.Enabled := true;
  btnPSO.Enabled := true;
end;

procedure TfrmTanim.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  LKimden.Free;
  LKime.Free;
  LKimden1.Free;
  LKime1.Free;
```

EK-1 (Devam) SUMHDP-1 için PSO algoritması delphi kodu

```
  LOnculArdil1.Free;
end;

end.
```

EK-2 Düşük varyans için test problemleri sonuçları

			düşük varyans								
			Z 0.975			Z 0.95			Z 0.90		
	GS	C	LB	çözüm	CPU süre	LB	çözüm	CPU süre	LB	çözüm	CPU süre
Arcus 1	83	4454	18	21	500	18	21	500	18	20	500
Arcus 1	83	4732	17	20	500	17	20	500	17	18	500
Arcus 1	83	5048	16	19	500	16	18	500	16	17	500
Arcus 1	83	5408	15	17	500	15	17	500	15	16	500
Arcus 1	83	5824	14	15	500	14	15	500	14	15	500
Arcus 1	83	5853	14	15	500	14	15	500	14	15	500
Arcus 1	83	6309	13	14	500	13	14	500	13	14	500
Arcus 1	83	6842	12	13	500	12	13	500	12	13	500
Arcus 1	83	6883	12	13	500	12	13	500	12	13	500
Arcus 1	83	7571	11	12	500	11	12	500	11	12	500
Arcus 1	83	8412	10	11	500	10	11	500	10	11	500
Arcus 1	83	8898	9	10	500	9	10	500	9	10	500
Arcus 1	83	10816	**8**	**8**	**0,082**	**8**	**8**	**0,134**	**8**	**8**	**0,042**
Arcus 2	111	6837	Mümkün çözüm yok						23	27	500
Arcus 2	111	7162	22	28	500	22	28	500	22	26	500
Arcus 2	111	7520	21	26	500	21	26	500	21	25	500
Arcus 2	111	7916	20	24	500	20	25	500	20	23	500
Arcus 2	111	8356	19	23	500	19	23	500	19	22	500
Arcus 2	111	8847	18	22	500	18	22	500	18	20	500
Arcus 2	111	9400	17	20	500	17	20	500	17	19	500
Arcus 2	111	10027	16	19	500	16	19	500	16	18	500
Arcus 2	111	10743	15	17	500	15	17	500	15	16	500
Arcus 2	111	11378	14	16	500	14	16	500	14	15	500
Arcus 2	111	11570	14	16	500	14	16	500	14	15	500
Arcus 2	111	17067	10	11	500	10	11	500	**10**	**10**	**1,422**
Barthold2	148	112	40	53	500	39	52	500	39	49	500
Barthold2	148	115	38	50	500	38	50	500	38	48	500
Barthold2	148	118	38	49	500	37	49	500	37	47	500
Barthold2	148	121	37	48	500	36	46	500	36	45	500
Barthold2	148	125	35	46	500	35	45	500	35	43	500
Barthold2	148	129	34	45	500	34	42	500	34	41	500
Barthold2	148	133	33	43	500	33	42	500	33	40	500
Barthold2	148	137	33	42	500	32	39	500	32	39	500
Barthold2	148	142	31	40	500	31	38	500	31	37	500
Barthold2	148	146	31	39	500	30	35	500	30	35	500
Barthold2	148	152	29	36	500	29	34	500	29	33	500
Barthold2	148	157	28	34	500	28	33	500	28	32	500
Barthold2	148	163	27	33	500	27	31	500	27	30	500
Barthold2	148	170	26	31	500	26	29	500	26	29	500
Barthold	148	513	12	13	500	12	13	500	12	13	500
Barthold	148	564	11	12	500	11	12	500	11	12	500
