

T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ
ENSTİTÜSÜ

ALT KÜME BULMA TABANLI AYRIK
OPTİMİZASYON PROBLEMLERİ İÇİN
AYRIK PARÇACIK SÜRÜ
OPTİMİZASYONU MODELLERİ

Salih CEBECİ
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

GEBZE
2012

T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ
ENSTİTÜSÜ

ALT KÜME BULMA TABANLI AYRIK
OPTİMİZASYON PROBLEMLERİ İÇİN
AYRIK PARÇACIK SÜRÜ
OPTİMİZASYONU MODELLERİ

Salih CEBECİ
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

TEZ DANIŞMANI
Doç. Dr. Fatih Erdoğan SEVİLGİN

GEBZE
2012

JURI ONAY FORMU

ÖZET

TEZİN BAŞLIĞI: ALT KÜME BULMA TABANLI AYRIK OPTİMİZASYON PROBLEMLERİ İÇİN AYRIK PARÇACIK SÜRÜ OPTİMİZASYONU MODELLERİ

YAZAR ADI : SALİH CEBECİ

Parçacık sürü optimizasyonu (PSO) sürekli optimizasyon problemleri için geliştirilmiş bir metasezgiseldir. PSO, daha sonra bazı ayrık optimizasyon problemleri için de uygulanmıştır. Yaptığımız araştırmalarda PSO'nun alt küme tabanlı ayrık optimizasyon problemlerine uygulandığına rastlanmamıştır. Bu çalışmada PSO'nun bu tip problemlere etkin bir şekilde uygulanabilirliği araştırılmıştır.

Çalışma kapsamında alt küme tabanlı ayrık optimizasyon problemlerinin çoğunu kapsayan “büyüme ve küçülme amaçlayan” alt küme problemleri tanımlanmış ve bu problemlere uygulanabilecek bir PSO modeli önerilmiştir. Geliştirilen model uygulanarak üç problem için çözüm üreten algoritmalar sunulmuştur. Bu problemler maksimum klik problemi(MKP), çok boyutlu sırt çantası problemi(ÇSP) ve kenar kapsama problemidir (KKP).

Karşılaştırma problemleri ile yapılan testler sonucunda modelin MKP ve KKP üzerinde başarı ile çalıştığı gözlemlenmiştir. Bu problemler için geliştirilen algoritmalar çözüm kalitesi açısından diğer metasezgisellerden daha iyi sonuçlar vermiştir. ÇSP için ise iyi sonuçlar alınamamıştır. Modelin ilk iki problem için başarı ile çalışırken son problem için başarılı olmamasının nedenleri sorgulanmıştır.

SUMMARY

**THESIS TITLE: DISCRETE PARTICLE SWARM OPTIMIZATION
MODELS FOR DISCRETE OPTIMIZATION PROBLEMS BASED ON
SUBSET SELECTION**

AUTHOR : SALİH CEBECİ

Particle Swarm Optimization (PSO) is a metaheuristic intended for continuous optimization problems. Later, PSO is applied to some discrete optimization problems as well. Application of PSO to any subset selection based problem is not observed during our research. In this work, a PSO model applicable to such problems is introduced.

In this work, “aiming increase and decrease” subset problems which cover most of the subset selection problems are defined and a PSO model is proposed for these problems. This model is applied to three such problems. These problems are maximum clique, multi-dimensional knapsack, and vertex cover problems.

It has been observed that the model is applicable successfully on maximum clique and vertex cover problems. The algorithms developed for these problems have given better results than other metaheuristics in terms of solution quality. For multi-dimensional knapsack problem, good results could not be attained. The reason why the model is applicable for the first two problems and why it is not for the last one was questionized.

TEŞEKKÜR

Benden desteğini esirgemeyen aileme, tez hocam Fatih Erdoğan Sevilgen'e, değerli hocalarım Yusuf Sinan Akgül ve Tunahan Çakır'a, iş arkadaşlarım Hakan Gümüş ve Umut Demirtaş'a teşekkür ederim.

Salih CEBECİ

İÇİNDEKİLER DİZİNİ

	<u>Sayfa</u>
ÖZET	v
SUMMARY	vi
TEŞEKKÜR	vii
İÇİNDEKİLER DİZİNİ	viii
SİMGELER VE KISALTMALAR DİZİNİ	x
ŞEKİLLER DİZİNİ	xi
TABLOLAR DİZİNİ	xii
1. GİRİŞ	1
2. OPTİMİZASYON PROBLEMLERİ	4
2.1. Tanımlar	4
2.2. Alt Küme Bulma Tabanlı Optimizasyon Problemleri	6
2.2.1. Maksimum Klık Problemi	9
2.2.2. Çok Boyutlu Sırt Çantası Problemi	10
2.2.3. Kenar Kapsama Problemi	11
3. METASEZGİSELLER	13
3.1. Genel Özellikleri	14
3.1.1. Model Oluşturma	14
3.1.2. Amaç Fonksiyonu	14
3.1.3. Olursuz Çözümlerle Başa Çıkma	14
3.1.4. Parametreleri Ayarlama	14
3.1.5. Performans Analizi	15
3.1.6. Tek Çözüm Tabanlı Metasezgiseller	15
3.1.7. Sürü Tabanlı Metasezgiseller	16
3.1.8. Melez Metasezgiseller	17
3.2. Sürü Tabanlı Metasezgisellere Örnekler	17

3.2.1.	Karınca Kolonisi Optimizasyonu	18
3.2.2.	Genetik Algoritma	19
3.2.3.	Parçacık Sürü Optimizasyonu	21
4.	GELİŞTİRİLEN AYRIK PSO MODELİ	26
4.1.	Tanımlar	27
4.2.	Modelin Detayları	28
4.2.1.	İklendirme	28
4.2.2.	Her Turda Yapılan İşlemler	28
4.2.3.	Durdurma Kriteri	31
4.2.4.	Modelin Parametreleri	31
4.3.	Zaman Karmaşıklık Analizi	32
5.	MODELİN PROBLEMLERE UYGULANMASI	34
5.1.	Modelin MKP'ye Uygulanması	34
5.2.	Modelin ÇSP'ye Uygulanması	35
5.3.	Modelin KKP'ye Uygulanması	35
6.	DENEYLER	37
6.1.	MKP ile Yapılan Deneyler	38
6.2.	ÇSP ile Yapılan Deneyler	58
6.3.	KKP ile Yapılan Deneyler	61
7.	SONUÇLAR	78
	KAYNAKLAR DİZİNİ	80
	ÖZGEÇMİŞ	84

SİMGELER VE KISALTMALAR DİZİNİ

PSO	: Parçacık Sürü Optimizasyonu
KKO	: Karınca Kolonisi Optimizasyonu
GA	: Genetik Algoritma
MKP	: Maksimum Klik Problemi
ÇSP	: Çok Boyutlu Sırt Çantası Problemi
KKP	: Kenar Kapsama Problemi
SB	: Sürü Büyüklüğü
MTS	: Maksimum Tur Sayısı

ŞEKİLLER DİZİNİ

<u>Şekil</u>	<u>Sayfa</u>
2.1. Optimizasyon Problemlerinin Sınıflandırması	6
3.1. Karınca Kolonisi Optimizasyonu Şablonu	18
3.2. Genetik Algoritma Şablonu	20
3.3. PSO’da Kullanılan Formüller	23
3.4. Parçacık Sürü Optimizasyonu Şablonu	23
4.1. PSO-A’da Pozisyonun Güncellenmesini Gösteren Formül	27
4.2. PSO-A Şablonu	30
6.1. SB - Çözüm Kalitesi Tablosu (MKP)	41
6.2. Farklı SB Değerleri İle Yapılan Test Sonuçları (MKP)	43
6.3. pKS - Çözüm Kalitesi Grafiği (MKP)	44
6.4. Farklı pKS Değerleri İle Yapılan Test Sonuçları (MKP)	45
6.5. pK - Çözüm Kalitesi Grafiği (MKP)	46
6.6. Lambda - Çözüm Kalitesi Grafiği (MKP)	47
6.7. Farklı Lambda Değerleri İle Yapılan Test Sonuçları (MKP)	49
6.8. Alpha – Çözüm Kalitesi Grafiği (MKP)	50
6.9. Farklı Alpha Değerleri İle Yapılan Test Sonuçları (MKP)	52
6.10. Seçim Oranları Testi (MKP)	53
6.11. Hepsi 1 ve pKS=0 Karşılaştırması (10000 tur) (MKP)	54
6.12. Parametrelerin İlk ve Son Değerlerinin Karşılaştırması (MKP)	56
6.13. pKS - Çözüm Kalitesi Grafiği (ÇSP)	59
6.14. Seçim Oranları Testi (ÇSP)	60
6.15. SB - Çözüm Kalitesi Grafiği (KKP)	63
6.16. Farklı SB Değerleri İle Yapılan Test Sonuçları (KKP)	65
6.17. pKS - Çözüm Kalitesi Grafiği (KKP)	66
6.18. Farklı pKS Değerleri İle Yapılan Test Sonuçları (KKP)	67
6.19. pK - Çözüm Kalitesi Grafiği (KKP)	68
6.20. Lambda - Çözüm Kalitesi Grafiği (KKP)	69
6.21. Alpha - Çözüm Kalitesi Grafiği (KKP)	71
6.22. Farklı Alpha Değerleri İle Yapılan Test Sonuçları (KKP)	73
6.23. Seçim Oranları Testi (KKP)	74
6.24. Parametrelerin İlk ve Son Değerlerinin Karşılaştırması (KKP)	76

TABLOLAR DİZİNİ

<u>Tablo</u>	<u>Sayfa</u>
2.1. Alt Küme Tabanlı Problemlere Örnekler	8
2.2. MKP'ye Uygulanan Bazı Metasezgiseller	10
2.3. ÇSP'ye Uygulanan Bazı Metasezgiseller	11
2.4. KKP'ye Uygulanan Bazı Metasezgiseller	12
2.5. MKP, ÇSP ve KKP – Özet	12
4.1. Noktaların Seçilme Oranları	29
4.2. Modelin Parametreleri	31
6.1. Parametrelerin İlk Değerleri	37
6.2. SB - Çözüm Kalitesi Tablosu (MKP)	42
6.3. Lambda - Çözüm Kalitesi Tablosu (MKP)	48
6.4. Alpha - Çözüm Kalitesi Tablosu (MKP)	51
6.5. Seçim Oranlarının Karşılaştırılması (MKP)	55
6.6. Parametrelerin İlk ve Son Değerleri	55
6.7. PSO-A'nın MKP'ye Uygulanışının Karşılaştırmalı Sonuçları	57
6.8. Test Sonuçlarının Karşılaştırması (MKP)	58
6.9. Seçim Oranlarının Karşılaştırılması (ÇSP)	60
6.10. PSO-A'nın ÇSP'ye Uygulanışının Karşılaştırmalı Sonuçları	61
6.11. SB - Çözüm Kalitesi Tablosu (KKP)	64
6.12. Lambda - Çözüm Kalitesi Tablosu (KKP)	70
6.13. Alpha - Çözüm Kalitesi Tablosu (KKP)	72
6.14. Seçim Oranlarının Karşılaştırılması (KKP)	75
6.15. Parametrelerin İlk ve Son Değerleri (KKP)	75
6.16. PSO-A'nın KKP'ye Uygulanışının Karşılaştırmalı Sonuçları	77

1. GİRİŞ

Karar değişkenleri kullanılarak tanımlanan bir problemde değişkenlerin uygun değerler alarak ulaşabileceği en iyi çözümü arayan problem tiplerine optimizasyon problemleri denir. Optimizasyon problemleri, çözüm uzayının cinsine göre sürekli ve ayrık olarak iki gruba ayrılabilir. Sürekli optimizasyon problemleri gerçek sayılı fonksiyonların, ayrık optimizasyon problemleri ise değişkenleri ayrık değerler olan optimizasyon problemleridir. Alt küme tabanlı ayrık optimizasyon problemleri ise, çözümleri belli bir kümenin alt kümesi olan ayrık optimizasyon problemleridir. Bu tip problemlerde çözüm olarak seçilen elemanların sırasının bir önemi olmayıp sadece hangi elemanlarının seçildiğinin önemi vardır.

Alt küme tabanlı ayrık problemlerin de içinde bulunduğu pek çok optimizasyon problemlerinde optimal sonucu bulmak çok zaman alacağı için bu problemlerin çözümünde genel olarak optimele yakın sonuçlar bulan ve hızlı çalışan yöntemler kullanılır. Bu tip yöntemlere, kesin çözüm vermesi garanti olmadığı ve sezgisel fikirler ile geliştirildikleri için sezgisel yöntemler adı verilir. Bazı sezgisel yöntemler belli bir tip probleme özgü olmayıp problemden bağımsız bir yöntem olarak karşımıza çıkarlar. Genellikle doğadan esinlenen bu yöntemlere metasezgiseller adı verilir.

Parçacık Sürü Optimizasyonu (PSO) sürü tabanlı bir metasezgiseldir [Kennedy and Eberhart, 1995]. Doğadaki kuş ve balık sürülerinden esinlenerek geliştirilen bu metasezgiselde sürüdeki her bir parçacık kişisel ve sosyal tecrübelerinden faydalanarak arama uzayında dolaşırlar. PSO başlangıçta sürekli optimizasyon problemleri için geliştirilmiş olup daha sonra bazı ayrık optimizasyon problemleri için de kullanılmıştır. Uygulandığı ayrık problemlerden bazıları gezgin satıcı problemi [Wang et al, 2003], iş planlama problemi [Liao et al, 2007] ve oryantiring problemidir [Şevkli et al, 2007]. Bu verilen örnek problemlerin ilk ikisinde çözümde eleman seçimi yoktur, sadece sıralama önemlidir. Oryantiring probleminde ise çözümde eleman seçimi ile birlikte sıralamanın da önemi vardır.

Sadece sıralamanın önemli olduğu problemler ve sıralama ile seçimin birlikte önemli olduğu problemlerin dışında bir de sadece seçimin önemli olduğu ayrık optimizasyon problemleri vardır. Bahsedilen üç tipteki problem arasında zorluk açısından bir karşılaştırma yapılamaz. Bu problem tipleri yapısal olarak birbirlerinden farklı problemlerdir. Örneğin seçim ve sıralamanın önemli olduğu bir problemi çözen bir algoritmayı, sadece seçimin önemli olduğu bir problemin çözümünde kullandığımızda, sıralama kriterine gereksiz vurgu yapılacağı için algoritma verimsiz çalışabilir. Sıralamanın önemli olmadığı bir problem için bunun bilincinde ve bunu dikkate alarak geliştirilen bir algoritmanın daha verimli çalışması beklenir.

Yaptığımız araştırmalarda PSO'nun sadece seçimin önemli olduğu sıralamanın önemsiz olduğu alt küme tabanlı problemlere uygulandığına rastlanmamıştır. Bu çalışmada PSO'nun bu tip problemlere uygulanabilirliği araştırılmıştır. Bunun için bu tip problemleri hız ve çözüm kalitesi açısından, diğer metasezgiseller seviyesinde çözen bir PSO modeli geliştirmek hedeflenmiştir. Böylelikle, birçok problemin çözümünde başarı ile kullanılan PSO ile gerçek hayatta çokça uygulama bulan pek çok problemi barındıran alt küme tabanlı ayrık problemlerin de başarılı bir şekilde çözülebileceğinin gösterilmesi amaçlanmıştır.

Bu çalışma kapsamında alt küme tabanlı problemlere uygulanabilecek bir PSO modeli geliştirilmiştir. Bu modele, A harfi "Alt küme"yi belirtecek şekilde PSO-A adı verilmiştir. PSO-A modelinin çalışması kısaca şöyledir. Her bir parçacık, içerisinde seçim yapılacak kümenin elemanlarından birer birer olarak her turda bir çözüm oluşturur. Bu seçimler yapılırken parçacığın kişisel ve sosyal tecrübelerinden yararlanılarak bu çalışma kapsamında geliştirilen olasılıklar kullanılır. Bu olasılıkların etkisi ile her parçacık, kendi ulaştığı en iyi çözümde ve komşularının ulaştığı en iyi çözümde bulunan elemanları seçmeye teşvik edilir. Alt kümeye eklendiğinde kısıtlamaları bozmayan eleman kalmayana kadar bu işleme devam edilir. Dolayısıyla hiç bir zaman olurlu çözüm olmayan bir alt küme oluşturulmaz. Bu işleme belirlenen bir döngü sayısı kadar devam edilir.

PSO-A modeli, detayları ikinci bölümde anlatılacak olan üç problem üzerinde test edilmiştir. Bu problemler maksimum klik problemi, çok boyutlu sırt çantası

problemi ve kenar kapsama problemidir. Bu problemler, üzerinde çokça çalışılan ve gerçek dünyada çok yerde uygulama alanı bulan problemler oldukları için seçilmişlerdir. Bu problemler ile yapılan testler sonucunda PSO-A modelinin maksimum klik ve kenar kapsama problemlerine başarıyla uygulandığı ancak sırt çantası probleminde başarısız olduğu görülmüştür.

Bu çalışmanın literatüre katkıları şunlar olmuştur;

- Alt küme tabanlı ayırık optimizasyon problemleri için bir parçacık sürü optimizasyonu modeli önerilmiştir. Bu modelin alt küme problemlerine başarı ile uygulanabileceği örneklerle gösterilmiştir.
- Bu geliştirilen model uygulanarak üç problem için çözüm üreten algoritmalar sunulmuştur. Bu algoritmaların karşılaştırmalı deneysel analizleri tartışılmıştır.
- Model parametreleri için uygun değerler araştırılmıştır. Parametrelerin çözüm kalitesine etkisi deneysel ve anlamsal olarak tartışılmış ve grafikler ile desteklenmiştir.

İkinci bölümde optimizasyon problemleri hakkında detaylı bilgi verilmiştir. Bu bölümde önce optimizasyon problemleri ile ilgili gerekli tanımlar verilmiş daha sonra da PSO-A modeli uygulanarak test edilen problemler anlatılmıştır. Üçüncü bölümde metasezgiseller açıklanmıştır. Bu bölümde PSO ile birlikte PSO'ya benzerlikleri dolayısıyla karınca kolonisi optimizasyonu ve genetik algoritma metasezgiselleri de anlatılmıştır. PSO-A modeli hakkında detaylı bilgi dördüncü bölümde verilmiştir. Bu bölümde önce modelin yapısı ile ilgili tanımlar verilmiştir. Daha sonra modelin çalışma prensipleri anlatılmıştır. Son olarak da PSO-A modeli ile üretilen algoritmaların karmaşıklığı hakkında bilgi verilmiştir. Beşinci bölümde programlama detayları anlatılmıştır. Gerçekleştirilen deneyler altıncı bölümde anlatılmıştır. PSO-A'nın üç örnek alt küme problemine uygulanışı sonrasında elde edilen sonuçlar bu bölümde gösterilmiştir. Bu bölümde ayrıca parametre seçimlerinin sonuçlar üzerindeki etkisi deneysel ve teorik olarak incelenmiştir. Yedinci bölümde ise sonuçlar tartışılmıştır.

2. OPTİMİZASYON PROBLEMLERİ

Optimizasyon, belirli şartlar ve kısıtlamalar altında en iyi sonucu alma sürecidir. Örneğin bir sistemi tasarlama ve gerçekleştirme aşamalarında mühendislerin çeşitli teknolojik ve idari kararlar almaları gerekir. Bu kararların nihai amacı ya harcanan çabayı en aza indirmek ya da istenen kazancı en çok hale getirmektir. Harcanan çaba ve istenen kazanç bazı karar değişkenlerinin bir fonksiyonu olarak ifade edilebileceği için, optimizasyon, bir fonksiyonun belirli şartlar altında en az ya da en çok olduğu durumu bulmak şeklinde tanımlanabilir.

2.1. Tanımlar

Bu bölümde önce optimizasyon probleminin matematiksel bir tanımı verilecektir. Ardından, çözüm uzaylarının tiplerine göre optimizasyon problemlerinin sınıflandırması yapılacaktır.

Tanım 2.1. Optimizasyon Problemi [Papadimitriou and Steiglitz, 1998] :

Bir *optimizasyon problemi*, olurlu (problem kısıtlarını sağlayan) çözümlerden oluşan bir $\mathcal{S}$ kümesi ile amaç fonksiyonu olan bir $\varphi: \mathcal{S} \rightarrow \mathbb{R}$ 'den oluşan $(\mathcal{S}, \varphi)$ ikilisidir. Problem, $\mathcal{S}$ kümesi içerisinde, $\forall S \in \mathcal{S} \varphi(S^*) \leq \varphi(S)$ (ya da $\forall S \in \mathcal{S} \varphi(S^*) \geq \varphi(S)$) olan bir S^* çözümü bulmaktır. Böyle bir S^* çözümüne *evrensel optimal çözüm* ya da sadece *optimal çözüm* adı verilir.

Bir optimizasyon problemi *enazlama* ya da *ençoklama* problemi olabilir. Problem, eğer optimal çözümün amaç fonksiyonundaki görüntüsü diğer bütün çözümlerden daha küçük ya da eşit olması isteniyorsa enazlama, büyük ya da eşit olması isteniyorsa ençoklama problemidir.

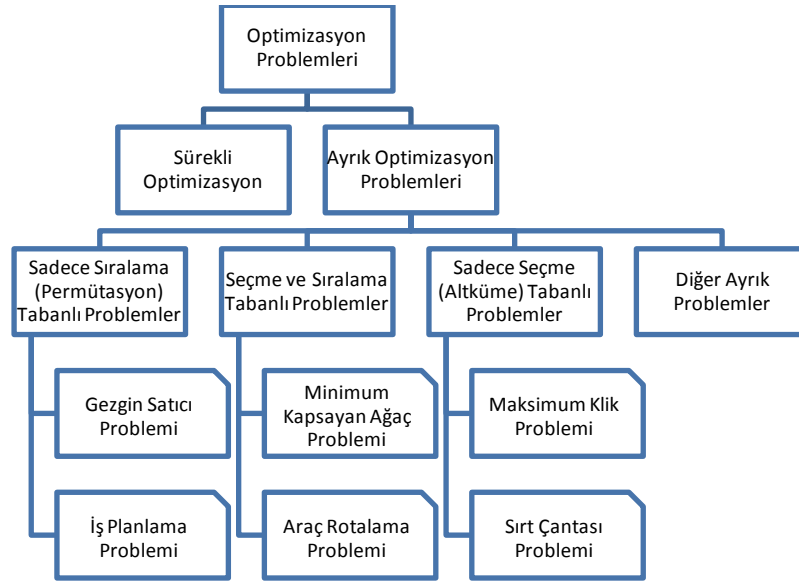
Bu tanımdan da anlaşılacağı üzere, bir optimizasyon problemi belirli bir çözüm uzayında, belirli bir amaç fonksiyonu değerini en küçük (veya en büyük) yapan elemanı bulma problemidir. Örnek olarak, gezgin satıcı problemini (traveling salesman problem) ele alalım. Bu durumda elimizde yönsüz, ağırlıklı bir tam çizge

vardır. Bu çizgenin düğümlerinin herhangi bir sıralanışı bizim için bir çözüme karşılık gelir. Bu sebeple çözüm uzayı çizgenin düğümlerinin bütün farklı permütasyonları yani sıralanışlarıdır. Örneğin çizge üç düğümden oluşuyor ve bu düğümlerin isimleri a, b, c ise, çözüm uzayı $\{abc, acb, bac, bca, cab, cba\}$ kümesidir. Bu problemde her çözümün amaç fonksiyonu değeri çözümün üzerinden geçtiği kenarların ağırlıkları toplamına eşittir. Bu problem bir enazlama problemidir. Gezgin satıcı probleminde optimal çözüm, amaç fonksiyonu değeri en küçük olan düğüm sıralamasıdır.

Yukarıdaki örnekte çözüm uzayı sonlu sayıda elemandan oluşmuştur. Bazı problemlerin çözüm uzayı reel sayılar ya da reel sayıların sayılamaz sonsuz elemanlı bir alt kümesi olabilir. Çözüm kümesi sayılamaz sonsuz sayıda olan bu tip problemler sürekli optimizasyon problemleridir. Sürekli optimizasyon problemleri genelde reel sayılı bir fonksiyonun değerini minimum yapmak şeklinde karşımıza çıkar. Bu fonksiyonlar çoğu zaman analitik yöntemlerle çözümü zor olan fonksiyonlardır.

Çözüm kümesi sonlu ya da sayılabilir sonsuz olan problemler ise ayrık optimizasyon problemleridir. Ayrık optimizasyon problemlerinin belli başlılarını üç grupta incelemek mümkündür. Birinci grupta sadece sıralamanın (permütasyon) önemli olduğu problemler vardır. Bu gruptaki problemlere örnek olarak gezgin satıcı problemi ve iş planlama problemi gösterilebilir. İkinci grupta sıranın önemli olduğu seçim tabanlı problemler vardır. Bu gruptaki problemlere örnek olarak en kısa yol problemi, minimum kapsayan ağaç problemi ve araç rotalama problemi gösterilebilir. Üçüncü grupta ise sıranın önemli olmadığı seçim (alt küme) tabanlı problemler vardır. Bizim çalışacağımız problemler de bu tipteki problemlerdir. Bu gruptaki problemlere örnek olarak maksimum klik problemi, sırt çantası problemi ve kenar kapsama problemi gösterilebilir. Bölüm 2.2’de bu problemler detaylı olarak anlatılacaktır.

Optimizasyon problemlerinin sınıflandırma şeması Şekil 2.1’de verilmiştir.



Şekil 2.1. Optimizasyon Problemlerinin Sınıflandırması

2.2. Alt Küme Bulma Tabanlı Optimizasyon Problemleri

Bu sınıftaki problemlerde çözüm uzayı belirli bir kümenin belirli şartları sağlayan alt kümelerinden oluşan kümedir. Bu problemleri formal olarak şu şekilde tanımlayabiliriz.

Tanım 2.2. Alt Küme Tabanlı Optimizasyon Problemi [Solnon and Bridge, 2006]:

U bir küme olsun. $\mathfrak{S} \subseteq P(U)$, U kümesinin bütün olurlu alt kümelerinden oluşan küme ise, $\mathfrak{S}$ çözüm uzayıdır. (Burada $P(U)$, U kümesinin bütün alt kümelerinden oluşan kümeyi temsil eder.) $\varphi: \mathfrak{S} \rightarrow \mathbb{R}$, çözüm kümesinden reel sayılara tanımlı amaç fonksiyonu olsun. $(U, \mathfrak{S}, \varphi)$ alt küme tabanlı bir optimizasyon problemini temsil eder. Problem, $\mathfrak{S}$ kümesi içerisinde, $\forall S \in \mathfrak{S} \varphi(S^*) \leq \varphi(S)$ (ya da $\forall S \in \mathfrak{S} \varphi(S^*) \geq \varphi(S)$) olan bir S^* çözümü bulmaktır. Bundan sonra kolaylık olması amacıyla alt küme tabanlı bir optimizasyon probleminde alt kümeleri problemin çözümü olacak olan U kümesine *evrensel küme*, bu kümenin her bir elemanına *nokta* adını vereceğiz.

Bu tanım alt küme problemlerini genel olarak tanımlamakla birlikte, literatürdeki problemler için biraz fazla serbest bir tanımdır. Literatürdeki bu tip problemlerin doğal olarak birkaç ortak özelliği daha vardır. Örneğin, hiçbir olurlu çözümün olmadığı, sadece tek sayıda elemana sahip alt kümeleri olurlu kabul eden ya da sadece belli bir sayıda elemana sahip alt kümeleri olurlu kabul eden problemler yukarıdaki tanıma uymalarına rağmen literatürdeki problemlerde bu kısıtlar genel olarak görülmez. En çok bilinen ve üzerinde çok çalışılan alt küme problemlerini genel olarak iki sınıf altında toplamak mümkündür. Birinci tipteki problemlerde alt kümenin “büyümesi” amaçlanır. Alt küme büyüdükçe kısıtların sağlanması zorlaşır ve amaç fonksiyonu değeri artar. Belli bir büyümeden sonra alt küme daha fazla büyüyemez yani alt kümeye daha fazla eleman eklenemez hale gelir. Bu tipteki bir problemde, olurlu bir alt kümeden bir eleman çıkardığımızda yine olurlu bir alt küme elde ederiz. İkinci tipteki problemlerde ise durum tam tersidir. Bu problemlerde alt kümenin “küçülmesi” amaçlanır. Aşağıda bu tip problemlerin formal tanımı verilmiştir.

Tanım 2.3. Büyüme Amaçlayan Alt Küme Problemi :

$\Pi = (U, \mathfrak{S}, \varphi)$ alt küme tabanlı bir en çoklama optimizasyon problemi olsun. Eğer Π problemi aşağıdaki şartları sağlıyorsa büyüme amaçlayan bir alt küme problemidir.

- i) $\emptyset \in \mathfrak{S}$
- ii) $\forall S \in \mathfrak{S} \forall s \in S \ S \setminus \{s\} \in \mathfrak{S}$
- iii) $\forall S \in \mathfrak{S} \forall u \in U \setminus S \ ((S \cup \{u\} \in \mathfrak{S}) \rightarrow (\varphi(S \cup \{u\}) \geq \varphi(S)))$

Yukarıdaki tanımı sözel olarak ifade etmemiz gerekirse; boş kümenin olurlu olduğu, olurlu bir kümeden bir nokta çıkarıldığında yine olurlu bir küme elde edilen ve olurlu bir kümeye bir nokta eklendiğinde küme olurlu kalıyorsa amaç fonksiyonu değerinin azalmayacağı problemlere büyüme amaçlayan alt küme problemleri diyeceğiz.