Barthold	148	626	10	11	500	**10**	**10**	**199,141**	**10**	**10**	**17,532**
Barthold	148	705	9	10	500	**9**	**9**	**1,468**	**9**	**9**	**1,422**
Barthold	148	805	**8**	**8**	**0,406**	**8**	**8**	**0,328**	**8**	**8**	**0,125**
Bowman	8	20	5	6	0,016	5	6	0,016	4	5	0,047
Buxey	29	30	Mümkün çözüm yok						12	14	500
Buxey	29	33	11	13	500	11	12	500	11	12	500
Buxey	29	36	10	12	500	10	11	500	10	11	500

EK-2 (Devam) Düşük varyans için test problemleri sonuçları

			düşük varyans								
			Z 0.975			Z 0.95			Z 0.90		
	GS	C	LB	çözüm	CPU süre	LB	çözüm	CPU süre	LB	çözüm	CPU süre
Buxey	29	41	9	11	500	9	10	500	9	10	500
Buxey	29	47	8	9	500	8	9	500	8	9	500
Buxey	29	54	7	8	500	7	8	500	**7**	**7**	**1,062**
Gunther	35	49	11	12	500	11	12	500	11	12	500
Gunther	35	54	10	11	500	10	11	500	10	11	500
Gunther	35	61	9	10	500	9	10	500	9	10	500
Gunther	35	69	8	9	500	8	9	500	8	9	500
Gunther	35	81	7	8	500	7	8	500	**7**	**7**	**0,500**
Hahn	53	2338	7	8	2,560	7	8	2,469	**7**	**7**	**1,406**
Hahn	53	2806	**6**	**6**	**1,578**	**6**	**6**	**1,766**	**6**	**6**	**0,625**
Hahn	53	3507	**5**	**5**	**0,160**	**5**	**5**	**0,031**	**5**	**5**	**0,047**
Hahn	53	4676	**4**	**4**	**0,070**	**4**	**4**	**0,016**	**4**	**4**	**0,032**
Heskiaoff	28	205	**6**	**6**	**2,491**	**6**	**6**	**0,595**	**6**	**6**	**0,250**
Heskiaoff	28	216	**6**	**6**	**0,610**	**6**	**6**	**0,016**	5	6	0,203
Heskiaoff	28	256	**5**	**5**	**0,111**	**5**	**5**	**0,016**	**5**	**5**	**0,015**
Heskiaoff	28	324	**4**	**4**	**0,016**	**4**	**4**	**0,015**	**4**	**4**	**0,010**
Heskiaoff	28	342	**4**	**4**	**0,016**	**4**	**4**	**0,015**	**4**	**4**	**0,010**
Jackson	11	9	6	7	0,125	6	7	0,170	6	7	0,094
Jackson	11	10	6	7	0,030	6	7	0,170	5	6	0,170
Jackson	11	13	4	5	0,029	4	5	0,160	4	5	0,093
Jackson	11	14	**4**	**4**	**0,031**	**4**	**4**	**0,371**	**4**	**4**	**0,010**
Jackson	11	21	**3**	**3**	**0,016**	**3**	**3**	**0,010**	**3**	**3**	**0,010**
Jaeschke	9	6	7	8	0,110	7	8	0,187	7	8	0,062
Jaeschke	9	7	6	7	0,265	6	7	0,187	6	7	0,062
Jaeschke	9	8	5	7	0,250	5	7	0,120	5	7	0,063
Jaeschke	9	10	4	5	0,110	4	5	0,203	4	5	0,063
Jaeschke	9	18	**3**	**3**	**0,001**	**3**	**3**	**0,010**	**3**	**3**	**0,001**
Kilbridge	45	79	8	9	0,288	8	9	0,330	**8**	**8**	**0,578**
Kilbridge	45	92	7	8	0,265	**7**	**7**	**2,032**	**7**	**7**	**0,313**
Kilbridge	45	110	**6**	**6**	**0,211**	**6**	**6**	**0,140**	**6**	**6**	**0,016**
Kilbridge	45	138	**5**	**5**	**0,023**	**5**	**5**	**0,016**	**5**	**5**	**0,016**
Kilbridge	45	184	**4**	**4**	**0,016**	**4**	**4**	**0,016**	**4**	**4**	**0,015**
Lutz1	32	1572	Mümkün çözüm yok						10	11	500
Lutz1	32	1768	9	10	500	9	10	500	9	10	500
Lutz1	32	2020	8	9	500	8	9	500	8	9	500
Lutz1	32	2357	7	8	500	7	8	500	**7**	**7**	**0,109**
Lutz1	32	2828	**6**	**6**	**4,937**	**6**	**6**	**0,141**	**6**	**6**	**0,016**
Lutz2	89	14	36	50	500	36	47	500	36	44	500
Lutz2	89	15	34	47	500	34	44	500	34	41	500
Lutz2	89	16	32	45	500	32	42	500	32	38	500
Lutz2	89	17	30	41	500	30	39	500	30	36	500
Lutz2	89	18	29	38	500	28	36	500	28	34	500
Lutz2	89	19	27	35	500	27	34	500	27	32	500
Lutz2	89	20	26	32	500	26	33	500	25	30	500
Lutz2	89	21	25	31	500	24	30	500	24	29	500
Lutz3	89	92	19	23	500	19	22	500	19	21	500
Lutz3	89	97	18	22	500	18	21	500	18	20	500
Lutz3	89	103	17	20	500	17	19	500	17	19	500

EK-2 (Devam) Düşük varyans için test problemleri sonuçları

			düşük varyans								
			Z 0.975			Z 0.95			Z 0.90		
	GS	C	LB	çözüm	CPU süre	LB	çözüm	CPU süre	LB	çözüm	CPU süre
Lutz3	89	110	16	19	500	16	18	500	16	18	500
Lutz3	89	118	15	17	500	15	17	500	15	16	500
Lutz3	89	127	14	15	500	14	15	500	14	15	500
Lutz3	89	137	13	14	500	13	14	500	13	14	500
Lutz3	89	150	12	13	500	12	13	500	12	13	500
Mansoor	11	62	**4**	**4**	**0,015**	**4**	**4**	**0,016**	**4**	**4**	**0,010**
Mansoor	11	94	**3**	**3**	**0,001**	**3**	**3**	**0,001**	**3**	**3**	**0,001**
Mertens	7	8	**5**	**5**	**0,020**	4	5	0,060	4	5	0,063
Mertens	7	10	**4**	**4**	**0,040**	**4**	**4**	**0,060**	**4**	**4**	**0,016**
Mertens	7	15	**3**	**3**	**0,001**	**3**	**3**	**0,010**	**3**	**3**	**0,010**
Mertens	7	18	**2**	**2**	**0,001**	**2**	**2**	**0,010**	**2**	**2**	**0,010**
Mitchell	21	15	Mümkün çözüm yok						8	9	0,188
Mitchell	21	21	**6**	**6**	**0,562**	**6**	**6**	**0,340**	**6**	**6**	**0,140**
Mitchell	21	26	**5**	**5**	**0,148**	**5**	**5**	**0,016**	**5**	**5**	**0,010**
Mitchell	21	35	**4**	**4**	**0,007**	**4**	**4**	**0,010**	**4**	**4**	**0,010**
Mitchell	21	39	3	4	0,105	3	4	0,188	**3**	**3**	**0,047**
Mukherje	94	211	Mümkün çözüm yok			21	25	500	21	24	500
Mukherje	94	222	Mümkün çözüm yok			20	23	500	20	23	500
Mukherje	94	234	19	23	500	19	22	500	19	21	500
Mukherje	94	248	18	21	500	18	21	500	18	20	500
Mukherje	94	263	17	20	500	17	20	500	17	19	500
Mukherje	94	281	16	18	500	16	18	500	16	18	500
Mukherje	94	301	15	17	500	15	17	500	15	17	500
Mukherje	94	324	14	16	500	14	16	500	14	15	500