Tanım 2.4. Küçülme Amaçlayan Alt Küme Problemi :

$\Pi = (U, \mathfrak{S}, \varphi)$ alt küme tabanlı bir enazlama optimizasyon problemi olsun. Eğer Π problemi aşağıdaki şartları sağlıyorsa küçülme amaçlayan bir alt küme problemidir.

- i) $U \in \mathfrak{S}$
- ii) $\forall S \in \mathfrak{S} \forall u \in U \setminus S \ S \cup \{u\} \in \mathfrak{S}$
- iii) $\forall S \in \mathfrak{S} \forall s \in S \ (S \setminus \{s\} \in \mathfrak{S}) \rightarrow (\varphi(S \setminus \{s\}) \leq \varphi(S))$

Yukarıdaki tanımı sözel olarak ifade etmemiz gerekirse; evrensel kümenin olurlu olduğu, olurlu bir kümeye bir nokta eklediğimizde yine olurlu bir küme elde edilen ve olurlu bir kümeden bir nokta çıkardığımızda küme olurlu kalıyorsa amaç fonksiyonu değerinin artmayacağı problemlere küçülme amaçlayan alt küme problemleri diyeceğiz.

Alt küme tabanlı problemlere örnek olarak Tablo 2.1’de verilen problemler gösterilebilir. Bu çalışmada, geliştirilen model bu problemlerden üçüne uygulanarak test edilmiştir. Bu yüzden bu problemlerin tanımları aşağıda detaylı olarak verilmiştir. Tablo 2.1.’deki 8 örnekten 4’ü büyüme, 3’ü küçülme amaçlayan alt küme problemi tanımını sağlar. Maksimum mantıksal sağlama problemi (Maximum Boolean Satisfaction) ise iki sınıfa da dahil olmayan bir problemidir.

Tablo 2.1. Alt Küme Tabanlı Problemlere Örnekler

Maksimum Klik Problemi (Maximum Clique Problem)	Büyüme
Çok Boyutlu Sırt Çantası Problemi (Multidimensional Knapsack Problem)	Büyüme
Maksimum Bağımsız Küme(Maximum Independent Set)	Büyüme
Maksimum Küme Paketleme(Maximum Set Packing)	Büyüme
Kenar Kapsama Problemi (Minimum Vertex Cover)	Küçülme
Küme Kapsama Problemi(Set Cover Problem)	Küçülme
Kenar Hükmetme Problemi(Edge Dominating Set)	Küçülme
Maksimum Mantıksal Sağlama (Maximum Boolean Satisfaction)	-

2.2.1. Maksimum Klik Problemi

Maksimum klik problemi (MKP) çizgeler üzerinde tanımlanmış alt küme tabanlı ayrık bir optimizasyon problemidir. V düğüm kümesi, E kenar kümesi olmak üzere $G = (V, E)$ yönsüz bir çizge olsun. $C \subseteq V$ düğümler kümesinin bir alt kümesi olmak üzere eğer $\forall a, b \in C \ a \neq b \rightarrow (a, b) \in E$ yani C içerisindeki birbirinden farklı her iki elemanın arasında bir kenar var ise C 'ye klik (clique) adı verilir. MKP, bir çizge verildiğinde bu çizge içerisindeki en fazla elemana sahip kliği bulma problemidir.

MKP; evrensel kümesi, V yani çizgenin bütün düğümleri; çözüm uzayı, V 'nin klik olma şartını sağlayan bütün alt kümeleri; amaç fonksiyonu, çözümün (alt kümenin) eleman sayısı olan ençoklama amaçlayan alt küme tabanlı ayrık bir problemidir. Boş küme klik olduğu, herhangi bir klikten bir nokta çıkarılırsa yine klik kalacağı ve bir kliğe bir eleman eklendiğinde klik olma durumu korunuyorsa amaç fonksiyonu değeri artacağı için MKP büyüme amaçlayan bir alt küme problemidir.

MKP'nin NP-Zor bir problem olduğu Richard Karp tarafından 3SAT probleminin MKP'ye polinom zamanda indirgenebildiği gösterilerek 1972'de kanıtlanmıştır [Karp, 1972].

MKP üzerinde birçok sezgisel yöntemler ve metasezgiseller uygulanmıştır. Bu yöntemlerde temel olarak, boş bir kümeden başlanarak, sezgisel seçimlere göre kümeye düğümler eklenmesi temel alınmıştır. Burada alt kümeye eklenecek düğüm seçilirken derecesi en yüksek olan düğümün seçimi ya da bu alt kümedeki elemanlarla ilişkisi en fazla olan düğümün seçimi gibi sezgisel seçimler kullanılmıştır. Kümeden eleman çıkarırken de benzer sezgisel seçimler kullanılmıştır.

MKP'ye uygulanan metasezgisellerden bazıları Tablo 2.2'de listelenmiştir.

Tablo 2.2. MKP'ye Uygulanan Bazı Metasezgiseller

Tabu Arama [Soriano and Gendreau, 1993]
Genetik Algoritma [Marchiori, 2002]
Karınca Kolonisi [Fenet and Solnon, 2004]
Mutasyon Tabanlı Bir Evrimsel Algoritma [Zhang, 2005]
Benzetim Tavlama Metodu [Geng et al, 2007]

2.2.2. Çok Boyutlu Sırt Çantası Problemi

Çok boyutlu sırt çantası problemi (ÇSP) de alt küme tabanlı ayrık bir optimizasyon problemidir. Bu problemde n tane nesne ve m boyutlu bir sırt çantası vardır. Sırt çantasının kapasitesi m boyutlu bir vektördür. Her nesnenin faydası ve m boyutlu bir ağırlığı vardır. Amacımız bu çantaya sığabilecek ve en fazla faydayı getiren nesneleri bulmaktır.

Daha formal bir ifade ile, U bütün nesnelerden oluşan küme ise amacımız $\sum_{i=1}^n p_i x_i$ fonksiyonunu en çoklayan $S \subseteq U$ alt kümesi bulmaktır. Bunu yaparken $\sum_{i=1}^n \sum_{j=1}^m r_{ij} x_i \leq b_j$ şartının da sağlanması gerekir.

Burada p faydayı gösterir ve pozitif reel sayılardır. x de bir nesnenin S 'de bulunup bulunmadığını gösterir. x_i , 0 ve 1 değerlerini alır. 0 değerini alması i . nesnenin S 'de bulunmadığı, 1 değerini alması bulunduğu anlamına gelir. r_{ij} i . nesnenin j . boyuttaki ağırlığını, b_j ise sırt çantasının j . boyuttaki kapasitesini gösterir.

ÇSP; evrensel kümesi, problemdeki bütün nesneler; çözüm uzayı, sırt çantası kapasitesini aşmayan nesnelerden oluşan alt kümeler; amaç fonksiyonu, sırt çantası içerisindeki nesnelerin faydalarının toplamı olan ençoklama amaçlayan alt küme tabanlı ayrık bir problemidir. Boş küme sırt çantasına sığıldığı, sırt çantasına sığan bir kümeden bir nokta çıkarırsak yine sırt çantasına sığacağı ve sırt çantasına sığan bir kümeye bir nokta eklendiğinde sırt çantasına sığma şartı korunuyorsa amaç fonksiyonu değeri artacağı için ÇSP büyüme amaçlayan bir alt küme problemidir.

ÇSP de NP-Zor bir problemidir[Puchinger et al, 2010]. Birçok metasezgisel bu problem üzerinde denenmiştir. Bunlardan başlıcaları Tablo 2.3’de listelenmiştir.

Tablo 2.3. ÇSP’ye Uygulanan Bazı Metasezgiseller

Genetik Algoritma [Chu and Beasley, 1998]
Tabu Arama [Hanafi and Freville, 1998]
Evrimsel Algoritma [Fidanova, 2002]
Benzetim Tavlama Metodu [Qian abd Ding, 2007]
Karınca Kolonisi Optimizasyonu [Kong et al., 2008]

2.2.3. Kenar Kapsama Problemi

Kenar kapsama problemi (KKP) çizgeler üzerinde tanımlanan alt küme tabanlı bir ayrık optimizasyon problemidir. Bu problemde amaç bir çizgenin kenarlarını kapsayan en az sayıda eleman içeren düğüm alt kümesini bulmaktır. Bir kenarın kapsanması demek bu kenarın birleştirdiği iki düğümden en az birinin çözümde bulunması demektir.

KKP’yi daha formal olarak şu şekilde tanımlayabiliriz. $G = (V, E)$ yönsüz bir çizge olsun. Bir $S \subseteq V$ kümesi eğer $\forall i, j \in V ((i, j) \in E) \rightarrow (i \in S \text{ veya } j \in S)$ şartını sağlıyorsa bir kapsayan kümedir. KKP, V ’nin aşağıdaki şartları sağlayan bir S alt kümesini bulma problemidir :

- i) S bir kapsayan kümedir.
- ii) $\forall T \subseteq V (T \text{ kapsayan küme}) \rightarrow (|S| \leq |T|)$

KKP; evrensel kümesi, V yani çizgenin bütün düğümleri; çözüm uzayı, V ’nin bütün kapsayan kümeleri; amaç fonksiyonu, çözümün (alt kümenin) eleman sayısı olan enazlama amaçlayan alt küme tabanlı ayrık bir problemidir. Evrensel küme kapsayan küme olduğu, kapsayan bir kümeye bir nokta eklendiğinde yine kapsayan küme elde edileceği ve kapsayan bir kümeden bir nokta çıkarıldığında kapsayan küme olma özelliği korunuyorsa amaç fonksiyonu değeri azalacağı için KKP küçülme amaçlayan bir alt küme problemidir.

Klik problemi bu probleme indirgenemediği için bu problemin de NP-Zor bir problem olduğu bilinmektedir [Garey and Johnson, 1979]. Bu problem için uygulanan metasezgisellerden bazıları Çizelge 2.4.'de listelenmiştir.

Tablo 2.4. KKP'ye Uygulanan Bazı Metasezgiseller

Melez Genetik Algoritma [Kotecha and Gambhava, 2003]
Karınca Kolonisi [Gilmour and Dras, 2006]
Benzetim Tavlama Metodu [Xu and Ma, 2006]

Yukarda detaylı olarak anlatılan MKP, ÇSP ve KKP problemlerinin özellikleri özet olarak Tablo 2.5'de gösterilmiştir.

Tablo 2.5. MKP, ÇSP ve KKP – Özet

	MKP	ÇSP	KKP
Nokta	Düğüm	Nesneler	Düğüm
Evrensel Küme	V	Bütün Nesneler Kümesi	V
Olurlu Alt küme	Klikler	Sırt Çantasına Sığan Nesneler Kümesi	Kapsayan Kümeler
Çoklama/Azlama	Çoklama	Çoklama	Azlama
Amaç Fonksiyonu	Eleman sayısı	Toplam fayda	Eleman sayısı
Büyüme/Küçülme	Büyüme	Büyüme	Küçülme

n elemanlı bir kümenin 2^n tane alt kümesi olacağı için alt küme tabanlı bir problemin evrensel kümesinde n nokta varsa çözüm uzayında 2^n tane çözüm olması mümkündür. Bu nedenle büyük n sayıları için bütün alt kümelerle bakarak en iyi çözümü bulmak makul bir zamanda mümkün olmamaktadır. Bu yüzden üçüncü bölümde anlatılacağı gibi bu tip problemleri çözmek için genellikle sezgisel yöntemler ve metasezgiseller kullanılır.

3. METASEZGİSELLER

Bilim, mühendislik, ekonomi gibi alanlardaki gerçek dünya optimizasyon problemlerinin birçoğu karmaşık ve çözümü zor problemlerdir. Böyle problemlerin makul bir zamanda kesin olarak çözümleri mümkün değildir. Bu yüzden bu tarz problemleri çözmek için yaklaşık çözüm veren algoritmalar kullanma yoluna gidilmiştir. Yaklaşık çözüm veren algoritmalar adından da anlaşılacağı gibi her zaman en iyi çözümün bulunacağını garanti etmezler. Ancak kesin çözüm veren algoritmalarından çok daha hızlı çalıştıkları ve kabul edilebilir derecede iyi sonuçlar verdikleri için tercih edilirler.

Yaklaşık çözüm veren algoritmalar genellikle çözüm uzayının tamamını araştırmanın çok zor olduğu büyük problemlere uygulanırlar. Bu yöntemler ile çözüm uzayının verimli bir şekilde gezilmesi sağlandığı için çok daha hızlı bir şekilde sonuca ulaşılır.

Yaklaşık çözüm veren algoritmalar sezgisel ve metasezgisel olmak üzere iki sınıfa ayrılabilirler. Sezgisel algoritmalar, belirli bir problem için geliştirilen ve uygulanan probleme bağımlı yöntemlerdir. Bu algoritmalar geliştirilirken üzerinde kullanılacağı problemin bilgilerinden yararlanır. Bu yüzden sadece bu problem için iyi sonuçlar vermesi beklenir.

Metasezgiseller ise, çeşitli optimizasyon problemlerine uygulanabilen, problemten bağımsız daha genel yöntemlerdir. Genellikle doğadan esinlenerek geliştirilen bu yöntemler, son yıllarda çokça çalışılmış ve bir çok optimizasyon problemi üzerinde başarı ile uygulanmıştır. Metasezgisellerin genel özellikleri Bölüm 3.1’de anlatılmıştır.

3.1. Genel Özellikleri

3.1.1. Model Oluşturma

Problemi metasezgisellerle çözmek için çözümün kodlanmasıdır. Yapılacak kodlama olası bütün çözümleri ifade etmeli, çözümler arasında iyileşmeye bağlı bir geçişe imkan vermeli ve bu geçiş zaman ve yer karmaşıklığı bakımından verimli bir şekilde sağlamalıdır. Verilen bir alfabeye göre sembollerin doğrusal bir şekilde temsil edildiği lineer temsil (linear representation); daha karmaşık yapı problemler için lineer olmayan (nonlinear) temsil; çözüm (fenotip) ve kod (genotip) arasındaki geçişin bir bağıntı ile (birebir, bire çok veya çok bir) ifade edildiği çözüm-temsil (solution-representation) yöntemleri kullanılarak problem çeşitli şekillerde modellenebilir.

3.1.2. Amaç Fonksiyonu

Amaç fonksiyonu metasezgisellerin tasarımında önemli bir yere sahiptir. Arama uzayında ‘iyi’ çözümlere ilerlenmesini sağlar. Dikkatli bir şekilde tanımlanmalı ve aramayı olumsuz çözümlere yönleltmemelidir. Amaç fonksiyonunun çok karmaşık olduğu zamanlarda, daha hızlı çalışan yönlendirici amaç fonksiyonları (guiding objective function) kullanılmaktadır.