Mukherje	94	351	13	15	500	13	14	500	13	14	500
Roszieg	25	16	9	10	500	9	10	500	9	10	500
Roszieg	25	18	8	9	500	8	9	500	8	9	500
Roszieg	25	21	7	8	500	7	8	500	**7**	**7**	**0,406**
Roszieg	25	25	6	7	500	**6**	**6**	**2,046**	**6**	**6**	**0,016**
Roszieg	25	32	**5**	**5**	**0,012**	**5**	**5**	**0,016**	**5**	**5**	**0,001**
Sawyer	30	30	12	15	500	12	14	500	12	13	500
Sawyer	30	33	11	13	500	11	13	500	11	12	500
Sawyer	30	36	10	11	500	10	11	500	10	11	500
Sawyer	30	41	9	10	500	9	10	500	9	10	500
Sawyer	30	47	8	9	500	8	9	500	8	9	500
Sawyer	30	54	7	8	500	7	8	500	**7**	**7**	**2,500**
Sawyer	30	75	5	6	500	**5**	**5**	**0,015**	**5**	**5**	**0,032**
Scholl	297	1742	41	50	500	41	49	500	41	48	500
Scholl	297	1787	40	49	500	40	48	500	40	46	500
Scholl	297	1834	39	47	500	39	47	500	39	45	500
Scholl	297	1883	38	46	500	38	46	500	38	43	500
Scholl	297	1935	37	45	500	37	44	500	37	42	500
Scholl	297	1991	36	43	500	36	42	500	36	41	500
Scholl	297	2049	35	42	500	35	41	500	35	40	500
Scholl	297	2111	34	40	500	34	39	500	34	38	500
Scholl	297	2177	33	39	500	33	37	500	33	37	500
Scholl	297	2247	32	38	500	32	35	500	32	35	500
Scholl	297	2322	31	37	500	31	34	500	31	34	500

EK-2 (Devam) Düşük varyans için test problemleri sonuçları

			düşük varyans								
			Z 0.975			Z 0.95			Z 0.90		
	GS	C	LB	çözüm	CPU süre	LB	çözüm	CPU süre	LB	çözüm	CPU süre
Scholl	297	2402	30	35	500	30	33	500	30	33	500
Scholl	297	2488	29	33	500	29	32	500	29	31	500
Scholl	297	2580	28	32	500	28	31	500	28	30	500
Scholl	297	2680	27	30	500	27	30	500	27	29	500
Scholl	297	2787	26	29	500	26	28	500	26	27	500
Tonge	70	207	Mümkün çözüm yok			18	22	500	18	21	500
Tonge	70	220	17	21	500	17	20	500	17	19	500
Tonge	70	234	16	19	500	16	19	500	16	18	500
Tonge	70	251	15	18	500	15	18	500	15	17	500
Tonge	70	270	14	16	500	14	16	500	14	15	500
Tonge	70	293	13	15	500	13	15	500	13	14	500
Tonge	70	320	12	13	500	12	13	500	12	13	500
Tonge	70	364	11	12	500	11	12	500	11	12	500
Tonge	70	410	10	11	500	9	10	500	9	10	500
Tonge	70	468	8	9	500	8	9	500	8	9	500
Tonge	70	527	**8**	**8**	**0,359**	7	8	500	7	8	500
Warnecke	58	68	Mümkün çözüm yok			24	30	500	24	29	500
Warnecke	58	71	23	29	500	23	28	500	23	27	500
Warnecke	58	74	22	28	500	22	27	500	22	25	500
Warnecke	58	78	21	27	500	21	25	500	21	24	500
Warnecke	58	82	20	26	500	20	24	500	20	23	500
Warnecke	58	86	19	24	500	19	22	500	19	21	500
Warnecke	58	92	18	22	500	18	21	500	18	20	500
Warnecke	58	97	17	20	500	17	19	500	17	19	500