3.1.3. Olursuz Çözümlerle Başa Çıkma

Çözüm uzayında ilerlerken lineer ya da lineer olmayan ve eşitlik ya da eşitsizlik olursuz çözümleri ile karşılaşılabilir, bu durumlarda reddetme, cezalandırma, onarma gibi stratejiler uygulanabilmektedir.

3.1.4. Parametreleri Ayarlama

Metasezgiseller genel yapı itibarıyla birçok parametreye sahiptirler. Bu parametrelerin değerleri problemlere göre değişebilmektedir. Algoritmaların performansı için bunların düzgün bir şekilde ayarlanması çok önemlidir. Bu

parametreler algoritma çalışmaya başlamadan önce (offline) ya da arama çözüm uzayında ilerlerken (online) ayarlanabilir.

3.1.5. Performans Analizi

Metasezgisellerin performans analizleri şu şekilde tanımlanabilir. Öncelikle problem örnekleri belirlenir. Algoritmaların belirli bir standarttaki iyi çözümü ne kadar sürede buldukları (computational effort), verilen belirli bir sürede ne kadar iyi bir sonuç elde ettikleri (solution quality), algoritmaların tekrarlanması sonucu oluşan çözümlerin kalite farkı (robustness) karşılaştırılır. Sonuçlar kapsamlı bir şekilde sunulur, tanımlanan hedeflere göre analiz yürütülür, deneylerin başkaları tarafından tekrarlanılabilirliği de önemlidir.

Metasezgiseller kullandıkları çözümlerin sayılarına göre Tek Çözüm Tabanlı Metasezgiseller ve Sürü Tabanlı Metasezgiseller olmak üzere ikiye ayrılırlar.

3.1.6. Tek Çözüm Tabanlı Metasezgiseller

Tek çözüm tabanlı metasezgiseller verilen tek bir çözümü iyileştirerek çalışırlar. Bu iyileştirme komşuluklar üzerinde *yürümek* ya da çözüm uzayında tanımlanan arama yollarında *ilerlemek* olarak tanımlanabilir. Yürümler yinelemeli bir prosedürle çözüm uzayında o anki çözümden yeni bir çözüme geçiş şeklinde gerçekleştirilir.

Komşuluk : Bir komşuluk fonksiyonu $N:S \rightarrow 2^S$ çözüm uzayındaki her s çözümüne bir çözümler kümesi $N(s) \subset S$ atar. Bir ilerleme operatörü m ile s' 'de küçük bir değişiklik meydana getirilerek bir komşu çözüm $s' \in N(s)$ oluşturulur. Eğer bir çözüm s bütün komşularından daha iyi ise buna yerel optimum denir. Buradan yeni bir başlangıç çözümü üreterek, objektif fonksiyonu azaltmayan komşuluk seçerek, komşuluğu değiştirerek ya da farklı bir amaç fonksiyonu kullanarak başka bir çözüme geçilebilir. Seçilecek yeni komşu, amaç fonksiyonunu ilk arttıran komşu, en iyi arttıran komşu ya da rastsal olarak seçilebilir.

İklendirme : Metasezgisellerin çalışması bir başlangıç çözümü alıp bunun iyileştirilmesine dayanmaktadır. Bu ilk çözüm rastsal ya da aç gözlü bir algoritma ile üretilebilir. Üretim yöntemi problemlere göre değişiklik gösterir. Aç gözlü bir sezgisel ile üretilen çözüm her zaman daha iyi olmayabilir. Yine de iyi çözümlerin komşuluğunda da iyi çözümler bulunduğu fikri nedeniyle çoğunlukla daha iyi sonuçlar vermesi beklenir.

Yerel arama, benzetim tavlama, yinelemeli yerel arama, değişken komşuluk arama tek çözüm tabanlı metasezgisellere örneklerdir.

3.1.7. Sürü Tabanlı Metasezgiseller

Sürü tabanlı metasezgiseller bir grup çözümü yinelenabilir şekilde daha iyi hale getiren yöntemler olarak görülebilirler. Öncelikle başlangıç sürüsü üretilir. Daha sonra yeni bir sürü çözüm üretilerek belirli bir seçim prosedürüne göre bir sonraki sürü oluşturulur. Bu işlem yinelemeli olarak belirli bir koşul sağlanana kadar devam eder. Yeni bir sürü üretimi ve bir sonraki sürünün seçimi sadece şimdiki sürüye bağlı (memoryless) ya da kaydedilmiş hafıza bilgisini içerecek şekilde (with memory) olabilir.

Arama Hafızası : Arama süresince elde edilen ya da kaydedilen bilgidен oluşur. Bir çok metasezgiselde arama hafızası sürü ile sınırlıdır. Evrimsel tabanlı bu tarz metasezgiselerde sürü seçimi ve yeniden üretilmesi mutasyon, çaprazlama, rekombinasyon gibi operatörlerin çözümlerin üzerine direkt uygulanmasıyla gerçekleştirilir. Yeni sürü şimdiki sürünün farklı özellikleri kullanılarak oluşturulur. Kara tahta tabanlı metasezgiselerde ise yeni bir sürü üretimi ayrı bir ortak hafıza kullanılarak gerçekleştirilir. Ortak hafızaya örnek olarak karınca kolonisi optimizasyonundaki feromon bilgisi ve parçacık sürü optimizasyonundaki sosyal tecrübe bilgisi gösterilebilir.

İklendirme : Başlangıç sürüsünün üretiminde ana şart farklılaşmayı sağlamaktır. Eğer başlangıç sürüsündeki çözümler yeteri kadar birbirlerinden farklı değil ise erken çökelmeye sebebiyet verebilirler. Rastsal üretim, farklılaştırmaya bağlı sıralı ve paralel üretim, sezgisel üretim şeklinde üretim yöntemleri vardır.

Rastasal üretimde çözüm temsilleri verilen bir alfabeden belirli bir dağılıma göre rastasal olarak üretilen değerlerin birleştirilmesi ile meydana getirilir. Sezgisel üretim, probleme özgü üretilen bir aç gözlü algoritma ile üretilir. Burada yine farklılaşmanın korunmasına dikkat edilmesi gerekmektedir. Farklılaştırmaya bağlı sırasal üretimde çözümler art arda ve üretilen yeni çözüm önceki çözümlerden olabildiğince farklı olacak şekilde, farklılaşmaya bağlı paralel üretimde ise belirli bir farklılaştırma kriteri (diversification criterion) dikkate alınarak çözümler tek seferde üretilir.

Durdurma Kriteri : Durdurma kriterleri durağan (static) ve uyarlamalı (adaptive) olmak üzere ikiye ayrılır. Yineleme sayısı, işlemci kaynaklarına konulacak bir sınır değeri durağan kriterlere örnek olarak verilebilir. Uyarlamalı kriterler daha önceden kestirilmesi zor kriterlerdir. Yeterli kalitede bir çözümün elde edilmesi buna örnek olarak verilebilir.

Genetik algoritma, parçacık sürü optimizasyonu, karınca kolonisi optimizasyonu, arı kolonisi, memetik algoritma sürü tabanlı metasezgisellere örnek olarak verilebilir.

3.1.8. Melez Metasezgiseller

Son yıllarda problemlere özel çözümlerde klasik metasezgisel tekniklerin yanında bunlarla birlikte başka metasezgiseller veya dallan ve sınırla (branch and bound) gibi farklı optimizasyon tekniklerinin kullanımı yaygınlaşmıştır.

3.2. Sürü Tabanlı Metasezgisellere Örnekler

Bizim çalışmamıza temel teşkil eden parçacık sürü optimizasyonu (PSO) ile Genetik Algoritma ve Karınca Kolonisi Optimizasyonunun çalışma mekanizmaları aşağıda anlatılmıştır. Bu iki metasezgiselin anlatılmasının nedenleri, bunların birçok açıdan PSO ile benzer karakter taşıması ve bizim geliştirdiğimiz PSO-A modeline benzerlik göstermeleri ile kullandığımız problemler için bunlar ile geliştirilmiş algoritmaların da olması ve karşılaştırma açısından bunların bilinmesinin önemli olmasıdır.

3.2.1. Karınca Kolonisi Optimizasyonu

Karınca kolonisi optimizasyonu (KKO) doğadaki karıncaların davranışlarından esinlenerek 1992’de Dorigo tarafından geliştirilmiş metasezgiseldir [Dorigo, 1992]. Doğada karıncalar iyi bir yiyecek kaynağı bulduklarında diğer karıncalarla iletişim kurmak için feromon salgırlar. Zaman içinde bu feromonlar iyi kaynakların yolları üzerinde yoğunlaşarak karıncaların yön tercihlerinde rehber olarak kullanılır. KKO da bu mekanizmadan esinlenerek geliştirilmiştir. Şekil 3.1’de KKO şablonu gösterilmiştir.

Algoritma temel olarak iki yinelemeli adımdan oluşmaktadır. Bunlar çözüm üretimi ve feromon güncellemesi adımlarıdır.

Algoritma : Karınca Kolonisi Optimizasyonu Şablonu

Girdi : Bir optimizasyon problemi

Çıktı : Bulunan en iyi çözüm ya da çözüm kümesi

1. Parametreleri ve feromon izini ilklendir
2. **do**
3. Her bir karınca için feromon izine bağlı çözüm oluştur
4. Feromon izini güncelle (Buharlaştırma, Güçlendirme)
5. **while** (sonlandırma koşulu karşılanana kadar)

Şekil 3.1. Karınca Kolonisi Optimizasyonu Şablonu

Çözüm Üretimi : Karıncalar olasılıksal aç gözlü seçim prosedürleri olarak görülebilir. Bir çözümü, çözümün bütün parçaları tamamlanıncaya kadar olasılıksal bir şekilde parça parça oluştururlar. Çözölmek istenen optimizasyon problemi, karıncaların yollar (path) oluşturduğu bir karar çizgesi olarak düşünülebilir. Karıncalar atanan bir başlangıç düğümünden başlayarak bir sonraki düğümlere bu düğümlerdeki feromon izi miktarını (ve problem bilgisini) içeren bir olasılıksal aç gözlü yöntem ile ilerleyip yollarını tamamlarlar. Gidilen her bir düğüm çözümün bir parçasını oluşturur ve yol tamamlandığında çözüm oluşmuş olur. Karıncalar gidilecek düğümleri aşağıdakilerden faydalanarak belirlerler:

Feromon İzleri : Feromon izleri “iyi” çözümlerin bilgisini tutar ve karıncalar tarafından yeni çözümlerin üretilmesinde rehberlik ederler. Yinelemeler sırasında dinamik olarak değışirler ve bütün karınca aramalarının hafızasını temsil ederler.

Probleme bağı sezgisel bilgiler: Probleme özgü bilgilerdir. Bunların kullanımı karıncaların daha iyi seçimler yapmasına yardımcı olur.

Feromon Güncellemesi : Feromon değeri üretilen çözümün kalitesine göre iki aşama ile güncellenir.

- **Buharlaşıma aşaması** : Karıncaların iyi çözümlere doğru erken çökmesine engel olmak ve farklılaşmayı (diversification) sağlamak için kullanılır. Belirli bir iterasyon miktarı sonrası iyi çözümlerde fazla bulunmayan düğümlere gidiş miktarının azalması düşüncesi için kullanılır. Feromon değeri aşağıdaki formüle göre güncellenir:

$$\tau_{ij} = (1 - \rho)\tau_{ij}, \forall i, j \in [1, n]$$

Burada $\rho \in]0,1]$ feromonların düşüş oranını belirler.

- **Güçlendirme Aşaması** : Üretilen çözümlere göre feromon değeri güçlendirilir. Çözüm elde edildikten sonra güçlendirme yapılabileceği gibi karıncaların her ilerlemesinde ya da bütün karıncaların çözümleri tamamlamaları sonrasında da güçlendirme yapılabilir. Çözüm kalitelerine göre, çözümlerin kalite sıralarına göre, sadece en iyi çözümlere olan aidiyete göre feromon değeri güçlendirilebilir.

3.2.2. Genetik Algoritma

Genetik algoritma (GA), doğal sistemlerdeki süreçleri anlamak için 1970’lerde J. Holland tarafından geliştirilmiştir [Holland, 1975]. Daha sonra 1980’lerde optimizasyon ve makine öğrenmesi gibi alanlarda kullanılmıştır [Goldberg, 1989]. Bir GA’da genellikle iki bireyden yeni bir birey oluşturan çaprazlama ve bir bireyi rastsal olarak değıştiren mutasyon sezgiselleri mevcuttur. Ayrıca yeni birey

oluşturacak ebeveynlerin seçiminde doğal seçim benzeri mekanizmalar kullanılır. Şekil 3.2’de GA şablonu gösterilmiştir.

Algoritma : Genetik Algoritma Şablonu

Girdi : Bir optimizasyon problemi

Çıktı : Bulunan en iyi çözüm ya da çözüm kümesi

1. N bireylik bir popülasyonu ilklendir
2. N bireyi değerlendir
3. **do**
4. N bireyden k tane birey oluştur
5. k bireyi değerlendir
6. k yeni bireyi popülasyondaki bireylerle değiştir
7. **while** (sonlandırma koşulu karşılanana kadar)

Şekil 3.2. Genetik Algoritma Şablonu

Ebeveyn Seçim Yöntemleri : Seçim mekanizması, evrimsel algoritmaların önemli bir parçasıdır. Genel prensip daha iyi olanın ebeveyn olma olasılığının daha yüksek olması üzerinedir. Seçim stratejisi hangi ebeveynlerin çiftleşme için seçileceği ve ne kadar yavru üreteceğini belirler. Bireylere zindelik (fitness) ataması, orantılı zindelik ve sıralama (rank) bazlı zindelik atamaları şeklinde iki farklı yöntemle yapılabilir. Orantılı zindelik bireylere mutlak bir zindelik ataması yaparken, sıralama bazlı olan göreceli bir atama (örneğin, bir sürü içindeki zindelik kriterine göre azalan bir listelemeye göre sıralama) yapar. Daha sonra, ebeveynler zindeliklerine göre aşağıdaki stratejilerden birine göre seçilirler.

- **Rulet Tekeri Seçimi :** Öncelikle her bireye sürü içerisinde zindeliğiyle orantılı bir seçilme olasılığı atanır. Daha sonra bunlar rulet tekeri üzerine oranlarınca yerleştirilerek olasılıksal seçimler yapılır. n bireyin seçimi için n bağımsız seçim yapılır.
- **Turnuva Seçimi :** Turnuva seçimi n elemanlı bir sürü içinden, $k < n$ olacak şekilde, k elemanın rastsal olarak seçilimi ve bu alt kümeden en iyi (zindelik değeri en yüksek) elemanın seçilimi işleminin m kez tekrarlanması sonucu m elemanın seçimine dayanır.

- **Sıralama Tabanlı Seçim** : Zindelik değeri yerine bireylerin zindelik değerlerine göre sıralaması kullanılır. Lineer ölçekleme, benzetim tavlama tabanlı Boltzmann seçimi buna örnek verilebilir.

Yeni Sürü Üretimi : Yeni çözümlerin oluşması sırasında bir önceki sürüden yukarıda anlatılan seçim teknikleri kullanılarak ebeveynlar seçilir. Bu ebeveynlere aşağıda anlatılan teknikler uygulanarak yeni sürü çözümleri elde edilir:

Mutasyon : Mutasyon operatörü tek bir çözüme uygulanır. Mutasyonlar seçili bireylerdeki küçük değişiklikleri temsil ederler. p_m olasılığı temsili her elementinin (gen) mutasyon ihtimalini gösterir. Mutasyon operatörü, bütün çözümlere gidebilme imkanı vermeli ve geçerli çözümler üretebilmesi gerekir. Ayrıca mutasyon operatörünün küçük değişiklikler üreterek çözüm uzayındaki yerellik (locality) prensibine bağlı kalınmasına dikkat edilmelidir.

Çaprazlama ve Yeniden Birleşim (Recombination) : Çaprazlama operatörünün ana amacı ebeveynlerdeki bazı karakteristik özelliklerin yeni nesillere aktarımının sağlanmasıdır. Ebeveyn sayısı iki veya ikiden çok olabilir. Çaprazlama yapılırken oluşacak yeni çözümün geçerli bir çözüm olabilmesine dikkat edilmelidir.

Sonraki Nesil Sürü Seçimi : Sonraki nesil seçilirken belirli bir kritere göre eski ve yeni nesilden seçimler yapılır. Yeni sürünün olduğu gibi aktarımı, seçkinlik (elitism) yöntemine göre en iyilerin bir sonraki nesile aktarımı gibi yöntemler vardır. Seçkinlik gibi yöntemler erken bir yerel en iyiye çökmeye sebep olabilirler. Bu nedenle farklılaştırmayı arttırmak için olasılıksal (stochastic) ve deterministik yöntemler kullanılarak bazı kötü bireyler de sonraki nesile aktarılabilir. Kara tahta tabanlı metasezgisellerde ise açık bir seçim aşaması yoktur, yeni sürü ortak hafızayı günceller.

3.2.3. Parçacık Sürü Optimizasyonu

Parçacık sürü optimizasyonu (PSO), 1995’de Kennedy ve Eberhart tarafından geliştirilmiş, esin kaynağı doğadaki balık ve kuş sürüleri olan sürü tabanlı bir

metasezgiseldir [Kennedy ve Eberhart, 1995]. Sürüdeki her bir bireye *parçacık* adı verilir. Sürüdeki parçacık sayısını SB (sürü boyutu) ile göstereceğiz.

PSO'da her bir parçacığın (sürüdeki bireyin) pozisyon, hız ve hafıza bileşenleri vardır. Yinelemeli bir metasezgisel olan PSO'da, parçacıklar her turda bu bileşenlerini güncelleyerek daha “iyi” çözümlere ulaşmaya çalışırlar. Pozisyon, parçacığın arama uzayındaki bulunduğu konum, yani çözüme karşılık gelir. Hız ise, parçacığın pozisyonunun değişim yönü ve şiddetini ifade eder. Bir parçacığın pozisyonunun amaç fonksiyonundaki görüntüsüne *GüncelDeğer* ismini vereceğiz.

Parçacıkların hafızası, kişisel ve sosyal tecrübe bilgilerini içerir. Parçacıklar üzerinde bir komşuluk topolojisi mevcuttur. Komşuluk topojisi hangi parçacıkların birbiri ile komşu olduğunu belirtir. PSO'da çeşitlik komşuluk topolojileri kullanılmaktadır [Clerc, 2006]. Her bir parçacık sadece komşusu olduğu parçacıklar ile bilgi alışverişi yapar. Parçacığın o ana kadar bulduğu en iyi çözüm *K-Eniyi*, parçacığın komşuları dikkate alındığında bulunan en iyi çözüm ise *S-Eniyi* ile ifade edilir ve bunlar PSO'da sırasıyla kişisel tecrübe ve sosyal tecrübe olarak kullanılırlar. *K-Eniyi* ve *S-Eniyi* çözümlerinin amaç fonksiyonundaki görüntülerini sırasıyla *K-Değeri* ve *S-Değeri* ile ifade edeceğiz.

Bir parçacığın hızı, eski hız ile parçacığın kişisel ve sosyal tecrübesinden etkilenecek güncellenir. Parçacığın pozisyonu ise eski pozisyonu ve hızından etkilenecek hesaplanır. PSO'da kullanılan formüller Şekil 3.3'te gösterilmiştir. Hızın güncellenmesi formülünde kullanılan w eski hızdan, C_1 kişisel tecrübeden, C_2 ise sosyal tecrübeden ne kadar etkileneceğini belirleyen katsayılarıdır. p_1 ve p_2 ise erken çökelmeyi engellemek ve aramaya rastsallık katmak için kullanılan değişkendir.

Hızın Güncellenmesi:

$$v_i(t) = w \times v_i(t-1) + \rho_1 C_1 \times (p_i - x_i(t-1)) + \rho_2 C_2 \times (p_g - x_i(t-1))$$

Pozisyonun Güncellenmesi:

$$x_i(t) = x_i(t-1) + v_i(t)$$

w : Eylemsizlik katsayısı

C_1 : Kişisel öğrenme faktörü

C_2 : Sosyal öğrenme faktörü

p_1, p_2 : $[0,1]$ arasında rastgele bir sayı

p_i : Bireysel en iyi pozisyon

p_g : Evrensel en iyi pozisyon

Şekil 3.3. PSO'da Kullanılan Formüller

PSO'nun çalışma şablonu Şekil 3.4'de verilmiştir. Bu şablonda görüldüğü gibi ilk önce sürüdeki bütün parçacıklar ilklendirilir. Bu aşamada her bir parçacığın ilk pozisyon ve hız değişkenleri atanır. Bu işlem rastsal olarak ya da açgözlü bir algoritma yardımı ile yapılabilir. Daha sonra, sonlandırma koşulu sağlanana kadar parçacıklar belirtilen formüller ile pozisyon ve hız değişkenlerini güncellerler. Her turda ayrıca her parçacık kişisel ve sosyal tecrübesini günceller.

Algoritma : Parçacık Sürü Optimizasyonu Şablonu

Girdi : Bir optimizasyon problemi

Çıktı : Optimal ya da optimale yakın bir çözüm

1. Bütün sürüyü ilklendir
2. **do**
3. bütün parçacıklar için
4. hızı güncelle
5. pozisyonu güncelle
6. kişisel tecrübeyi güncelle
7. sosyal tecrübeleri güncelle
8. **while** sonlandırma koşulu karşılanana kadar

Şekil 3.4. Parçacık Sürü Optimizasyonu Şablonu

PSO, sürekli optimizasyon problemleri için geliştirilmiş olduğu için hız ve pozisyon bileşenleri ile bunlar üzerinde tanımlanan operasyonlar reel sayılar için uygundur. Bu nedenle PSO'yu ayrık bir probleme direkt olarak uygulamak mümkün değildir. PSO'yu ayrık bir probleme uygulayabilmek için ayrık problem üzerinde PSO bileşenlerini anlamlandıracak yeni bir model geliştirmek gerekmektedir. Aşağıda literatürdeki bazı ayrık PSO modelleri kısaca anlatılmıştır.

Gezgin satıcı problemi (Traveling Salesman Problem) için kullanılmak üzere tasarlanan bir PSO modelinde bir parçacığın konumu, düğüm kümesinin bir permütasyonu; hızı ise bu permütasyondaki elemanları yer değiştirme operasyonu olarak tanımlanmıştır [Wang et al, 2005]. Bu modele göre örneğin beş düğümlü bir çizgedeki örnek bir pozisyon ve örnek bir hız sırasıyla şu şekilde gösterilir: $P = (1,5,4,3,2)$, $V = T(5,2)$. Bu pozisyon ve hızın toplamı bize yeni pozisyonu verir. $P + V = (1,2,5,4,3)$. Bu işlemde, P 'nin 5. konumda bulunan elemanı alınarak 2. konuma taşınmıştır. Hızın bir skaler ile çarpımı da benzer şekilde tanımlanmıştır.

Genelleştirilmiş gezgin satıcı problemi (Generalized Traveling Salesman Problem) için kullanılmak üzere tasarlanan bir diğer PSO modelinde ise, tanımlanan yıkma (destruction) ve inşaa etme (construction) işlemleri ile eski çözümün önce bozulması, daha sonra sırasıyla kişisel tecrübe ve sosyal tecrübeden faydalanarak tamir edilmesi esas alınmıştır [Taşgetiren et al, 2007]. Modelde pozisyon şu formül kullanılarak güncellenir :

$$P^{i+1} = c_2 \oplus CR_2(c_1 \oplus CR_1(w \oplus DC_1(P^i), K^i), S^i)$$

Bu formüldeki ilk işlem $w \oplus DC_1(P_j^i)$ 'dir. Buradaki DC_1 , yıkma ve inşaa etme işlemidir. Bu işlem eylemsizlik değişkenini belirten w olasılığıyla yapılır. İkinci işlem CR_1 (çaprazlama) işlemidir. Kişisel en iyi (K_j^i) ile birinci işlemin sonucu c_1 olasılıkla çaprazlanır. Son olarak ikinci işlemin sonucu c_2 olasılıkla evrensel en iyi ile çaprazlanır (CR_2).

Özellik seçimi problemi (Feature Selection Problem) için geliştirilen bir PSO modelinde ise bir parçacığın konumu, sahip olduğu özelliklerin kümesi; hızı ise her

özelliğın seçilme olasılıklarından oluşan özellik sayısı büyüklüğünde bir vektördür [Correa et al, 2006]. Bu modelde hızın güncellenmesi aşağıdaki formül ile sağlanır :

$$V_{i+1} = wV_i + c_1(K_i - P_i) + c_2(S_i - P_i)$$

Bu formülde, w eylemsizlik, c_1 ve c_2 , 0 ile 1 arasında rastsal sayılardır.

4. GELİŞTİRİLEN AYRIK PSO MODELİ

Bu bölümde, alt küme problemleri için geliştirdiğimiz PSO modeli olan PSO-A modelinin detayları anlatılacaktır. Daha önceden belirtildiği gibi PSO-A ismindeki “A” harfi alt kümeyi temsil etmektedir. PSO-A ikinci bölümde anlatılan büyüme ve küçülme amaçlayan alt küme tabanlı bütün ayrık optimizasyon problemlerinin çözümü için kullanılabilecek genel bir modeldir.

Büyüme veya küçülme amaçlayan bir alt küme problemi olmak geliştirilen modelin çeşitli aşamalarında gerekli olan bir kısıtlamadır. Bu aşamalar ilerleyen bölümlerde anlatılmıştır.

PSO kavramları olan *parçacık* ve *sürü* PSO-A’da da aynı şekilde kullanılmaktadır. PSO-A da aynı PSO gibi yinelemeli bir modeldir.

PSO-A’daki *pozisyon* kavramı, yine PSO’da olduğu gibi parçacığın arama uzayındaki konumu yani çözümü anlamına gelir. PSO-A, alt küme problemlerine uygulanan bir model olduğu için, PSO-A’daki *pozisyon*, uygulandığı problemin evrensel kümesinin bir alt kümesine karşılık gelir. *GüncelDeğer* de yine PSO’daki tanımın aynı haliyle kullanılmıştır.

PSO-A’da olursuz çözümlerin oluşmasına hiçbir zaman izin verilmez. İlkendirme aşamasında bütün parçacıkların pozisyonlarına olurlu bir çözüm atanır. Daha sonraki aşamalarda da pozisyonların sürekli olarak olurlu kalması sağlanır.

PSO-A’da, PSO’daki *hız* kavramı belirgin bir bileşen olarak ortaya çıkmaz. Bir pozisyonun güncellenmesi işlemi sırasında hız katmanı kullanılmaz. Pozisyon, direkt olarak eski pozisyon ile kişisel ve sosyal tecrübelerden yararlanarak güncellenir. Pozisyonun güncellenmesi formülü Şekil 4.1.’de gösterilmiştir. Bu formülde P pozisyonu, K kişisel en iyiyi, S sosyal en iyiyi, f_λ ve ρ sırasıyla detayları ilerde anlatılacak olan bozma ve onarma işlemlerini simgeleyen fonksiyonlardır.

$$P_{i+1} = f_{\lambda}(P_i) \oplus p(K_i, S_i)$$

Şekil 4.1. PSO-A'da Pozisyonun Güncellenmesini Gösteren Formül

Kişisel tecrübe, sosyal tecrübe, K-Eniyi, S-Eniyi, K-Değeri, S-Değeri, komşuluk topolojisi kavramları PSO'daki halleriyle kullanılmıştır.

PSO-A'da herhangi bir komşuluk topolojisi kullanılabilir. Yapılan deneylerde daha iyi sonuçlar verdiği görüldüğü için geliştirilen algoritmalarda halka topolojisi tercih edilmiştir.

Bu bölümün geri kalanında önce modelde geçen yeni kavramlar tanımlanacak ardından modelin genel çalışma prensipleri detaylı olarak anlatılacak, son olarak da modelin çalışma zamanı analizi yapılacaktır. Bu adımlar sırasında büyüme amaçlayan problemler dikkate alınmıştır. Küçülme amaçlayan problemlerde anlatımlar ufak farklılıklar gösterir ve bu farklılıklar parantez içerisinde gösterilmiştir.

4.1. Tanımlar

PSO-A'da bir parçacığın pozisyonunun güncellenmesi aşamasında iki yardımcı kümeden faydalanılır. Bu kümelerin tanımları aşağıda verilmiştir.

Tanım 4.1. Geçici Pozisyon :

Bir parçacığın geçici pozisyonları, parçacığın bir önceki turdaki pozisyonundan bazı noktalar çıkarılmış (pozisyonuna bazı noktalar eklenmiş) halinden başlanarak, olurlu çözüm olma şartı korunacak şekilde teker teker eklenen (çıkarılan) elemanlarla genişleyen (daralan) ve sonunda yeni bir pozisyona dönüşen alt kümelerin tamamına verilen isimdir.

Tanım 4.2. Aday Noktalar :

Bir parçacığın o andaki geçici pozisyonuna eklendiğinde (çıkarıldığında) olurlu çözüm olma şartını bozmayan noktaların oluşturduğu kümeye *aday noktalar* adı verilir.

4.2. Modelin Detayları

PSO-A modelinde, temel olarak PSO'da olduğu gibi sürüdeki her parçacık kişisel ve sosyal tecrübelerinden yararlanarak her turda pozisyonunu güncelleyerek daha iyi çözümlere ulaşmaya çalışır. Sonlanma koşulu sağlanana kadar turlara devam edilir.