Warnecke	58	104	16	18	500	16	18	500	16	18	500
Warnecke	58	111	15	16	500	15	16	500	15	16	500
Wee-mag	75	34	Mümkün çözüm yok						46	63	500
Wee-mag	75	35	Mümkün çözüm yok			45	62	500	45	62	500
Wee-mag	75	36	Mümkün çözüm yok			44	62	500	43	62	500
Wee-mag	75	37	Mümkün çözüm yok			42	62	500	42	61	500
Wee-mag	75	38	42	61	500	41	61	500	41	60	500
Wee-mag	75	39	41	61	500	40	60	500	40	60	500
Wee-mag	75	40	40	60	500	39	59	500	39	59	500
Wee-mag	75	41	39	60	500	38	59	500	38	59	500
Wee-mag	75	42	38	60	500	38	60	500	37	59	500
Wee-mag	75	43	37	60	500	37	59	500	36	58	500
Wee-mag	75	45	35	58	500	35	58	500	35	59	500
Wee-mag	75	46	34	57	500	34	57	500	34	57	500
Wee-mag	75	47	34	57	500	34	57	500	33	57	500
Wee-mag	75	49	32	57	500	32	55	500	32	52	500
Wee-mag	75	50	32	54	500	32	52	500	31	51	500
Wee-mag	75	52	30	51	500	30	49	500	30	44	500
Wee-mag	75	54	29	46	500	29	45	500	29	39	500
Wee-mag	75	56	28	42	500	28	39	500	28	35	500

EK-3 Yüksek varyans için test problemleri sonuçları

			yüksek varyans								
			Z 0.975			Z 0.95			Z 0.90		
	GS	C	LB	çözüm	CPU süre	LB	çözüm	CPU süre	LB	çözüm	CPU süre
Arcus 1	83	4454	Mümkün çözüm yok						18	21	500
Arcus 1	83	4732	Mümkün çözüm yok			18	22	500	17	20	500
Arcus 1	83	5048	17	21	500	17	21	500	16	19	500
Arcus 1	83	5408	16	20	500	16	19	500	15	17	500
Arcus 1	83	5824	15	18	500	14	17	500	14	16	500
Arcus 1	83	5853	15	18	500	14	17	500	14	16	500
Arcus 1	83	6309	14	17	500	13	16	500	13	15	500
Arcus 1	83	6842	13	15	500	12	14	500	12	14	500
Arcus 1	83	6883	12	14	500	12	14	500	12	13	500
Arcus 1	83	7571	11	13	500	11	12	500	11	12	500
Arcus 1	83	8412	10	11	500	10	11	500	10	11	500
Arcus 1	83	8898	10	11	500	10	11	500	9	10	500
Arcus 1	83	10816	8	9	500	8	9	500	8	9	500
Arcus 2	111	7520	Mümkün çözüm yok						22	28	500
Arcus 2	111	7916	Mümkün çözüm yok						21	27	500
Arcus 2	111	8356	Mümkün çözüm yok			20	27	500	20	24	500
Arcus 2	111	8847	Mümkün çözüm yok			19	25	500	19	23	500
Arcus 2	111	9400	18	25	500	18	24	500	17	20	500
Arcus 2	111	10027	17	23	500	17	22	500	16	19	500
Arcus 2	111	10743	16	21	500	16	19	500	15	18	500
Arcus 2	111	11378	15	19	500	15	18	500	15	17	500
Arcus 2	111	11570	15	19	500	15	18	500	14	15	500
Arcus 2	111	17067	10	13	500	10	12	500	10	11	500
Barthold2	148	125	Mümkün çözüm yok						35	46	500
Barthold2	148	129	Mümkün çözüm yok						35	46	500
Barthold2	148	133	Mümkün çözüm yok						34	44	500
Barthold2	148	137	Mümkün çözüm yok			33	44	500	33	43	500