4.2.1. İlkendirme

PSO-A modeli uygulanarak geliştirilen bir algoritmanın başlangıcında her parçacığın pozisyon, K-Enİyi ve S-Enİyi bileşenleri olurlu bir çözüm olarak atanır. Başlangıç çözümü olarak açgözlü bir algoritma ile bulunan bir çözüm kullanılabileceği gibi boş küme (evrensel küme) de kullanılabilir. Geliştirilen algoritmalarda erken çökelmeyi engellemek için ikinci yöntem tercih edilmiştir. Büyüme (küçülme) amaçlayan alt küme problemi tanımı gereği boş küme (evrensel küme) olurlu bir çözümdür. GüncelDeğer, K-Değeri ve S-Değeri, bu değerlere göre hesaplanarak ilklendirilir.

4.2.2. Her Turda Yapılan İşlemler

Her parçacık, her turda pozisyonunu , bir önceki turdaki pozisyon, K-Enİyi ve S-Enİyi bileşenlerinden etkilenecek güncellerler. Bu işlem iki aşamada gerçekleştirilir.

İlk olarak bir önceki turdaki pozisyona *bozma* işlemi yapılır. Bozma işlemi sırasında pozisyonundaki noktalardan bazıları kümeden çıkartılır (pozisyonunda olmayan noktalardan bazıları kümeye eklenir). Bunun için *bozma katsayısı* adını vereceğimiz

ve λ ile göstereceğimiz bir parametre kullanılır. 0 ile 1 arasında sabit bir değer olan λ , pozisyondaki her noktanın kümeden çıkarılma (pozisyonun dışındaki her noktanın kümeye eklenme) olasılığını belirler. Buna göre λ , 0 olursa pozisyon aynen korunur, 1 olursa pozisyondaki bütün noktalar kümeden çıkarılır (pozisyonun dışındaki her nokta kümeye eklenir).

Bozma aşaması tamamlandıktan sonra *onarma* işlemine geçilir. Önce oluşan geçici pozisyon için aday noktalar hesaplanır. Daha sonra aday noktaların her birinin geçici pozisyona eklenme (geçici pozisyondan çıkarılma) olasılıkları hesaplanır. Bu olasılıklar hesaplanırken noktaların *K-Enİyi* ve *S-Enİyi* kümelerinde bulunup bulunmaması dikkate alınır. Bir noktanın bu iki kümenin herhangi birinde ya da ikisinde de bulunması seçilme olasılığını etkiler. Noktaların bu kümelerde bulunmasının seçilme olasılıklarını nasıl etkilediği Tablo 4.1.'de gösterilmiştir.

Tablo 4.1. Noktaların Seçilme Oranları

<i>Durum</i>	<i>Seçim Oranı</i>
$v \in K, v \in S$	p_{KS}
$v \in K, v \notin S$	p_K
$v \notin K, v \in S$	p_S
$v \notin K, v \notin S$	p_0

Buradaki p_{KS} , p_K , p_S ve p_0 modelin noktaların seçilme oranlarını belirleyen parametreleridir. Aday noktalar, bu parametrelere göre belirlenen olasılıklar yardımı ile seçilir. Geliştirilen algoritmalarda, farklılaşma kriterini artırmak için bu olasılıklara göre rulet tekeri seçimi yapılması tercih edilmiştir. Seçilen nokta geçici pozisyona eklenir (geçici pozisyondan çıkarılır) ve aday noktalar tekrar güncellenir. Bu işleme aday noktalar kümesinde eleman kalmayana kadar devam edilir. Son durumdaki geçici pozisyon, pozisyon olarak atanır.

p_{KS} , p_K , p_S ve p_0 parametreleri algoritma boyunca değişmeyen sabit değerler olabileceği gibi dinamik de olabilirler. Geliştirilen algoritmalarda bu parametrelerden ilk üçü sabit tutulmuş, dördüncü parametre olan p_0 ise her turdan sonra “ α ” ile göstereceğimiz ve “ p_0 değişim katsayısı” adını vereceğimiz 1’den küçük (1’den

büyük) bir değer ile çarpılarak zaman içerisinde değiştirilmiştir. Böylelikle, zaman içerisinde p_0 değerini küçülterek (büyütür), farklılaştırmanın azalması ve çökmenin gerçekleşmesi amaçlanmıştır.

Büyüme (küçülme) amaçlayan alt küme problemi tanımı gereği işlemi bozma ve onarma işlemleri sırasında oluşan bütün geçici pozisyonlar ve dolayısıyla oluşan pozisyon olurlu çözümlerdir. Aynı tanım gereği, oluşan pozisyonun amaç fonksiyonu değeri, pozisyonun oluşumu sırasında ortaya çıkan bütün geçici pozisyonlardan daha “iyi”dir.

Daha sonra kişisel tecrübenin güncellenmesi işlemi yapılır. Bunun için öncelikle GüncelDeğer hesaplanır. GüncelDeğer, parçacığın K-Değeri’nden daha iyi ise K-Enİyi ve K-Değeri bileşenleri güncellenir.

Bütün parçacıkların turları tamamlandıktan sonra sosyal tecrübelerin güncellenmesi işlemine geçilir. Bu işlemde, her parçacığın S-Değeri, komşularının K-Değeri ile karşılaştırılır. En iyi K-Değeri’ne sahip komşunun K-Değeri eğer parçacığın S-Değeri’nden daha iyi ise parçacığın S-Enİyi ve S-Değeri bileşenleri güncellenir.

PSO-A’nın genel şablonu Şekil 4.2.’de gösterilmiştir.

Algoritma : PSO-A Şablonu

Girdi : Alt küme tabanlı bir optimizasyon problemi

Çıktı : Optimal ya da optimale yakın bir çözüm

1. her bir parçacığı ilklendir

2. **do**

3. her parçacık için

4. Pozisyonu boz

5. Pozisyonu onar

6. Kişisel tecrübeyi güncelle

7. bütün parçacıklar için sosyal tecrübeyi güncelle

8. **while** (sonlandırma koşulu karşılanana kadar)

Şekil 4.2. PSO-A Şablonu

4.2.3. Durdurma Kriteri

Belli bir tur sayısı tamamlandığında ya da belli bir tur süresince gelişme sağlanmadığında çalışma sonlandırılır. O ana kadar ulaşılan en iyi GüncelDeğer ve bu değere sahip pozisyon sonuç olarak döndürülür. Geliştirilen algoritmalarda belli bir tur sayısı (MTS: Maksimum tur sayısı) tamamlanması durdurma kriteri olarak kullanılmıştır. Ayrıca geliştirilen algoritmaları optimal çözümü bilinen karşılaştırma problemleri üzerinde test ederken, optimal çözüm bulunduğu anda algoritmanın sonlandırılması tercih edilmiştir.

4.2.4. Modelin Parametreleri

PSO-A modelinin parametreleri Tablo 4.2.'de listelenmiştir. Bu parametrelere atanan değerler bu model uygulanarak geliştirilen bir algoritmanın çözüm kalitesini ve çalışma zamanını doğrudan etkilemektedir. Bu yüzden amacımıza yönelik olarak bu parametrelere uygun değerler atamak gerekmektedir.

Tablo 4.2. Modelin Parametreleri

<i>Parametre</i>	<i>Anlamı</i>
SB	Sürü Büyüklüğü
MTS	Maksimum Tur Sayısı
p_{KS}	K -Enİyi ve S -Enİyi'de bulunan noktanın seçilme oranı
p_K	Sadece K -Enİyi'de bulunan noktanın seçilme oranı
p_S	Sadece S -Enİyi'de bulunan noktanın seçilme oranı
p_0	K -Enİyi veya S -Enİyi'de bulunmayan noktanın seçilme oranı
λ	Pozisyonu bozma katsayısı
α	p_0 değişim katsayısı

4.3. Zaman Karmaşıklık Analizi

PSO-A yinelemeli bir modeldir. PSO-A modeli kullanılarak geliştirilen bir algoritmanın çalışma zamanı tur sayısı ile doğrusal olarak artar. Sonlanma koşulu, en iyi çözümün bulunması ya da maksimum tur sayısına ulaşmak olduğu için, en kötü durumda maksimum tur sayısı kadar tur sonunda çalışma sonlanacaktır.

Her turda yapılacak işlemler, her parçacığın pozisyonunu bozması, onarması, kişisel tecrübesini güncellemesi ile bütün parçacıkların sosyal tecrübelerini güncellemesidir. Buradaki işlemlerin ilk üçünün çalışma zamanı doğal olarak SB (sürü büyüklüğü) ile doğrusal olarak artar. Bir parçacığın pozisyonunu bozması sadece pozisyonadaki nokta sayısı ile ilişkilidir. Pozisyonadaki nokta sayısının üst sınırı evrensel kümedeki nokta sayısı olduğu için bozma işlemi $O(|U|)$ 'dir. Bozma işleminden sonra aday noktalar kümesinin tekrar hesaplanması gerekmektedir. Bu işlemin karmaşıklığı modelin uygulandığı probleme göre farklılık gösterir. Bu yüzden bu aşamada, bu işlemin karmaşıklığını ϕ ile göstereceğiz. Bir parçacığın pozisyonunu onarması aday noktalar kümesi boşalana kadar, aday noktalardan nokta seçerek geçici pozisyona ekleme (geçici pozisyondan çıkarma) işlemidir. Bir nokta seçimi işlemi $O(|U|)$ karmaşıklıktadır. Ayrıca her nokta ekleme işleminden sonra aday noktaların güncellenmesi gerekmektedir. Bu işlemin karmaşıklığı da üzerinde çalışılan probleme özgü olduğu için işlemin karmaşıklığını genel olarak γ ile göstereceğiz. Dolayısıyla aday noktalardaki bir noktanın geçici pozisyona ekleme (geçici pozisyondan çıkarma) işleminin karmaşıklığı $O(|U|) + \gamma$ 'dir. Aday noktalar kümesinde en fazla $O(|U|)$ nokta olacağı için, bir parçacığın pozisyonunu onarması işleminin karmaşıklığı $O(|U| * (|U| + \gamma))$ 'dir. Bir parçacığın kişisel tecrübesini güncellemesi sabit zamanda yapılır. Bir parçacığın bir turda yaptığı bütün işlemlerin toplam karmaşıklığı $O(|U| + \phi + |U| * (|U| + \gamma))$ olarak bulunur. Bu ifade sadeleştirilirse $O(\phi + |U| * (|U| + \gamma))$ elde edilir. Bir turda bütün parçacıklar için yapılan işlemlerin karmaşıklığı $O(SB * (\phi + |U| * (|U| + \gamma)))$ olarak bulunur. Bütün parçacıkların sosyal tecrübelerini güncellemesi işleminin karmaşıklığı komşuluk topolojisine göre farklılık gösterir ve en kötü durumda $O(SB * SB)$ 'dir. Buna göre bir turda yapılan bütün işlemlerin karmaşıklığı

$O(SB * (\phi + |U| * (|U| + \gamma)) + (SB * SB))$ olarak bulunur. Sadeleştirirsek $O(SB * ((\phi + |U| * (|U| + \gamma)) + SB))$ elde ederiz.

Toplamda en fazla MTS (maksimum tur sayısı) tur yapılacağı için PSO-A modeli kullanılarak geliştirilen bir algoritmanın toplam karmaşıklığı $O(MTS * SB * ((\phi + |U| * (|U| + \gamma)) + SB))$ olarak bulunur.

Gerçekleştirilen algoritmalarda komşuluk topolojisi olarak halka topolojisi kullanıldığı için sosyal tecrübelerin güncellenmesi işleminin karmaşıklığı $O(SB)$, bir turda yapılan bütün işlemlerin karmaşıklığı $O(SB * (\phi + |U| * (|U| + \gamma)) + SB)$ olur. Bu ifadeyi sadeleştirirsek $O(SB * (\phi + |U| * (|U| + \gamma)))$ elde ederiz. Buna göre modelin toplam karmaşıklığı $O(MTS * SB * (\phi + |U| * (|U| + \gamma)))$ olur.

5. MODELİN PROBLEMLERE UYGULANMASI

PSO-A modeli, büyüme ve küçülme amaçlayan alt küme problemlerine uygulanabilecek genel bir modeldir ve uygulandığı probleme özgü olarak çok az farklılık gösterir. Modelin farklı problemlere uygulanışı sırasındaki en temel farklılıklar bozma işlemi tamamlandıktan sonra aday noktalar kümesinin hesaplanması ve onarma işlemi sırasındaki her adımda aday noktalar kümesinin güncellenmesi işlemleridir. Bu işlemlerin karmaşıklıklarını modelin karmaşıklık analizi yapılırken ϕ ve γ simgeleri ile göstermiştik. Bu bölümde modelin, MKP, ÇSP ve KKP problemlerine uygulanması sırasında bu işlemlerin nasıl yapıldığı anlatılacak ve zaman karmaşıklık analizleri yapılacaktır.

5.1. Modelin MKP'ye Uygulanması

MKP'de bozma işleminden sonra aday noktalar kümesi hesaplanırken geçici çözümün dışında kalan bütün noktaların teker teker aday noktalar kümesine dahil olup olmadığına bakılır. MKP'de geçici çözüm dışındaki bir noktanın aday noktalar kümesine dahil olabilmesi için, geçici çözüm içerisindeki her nokta ile ilişkili olması gerekmektedir. Bu yüzden, bir nokta için bu işlemin karmaşıklığı $O(|U|)$ olur. Dolayısıyla geçici çözüm dışındaki bütün noktalar için bu işlemin karmaşıklığı $O(|U| * |U|)$ olarak bulunur.

Onarma işlemi sırasında geçici çözüme bir nokta eklendikten sonra aday noktalar kümesindeki bazı noktaların aday olma özelliklerini kaybederler. Aday olma özelliğini kaybeden bu noktalar, MKP'de geçici çözüme eklenen son nokta ile ilişkisi olmayan noktalardır. Bu nokta ile ilişkili olan noktalar aday olma özelliklerini korurlar. Buna göre MKP'de aday noktalar kümesini güncellerken aday noktalar kümesindeki noktaların geçici çözüme son eklenen nokta ile ilişkili olup olmadığına bakılması gerekir. Bu işlemin karmaşıklığı $O(|U|)$ olur.

Dolayısıyla PSO-A'nın MKP'ne uygulanması için geliştirilen algoritmanın karmaşıklığı $O(MTS * SB * ((|U| * |U|) + |U| * (|U| + |U|)))$ olarak bulunur. Bu ifadeyi sadeleştirirsek $O(MTS * SB * |U|^2)$ elde ederiz.

5.2. Modelin ÇSP'ye Uygulanması

ÇSP'de bozma işleminden sonra aday noktalar kümesi hesaplanırken geçici çözümün dışında kalan bütün noktaların teker teker aday noktalar kümesine dahil olup olmadığına bakılır. ÇSP'de geçici çözüm dışındaki bir noktanın aday noktalar kümesine dahil olabilmesi için; nesnenin, çantaya eklenebilir bir durumda olması gerekir. Çantanın ne kadar dolu olduğunu bilirse bu işlemi $O(m)$ zamanda yapabiliriz (m , boyut sayısını gösteriyor). Dolayısıyla geçici çözüm dışındaki bütün noktalar için bu işlemin karmaşıklığı $O(|U| * m)$ olarak bulunur.

Onarma işlemi sırasında geçici çözüme bir nokta eklendikten sonra aday noktalar kümesindeki bazı noktaların aday olma özellikleri kaybolur. Aday olma özelliğini kaybeden bu noktalar, son eklenen nokta ile birlikte çantaya eklenemez hale gelen noktalardır. Aday noktaların hesaplanması işlemi gibi bu işlem de $O(m)$ zamanda yapılabilir. Aday noktaların hepsi için bu işlem tekrarlanacağı için bu işlemin toplam karmaşıklığı da $O(|U| * m)$ olur.

Dolayısıyla PSO-A'nın ÇSP'ye uygulanması için geliştirilen algoritmanın karmaşıklığı $O(MTS * SB * (|U| * m + |U| * (|U| + |U| * m)))$ olarak bulunur. Bu ifadeyi sadeleştirirsek $O(MTS * SB * m * |U|^2)$ elde ederiz.

5.3. Modelin KKP'ye Uygulanması

KKP'de bozma işleminden sonra aday noktalar kümesi hesaplanırken geçici çözümün içerisindeki bütün noktaların teker teker aday noktalar kümesine dahil olup olmadığına bakılır. KKP'de geçici çözüm içerisindeki bir noktanın aday noktalar kümesine dahil olabilmesi için; noktanın, ilişkili olduğu bütün noktaların geçici çözümün içerisinde olması gerekir. Bu kontrolün yapılması işleminin karmaşıklığı

bir nokta için $O(|U|)$ 'dur. Dolayısıyla geçici çözüm içerisindeki bütün noktalar için bu işlemin karmaşıklığı $O(|U| * |U|)$ olarak bulunur.

Onarma işlemi sırasında geçici çözümünden bir nokta çıkarıldıktan sonra aday noktalar kümesindeki bazı noktaların aday olma özellikleri kaybolur. Aday olma özelliğini kaybeden bu noktalar, son çıkarılan nokta ile ilişkisi olmayan noktalardır. Bu işlem bir nokta için sabit zamanda yapılır. Aday noktaların hepsi için bu işlem tekrarlanacağı için bu işlemin toplam karmaşıklığı $O(|U|)$ olur.

Dolayısıyla PSO-A'nın KKP'ye uygulanması için geliştirilen algoritmanın karmaşıklığı $O(MTS * SB * ((|U| * |U|) + |U| * (|U| + |U|)))$ olarak bulunur. Bu ifadeyi sadeleştirirsek $O(MTS * SB * |U|^2)$ elde ederiz.

6. DENEYLER

PSO-A'nın uygulandığı maksimum klik problemi (MKP), çok boyutlu sırt çantası problemi (ÇSP) ve kenar kapsama problemi (KKP) ile yapılan deneyler bu bölümde anlatılmıştır. Bu deneylerle PSO-A'nın bu problemler üzerindeki çözüm kaliteleri, çalışma zamanları ve parametre seçiminin etkileri incelenmiştir.

Algoritmalar için uygun parametre değerlerini bulmak için çeşitli testler ile parametre analizleri yapılmıştır. Bunun için iki tipte testler yapılmıştır. Birinci tip testlerde, diğer bütün parametreler sabit tutularak bir parametre değerinin değişimi ile çözüm kalitesinin değişimi incelenmiştir. Bu testler yapılırken sabit tutulan parametrelerin ilk değerleri Tablo 6.1.'de gösterilmiştir. Bu değerler daha önceden yapılan testler sonucunda anlamlı sonuçlar vereceği gözlenen değerlerdir. Bu sırada SB (sürü boyutu) parametresini test ederken orantılı bir test olması için MTS (maksimum tur sayısı) parametresi sabit tutulmamış $200000/SB$ olarak seçilmiştir. pS parametresi, onu oransal etkileyecek diğer bütün parametrelerin analizi yapıldığı; $p0$ parametresi de α parametresinden direk etkilendiği için bu parametrelerin ayrıca analizleri yapılmamıştır. Buna göre birinci tip testlerde SB, pKS , pK , λ ve α parametrelerinin değişiminin algoritmanın çözüm kalitesine etkisi araştırılmıştır. Bunun için karşılaştırma problemlerinden seçilen temsili bir problem üzerinde 25 kez çalıştırılmış ve bulunan çözümlerin ortalaması alınmıştır.

Tablo 6.1. Parametrelerin İlk Değerleri

<i>Parametre</i>	<i>Değer</i>
SB	25
MTS	5000
pKS	1.0
pK	1.0
pS	1.0
$p0$ (Başlangıç değeri)	1.0
λ	0.5
α	0.999 (1.001)

İkinci tip testlerde ise parametrelerden biri değiştirilerek ve diğer parametre değerleri yine Tablo 6.1.'deki gibi tutularak algoritma 50 kez çalıştırılmıştır. Birinci tipteki testlerden farklı olarak bu testlerde sadece algoritma sonlandığındaki en iyi değer değil, algoritma çalışırken her turda bulduğu en iyi değerlerin ortalaması alınmıştır. Böylelikle değişen parametrenin algoritmanın çalışma karakterini nasıl etkilediğini daha net bir şekilde görmek mümkün olmuştur.

Bu testlerin dışında bir de PSO-A modelinin en önemli değişkenlerinden olan ve noktaların seçilme olasılıklarını belirleyen dolayısıyla modelin temelini oluşturan pKS , pK , pS ve $p0$ parametreleri için bir test yapılmıştır. Bu teste seçim oranları testi adını vereceğiz. Bu testte, algoritmanın pKS , pK , pS ve $p0$ parametrelerinin hepsinin 1, hepsinin 0 ve bu parametrelerden sadece biri 0, diğerleri bir olarak seçildiği durumlarda nasıl davranış gösterdiği incelenmiştir. Bütün parametrelerin 0 seçildiği test, noktanın rastsal olarak seçilmesi anlamına gelmektedir.

Yapılan testlerin sonuçları, tablo ve grafikler ile sunulmuş ve bu sonuçların anlamı hakkında yorumlarda bulunulmuştur. Daha sonra bu testler sonucunda seçilen parametre değerleri kullanılarak algoritma bütün karşılaştırma problemleri üzerinde 20'şer kez çalıştırılmış ve sonuçlar tablolar ile sunulmuştur.

Algoritmalar C++ programlama dilinde gerçekleştirilmiştir. Kullanılan bilgisayarın işlemcisi 2.40 Ghz. Core 2 Quad, hafızası 3GB, işletim sistemi de Windows XP'dir.

Bu bölümde ayrıca bu problemler üzerinde uygulanan ve performansları PSO-A ile karşılaştırılan diğer metasezgiseller hakkında bilgi verilmiştir.

6.1. MKP ile Yapılan Deneyler

PSO-A modeli MKP üzerinde uygulanarak ikinci DIMACS karşılaştırmalı değerlendirme çizgeleri olan 36 çizge ile [Johnson and Trick, 1996] test edilmiş ve sonuçları üç sürü tabanlı metasezgisel ile karşılaştırılmıştır. Aşağıda bu metasezgiseller ile geliştirilen algoritmalar hakkında kısaca bilgi verilmiştir.

GLS [Marchiori, 2002] : Bu çalışmada klik problemini çözmek için genetik algoritma bir yerel arama ile desteklenerek kullanılmıştır. Bu algoritmada bütün bireyler bir maksimal kliği ifade eder. Her adımda popülasyondaki iki tane birey ebeveyn olarak seçilir, bunlar birleştirilerek iki tane çocuk birey oluşturulur, yeni oluşan kliklere yerel arama uygulanarak maksimal klik haline getirilir, ve son olarak iki ebeveyn ve iki çocuk arasından en iyi ikisi popülasyona eklenir. GLS, parametreleri değiştirilerek üç farklı algoritma olarak gerçekleştirilmiştir. Popülasyon büyüklüğünü 10, mutasyon oranını 0.1, çaprazlama oranını 0.9 ve sonlanma şartını 2000 jenerasyon yapınca genetik algoritma (GENE); popülasyon büyüklüğünü 1, mutasyon oranını 0 ve sonlanma şartını 20000 jenerasyon yapınca tek bir parçacığın arka arkaya yerel araması (ITER); popülasyon büyüklüğünü 20000 ve sonlanma şartını 0 jenerasyon yapınca çok parçacığın tek bir yerel araması (MULT) ortaya çıkar. Tablo 5.3’de üç versiyondan en iyi değeri veren versiyon kullanılmıştır ve (G), (I) ve (M) harfi ile hangi versiyon olduğu belirtilmiştir.

ANT [Fenet and Solnon, 2004]: Bu algoritmada her bir karınca her turda yeni bir klik oluşturur. Her karıncanın çözümü iklendirme aşamasında boş küme olarak atanır. İlk düğüm rastsal seçilir. Diğer düğümler, eklenebilir düğümler arasından feromon iziyle orantılı olarak rulet seçimi ile seçilir. Aday noktalar kümesi, çözüm kümesine eklendiğinde klik olmayı bozmayacak elemanların oluşturduğu kümedir. Bu kümedeki bir düğümün çözüme eklenme olasılığı bu düğümün o anda çözümde bulunan bütün düğümlerle olan kenarlarının üzerindeki feromon miktarlarının toplamı ile orantılıdır. Sürekli olarak eklenebilir düğümlerin oluşturduğu kümeden düğümler eklendiği için klik olma durumu hiçbir zaman bozulmaz. Bu işleme klik olma durumunu bozmadan hiçbir düğüm eklenemez duruma gelene kadar devam edilir. Tur bittiğinde parçacıklar tarafından bulunan en iyi kliği oluşturan kenarlara kliğin kalitesi ölçüsünde feromon eklenir ve bütün kenarların feromonları belli bir oranda azaltılır (buharlaştırma). Optimum çözüm bulunana kadar ya da belli bir iterasyon sayısına ulaşana kadar bu işlem tekrarlanır.

EA/G [Zhang, 2005] : Mutasyon tabanlı evrimsel bir algoritmadır. Bu çalışmada *rehberli mutasyon* adı verilen bir algoritma geliştirilmiştir. Rehberli mutasyon, bir bireyin elemanlarını bir olasılık vektörü yardımı ile değiştirilerek yeni alt kümeler oluşturmaya yarar. Burada kullanılan olasılık vektörü sürüdeki iyi

uygunluk değerine sahip bireylerin katkısı ile oluşturulur. Mutasyon sonucunda klik olmayan alt kümeler oluşabileceği için birey daha sonra *düzeltilme* algoritması kullanılarak maksimal klik haline getirilir. Düzeltme algoritması olarak Marchiori'nin yöntemi kullanılmıştır [Marchiori, 1998]. Bu yöntemde, sürünün içerisindeki en iyi birey alınarak ona önce rehberli mutasyon daha sonra da düzeltme algoritması uygulanarak yeni bireyler oluşturulmakta ve bu bireyler sürü içerisindeki düşük uygunluk değerine sahip bireyler ile yer değiştirmektedir. Bu işlem optimal klik bulunana kadar ya da belirlenen iterasyon sayısı oluşana kadar tekrarlanmaktadır.

Bu üç yöntemden ANT, PSO-A'ya oldukça benzer. Sürekli olarak olurlu çözümler üzerinde dolaşmak, aday noktalar kümesi, eklenecek nokta seçilirken rulet seçimi yapılması iki yöntemin ortak noktalarıdır. ANT'da her turda karıncalar yeni çözüm oluştururken PSO-A'da eski çözüm belli bir oranda bozularak yeni bir çözüm inşaa edilir.

ANT ile kıyaslandığında PSO-A'nın karmaşıklığı daha azdır. İki yöntemin temel operasyonları eklenecek noktayı bulmaktır. Seçim oranları PSO-A'da bir parçacık için en fazla bir kez hesaplanır. Bunun nedeni bir parçacığın kişisel ya da sosyal tecrübeleri güncellenmeyen turlardan sonra seçim oranlarını değiştirmemesidir. Ayrıca PSO-A'da tur içerisinde her nokta eklendikten sonra oranlar değişmez. ANT'da ise her nokta eklendikten sonra noktaların seçilme olasılıkları değişir ve güncellenmesi gerekir.

Karşılaştırma Problemleri :

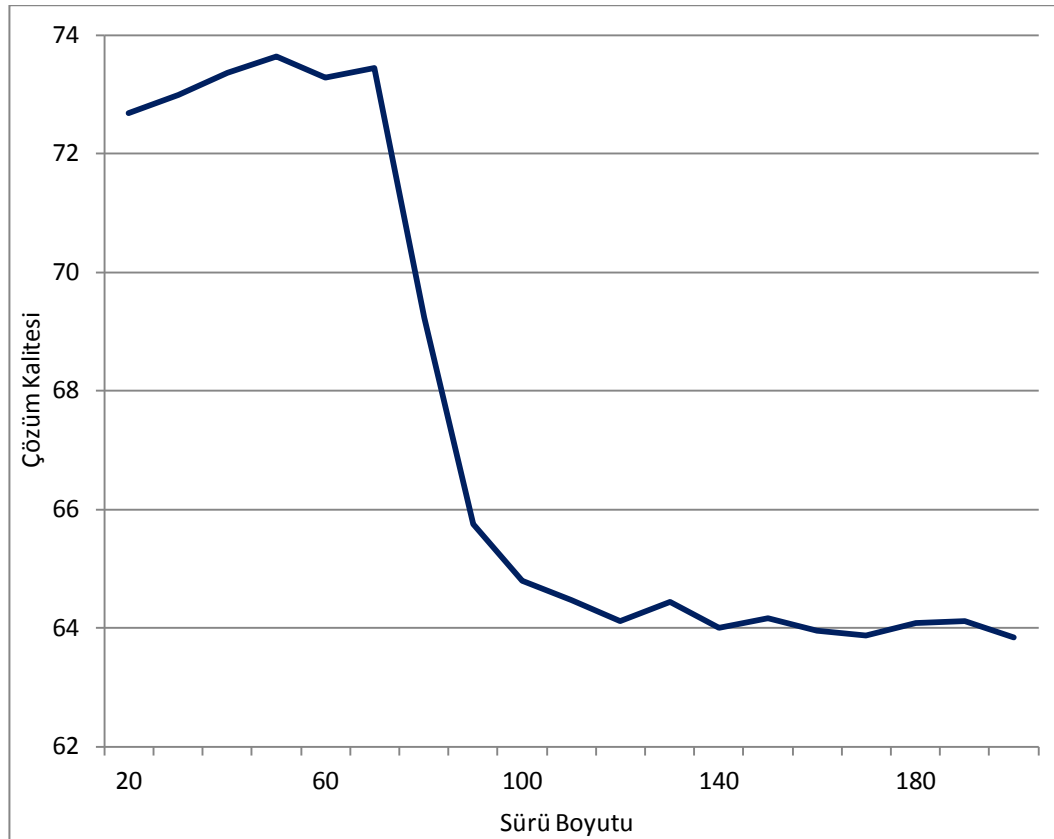
İkinci DIMACS klik problemleri içinde 8 farklı problem tipi bulunmaktadır. $C_{x,y}$ tipi problemler, x düğüme sahip, her iki düğüm arasında $0.y$ olasılıkla kenar bulunacak şekilde rastsal oluşturulan çizgelerdir. *DSJC* de rastsal çizgelerdir. *MANN* problemleri minimum küme kapsama problemleridir. *Brock* tipindeki problemler, düşük dereceli düğümlerden oluşan kliklere sahip, klikleri saklamak için özel olarak oluşturulmuş çizgelerdir. *Gen* problemleri yapay olarak oluşturulan ve içinde büyük ve bilinen bir kliğin saklandığı problemlerdir. *Hamming* problemleri kodlama

teorisinde kullanılan hamming uzaklığı ile ilgilidir. *Keller* problemleri geometri temellidir. *P-hat* problemleri büyük kliklere sahip rastsal çizgelerdir.