Barthold2	148	142	Mümkün çözüm yok			32	43	500	32	41	500
Barthold2	148	146	Mümkün çözüm yok			31	40	500	31	39	500
Barthold2	148	152	30	41	500	30	38	500	30	38	500
Barthold2	148	157	29	39	500	29	37	500	29	37	500
Barthold2	148	163	28	38	500	28	36	500	28	35	500
Barthold2	148	170	27	36	500	27	34	500	27	33	500
Barthold	148	513	Mümkün çözüm yok						12	13	500
Barthold	148	564	11	12	500	11	12	500	11	12	500
Barthold	148	626	10	11	500	10	11	500	10	11	500
Barthold	148	705	9	10	500	9	10	500	9	10	500
Barthold	148	805	8	9	500	8	9	500	8	9	500
Bowman	8	20	5	6	0,046	5	6	0,062	5	6	0,063
Buxey	29	41	8	11	500	9	11	500	9	11	500
Buxey	29	47	8	10	500	8	10	500	8	10	500
Buxey	29	54	7	9	500	7	8	500	7	8	500
Gunther	35	61	Mümkün çözüm yok						9	11	500
Gunther	35	69	Mümkün çözüm yok			8	11	500	8	10	500
Gunther	35	81	7	9	500	7	9	500	7	8	500
Hahn	53	2338	Mümkün çözüm yok						7	8	2,359

EK-3 (Devam) Yüksek varyans için test problemleri sonuçları

			yüksek varyans								
			Z 0.975			Z 0.95			Z 0.90		
	GS	C	LB	çözüm	CPU süre	LB	çözüm	CPU süre	LB	çözüm	CPU süre
Hahn	53	2806	6	7	2,360	6	7	2,343	6	7	2,375
Hahn	53	3507	5	6	2,344	5	6	2,359	**5**	**5**	**0,125**
Hahn	53	4676	**4**	**4**	**1,718**	**4**	**4**	**0,640**	**4**	**4**	**0,203**
Heskiaoff	28	205	6	8	0,704	6	7	0,922	6	7	0,375
Heskiaoff	28	216	6	7	0,906	6	7	0,547	**6**	**6**	**1,362**
Heskiaoff	28	256	5	6	0,310	5	6	0,390	**5**	**5**	**0,631**
Heskiaoff	28	324	4	5	0,235	**4**	**4**	**0,210**	**4**	**4**	**0,391**
Heskiaoff	28	342	**4**	**4**	**0,125**	**4**	**4**	**0,016**	**4**	**4**	**0,031**
Jackson	11	10	Mümkün çözüm yok						6	7	0,172
Jackson	11	13	**5**	**5**	**2,141**	**5**	**5**	**0,969**	4	5	0,140
Jackson	11	14	4	5	0,500	4	5	0,109	4	5	0,094
Jackson	11	21	**3**	**3**	**0,031**	**3**	**3**	**0,047**	**3**	**3**	**0,010**
Jaeschke	9	8	6	7	0,750	6	7	0,344	6	7	0,078
Jaeschke	9	10	5	7	0,078	5	7	0,125	5	6	0,906
Jaeschke	9	18	**3**	**3**	**0,048**	**3**	**3**	**0,010**	**3**	**3**	**0,010**
Kilbridge	45	92	7	8	0,922	7	8	2,938	7	8	0,469
Kilbridge	45	110	6	7	0,438	6	7	0,610	**6**	**6**	**3,585**
Kilbridge	45	138	5	6	0,437	**5**	**5**	**0,734**	**5**	**5**	**0,140**
Kilbridge	45	184	**4**	**4**	**0,063**	**4**	**4**	**0,016**	**4**	**4**	**0,030**
Lutz1	32	2020	Mümkün çözüm yok			8	10	500	8	10	500
Lutz1	32	2357	Mümkün çözüm yok			7	9	500	7	8	500
Lutz1	32	2828	6	7	500	6	7	500	6	7	500
Lutz2	89	15	Mümkün çözüm yok						35	48	500
Lutz2	89	16	Mümkün çözüm yok						33	45	500
Lutz2	89	17	Mümkün çözüm yok			31	45	500	31	42	500
Lutz2	89	18	Mümkün çözüm yok			29	42	500	29	40	500
Lutz2	89	19	28	43	500	28	41	500	27	37	500
Lutz2	89	20	27	40	500	27	38	500	26	35	500
Lutz2	89	21	26	37	500	25	35	500	25	34	500
Lutz3	89	97	Mümkün çözüm yok						18	22	500
Lutz3	89	103	Mümkün çözüm yok						17	21	500
Lutz3	89	110	Mümkün çözüm yok			17	21	500	16	20	500
Lutz3	89	118	16	20	500	16	19	500	15	18	500
Lutz3	89	127	14	17	500	14	17	500	14	17	500
Lutz3	89	137	14	16	500	13	16	500	13	15	500
Lutz3	89	150	13	15	500	12	14	500	12	13	500
Mansoor	11	62	Mümkün çözüm yok			4	5	3,625	**4**	**4**	**0,063**
Mansoor	11	94	**3**	**3**	**0,016**	**3**	**3**	**0,012**	**3**	**3**	**0,001**
Mertens	7	8	Mümkün çözüm yok						**5**	**5**	**0,313**
Mertens	7	10	4	5	0,050	4	5	0,032	4	5	0,047
Mertens	7	15	**3**	**3**	**0,010**	**3**	**3**	0,010	**3**	**3**	**0,010**
Mertens	7	18	2	3	0,031	2	3	0,031	**2**	**2**	**0,016**
Mitchell	21	21	6	8	0,329	6	7	0,375	6	7	0,313
Mitchell	21	26	5	6	0,359	5	6	0,125	**5**	**5**	**0,671**
Mitchell	21	35	**4**	**4**	**0,062**	**4**	**4**	**0,157**	**4**	**4**	**0,015**
Mitchell	21	39	**4**	**4**	**0,010**	**4**	**4**	**0,015**	3	4	0,141

EK-3 (Devam) Yüksek varyans için test problemleri sonuçları

			yüksek varyans								
			Z 0.975			Z 0.95			Z 0.90		
	GS	C	LB	çözüm	CPU süre	LB	çözüm	CPU süre	LB	çözüm	CPU süre
Mukherje	94	222	Mümkün çözüm yok						20	25	500
Mukherje	94	234	Mümkün çözüm yok						20	24	500
Mukherje	94	248	Mümkün çözüm yok						18	22	500
Mukherje	94	263	Mümkün çözüm yok			18	22	500	17	21	500
Mukherje	94	281	17	22	500	17	21	500	16	19	500
Mukherje	94	301	16	20	500	16	20	500	15	18	500
Mukherje	94	324	15	19	500	15	18	500	14	17	500
Mukherje	94	351	14	18	500	13	16	500	13	15	500
Roszieg	25	16	Mümkün çözüm yok						9	11	500
Roszieg	25	18	Mümkün çözüm yok						8	10	500
Roszieg	25	21	7	9	500	7	9	500	7	8	500
Roszieg	25	25	6	8	500	6	7	500	6	7	500
Roszieg	25	32	5	6	500	5	6	500	**5**	**5**	**0,031**
Sawyer	30	33	Mümkün çözüm yok						11	14	500
Sawyer	30	36	11	14	500	10	13	500	10	12	500
Sawyer	30	41	9	12	500	9	12	500	9	11	500
Sawyer	30	47	8	11	500	8	10	500	8	10	500
Sawyer	30	54	7	9	500	7	9	500	7	8	500
Sawyer	30	75	5	6	500	5	6	500	5	6	500
Scholl	297	2049	Mümkün çözüm yok						36	43	500
Scholl	297	2111	Mümkün çözüm yok			35	44	500	35	41	500
Scholl	297	2177	34	43	500	34	43	500	34	40	500
Scholl	297	2247	33	42	500	33	42	500	33	39	500
Scholl	297	2322	32	41	500	32	39	500	32	37	500
Scholl	297	2402	31	39	500	31	38	500	30	35	500
Scholl	297	2488	30	38	500	30	36	500	29	34	500
Scholl	297	2580	29	36	500	29	35	500	28	33	500
Scholl	297	2680	28	35	500	28	33	500	27	32	500
Scholl	297	2787	27	33	500	27	32	500	26	30	500
Tonge	70	220	Mümkün çözüm yok						18	22	500
Tonge	70	234	Mümkün çözüm yok						16	20	500
Tonge	70	251	Mümkün çözüm yok						15	18	500
Tonge	70	270	15	20	500	15	19	500	14	17	500
Tonge	70	293	14	18	500	14	18	500	13	16	500
Tonge	70	320	13	16	500	12	15	500	12	14	500
Tonge	70	364	11	14	500	11	14	500	11	13	500
Tonge	70	410	10	12	500	10	12	500	10	11	500
Tonge	70	468	9	10	500	9	10	500	8	9	500
Tonge	70	527	8	9	500	8	9	500	8	9	500
Warnecke	58	82	Mümkün çözüm yok						21	25	500
Warnecke	58	86	Mümkün çözüm yok						20	24	500
Warnecke	58	92	19	24	500	19	24	500	18	22	500
Warnecke	58	97	18	22	500	18	22	500	18	21	500
Warnecke	58	104	17	20	500	17	20	500	16	18	500
Warnecke	58	111	16	19	500	16	18	500	15	17	500
Wee-mag	75	41	Mümkün çözüm yok						39	61	500

EK-3 (Devam) Yüksek varyans için test problemleri sonuçları

			yüksek varyans								
			Z 0.975			Z 0.95			Z 0.90		
	GS	C	LB	çözüm	CPU süre	LB	çözüm	CPU süre	LB	çözüm	CPU süre
Wee-mag	75	42	Mümkün çözüm yok						38	60	500
Wee-mag	75	43	Mümkün çözüm yok						37	60	500
Wee-mag	75	45	Mümkün çözüm yok						36	59	500
Wee-mag	75	46	Mümkün çözüm yok			35	60	500	35	58	500
Wee-mag	75	47	Mümkün çözüm yok			35	59	500	34	59	500
Wee-mag	75	49	Mümkün çözüm yok			33	58	500	33	57	500
Wee-mag	75	50	33	59	500	33	57	500	32	57	500
Wee-mag	75	52	32	58	500	31	57	500	31	53	500
Wee-mag	75	54	31	56	500	30	56	500	30	52	500
Wee-mag	75	56	29	53	500	29	52	500	29	50	500

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: BALİ, Özkan
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 26.06.1976 Bursa
Medeni hali	: Evli
Telefon	: 0 (312) 417 51 90
Faks	: -
e-posta	: ozkanbali@yahoo.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Yüksek lisans	Gazi Üniversitesi / Endüstri Müh. Bölümü	2004
Yüksek lisans	Başkent Üniversitesi / İşletme Bölümü	2001
Lisans	Yıldız Teknik Ü. / Endüstri Müh. Bölümü	1998
Lise	Bursa Çınar Lisesi	1993

İş Deneyimi

Yıl	Yer	Görev
1998-2009	Kara Harp Okulu	Öğretim Elemanı

Yabancı Dil

İngilizce

Yayınlar

1. Bali, Ö., Gencer, C.**,** "AHP, bulanık AHP ve bulanık mantıkla KHO'ya öğretim elemanı seçimi", Savunma Bilimleri Dersigi, 4(1):24-43, 2005.

Hobiler

Fotoğrafçılık, müzik dinlemek, sinema.