MKP’de temsili problem olarak ikinci DIMACS problemlerinden olan 2000 düğümlü, bilinen en büyük klik değeri 80 olan C2000.9 isimli çizge kullanılmıştır. Daha önce daha kolay problemlerin parametre analizinde kullanılması denenmiş ancak test sonuçlarını yorumlamak mümkün olmamıştır. Bu yüzden çok düğüme sahip, zor bir problem olan C2000.9 problemi parametre analizlerinde kullanılmaya başlanmıştır.

Test Sonuçları :

Sürü boyutunun (SB) çözüm kalitesine etkisi Şekil 6.1 ve Tablo 6.2.’de gösterilmiştir. Çözüm kalitesinin, sürü boyutu arttıkça belli bir noktadan sonra düşmesinin nedeni tur sayısının SB arttıkça düşmesi ve algoritmanın çökelme sağlayamadan sonlanmasıdır. Bu testler sonucunda SB, 80 olarak seçilmiştir.

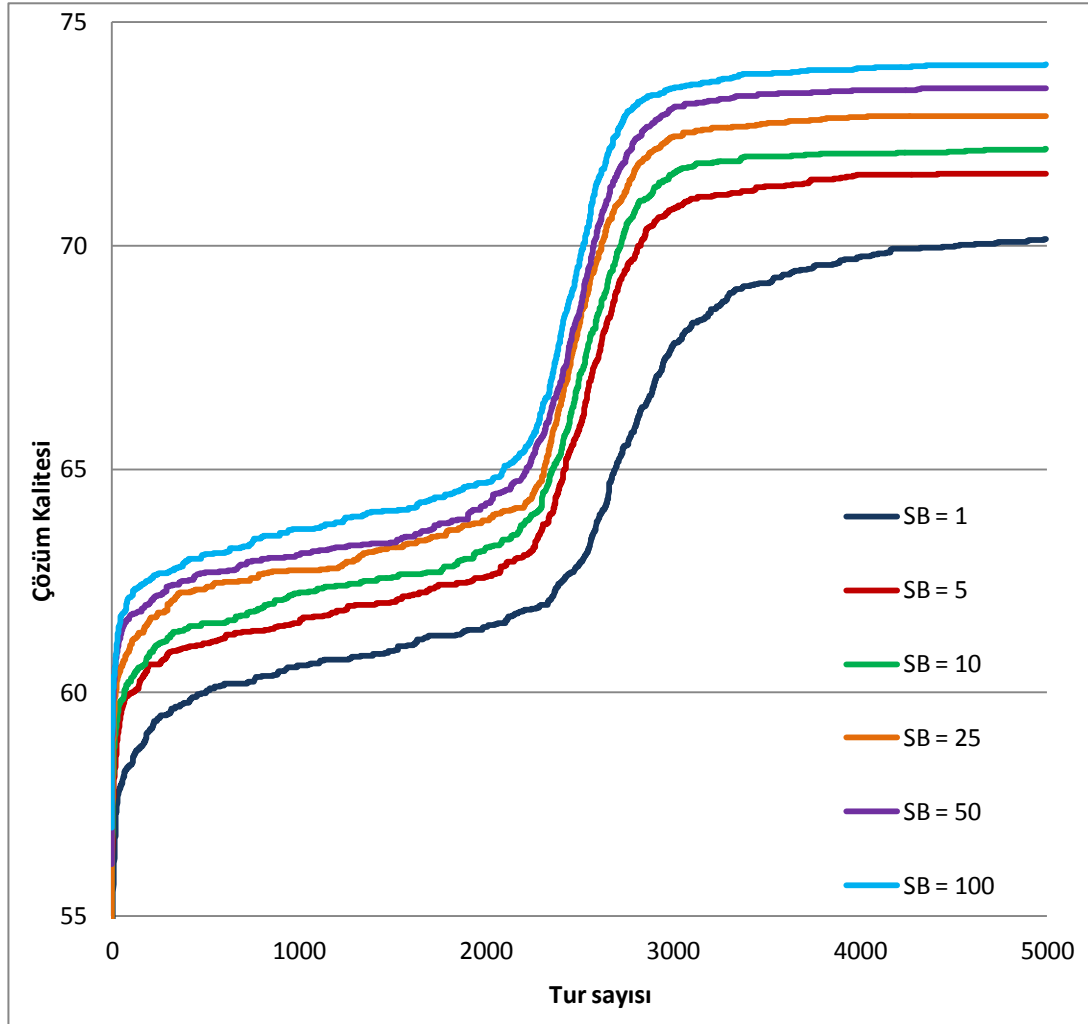


Şekil 6.1. SB - Çözüm Kalitesi Tablosu (MKP)

Tablo 6.2. SB - Çözüm Kalitesi Tablosu (MKP)

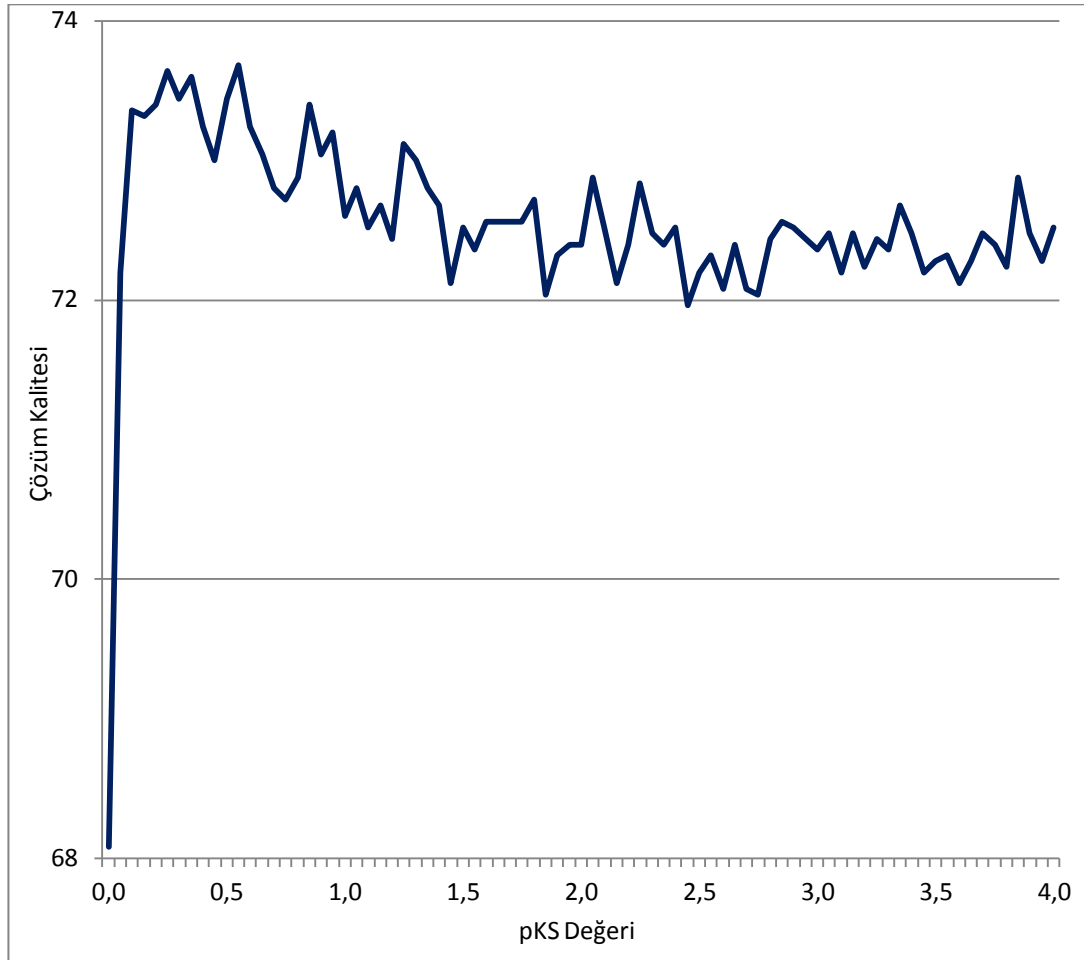
SB	En İyi	Ortalama	Std.	Süre
20	74	72,680	0,950	59,768
30	74	73,000	0,580	65,197
40	75	73,360	0,700	68,454
50	76	73,640	1,150	69,052
60	75	73,280	0,840	71,410
70	76	73,440	1,040	73,786
80	73	69,240	1,360	71,732
90	68	65,760	1,130	71,291
100	66	64,800	0,760	83,599
110	66	64,480	0,710	83,879
120	66	64,120	0,670	85,467
130	66	64,440	0,710	85,269
140	66	64,000	0,710	85,711
150	66	64,160	0,800	84,318
160	66	63,960	0,730	83,798
170	65	63,880	0,730	68,792
180	66	64,080	0,810	70,508
190	67	64,120	0,880	68,738
200	66	63,840	0,690	64,560

Farklı SB değerlerinin çözüm kalitesine etkisi Şekil 6.2.'de gösterilmiştir. Bu grafiğe göre sürü boyutunun artması küçük sayılarda çözüm kalitesinde daha büyük bir artış sağlarken, daha büyük sayılarda artış azalmaktadır. Ayrıca şekilden anlaşılacağı üzere SB'nin değişimi çökelme zamanını çok az etkilemektedir.



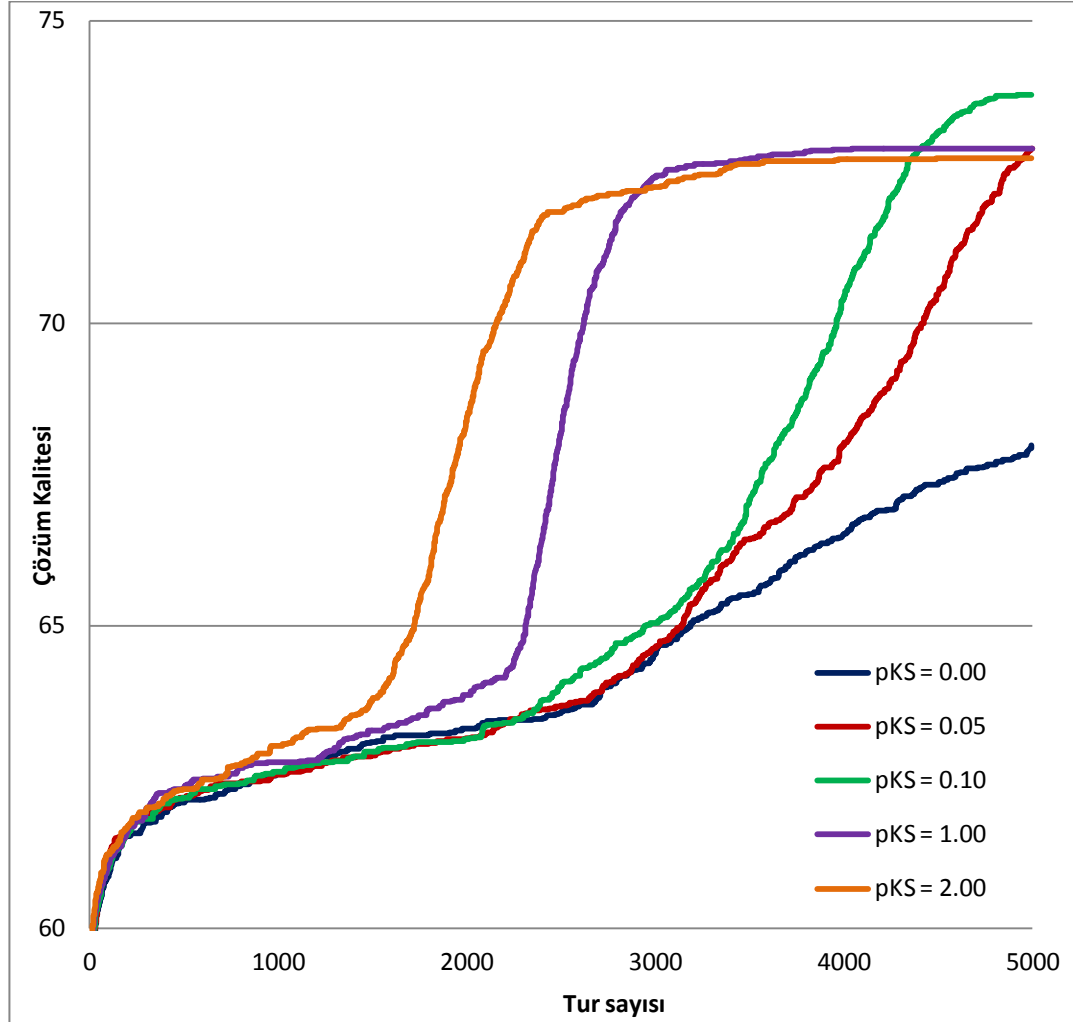
Şekil 6.2. Farklı SB Değerleri İle Yapılan Test Sonuçları (MKP)

pKS 'nin sürü çözüm kalitesine etkisi Şekil 6.3.'de gösterilmiştir. pKS , 0 değerini aldığı anda algoritmanın çökmesi çok zorlaşmakta olduğu için çözüm kalitesi çok zayıf kalmıştır. 0'dan büyük, küçük değerlerde çözüm kalitesinin yüksek olduğu görülmektedir. Bunun nedeni, pKS değeri küçük seçildiğinde çökelmenin gecikmesinin sağlanması ve farklılaşmanın artmasıdır. Algoritma daha uzun sürede daha iyi bir değer bularak çökelmiştir. Bütün problemler için testler yapılırken pKS , 0.14 olarak seçilmiştir.



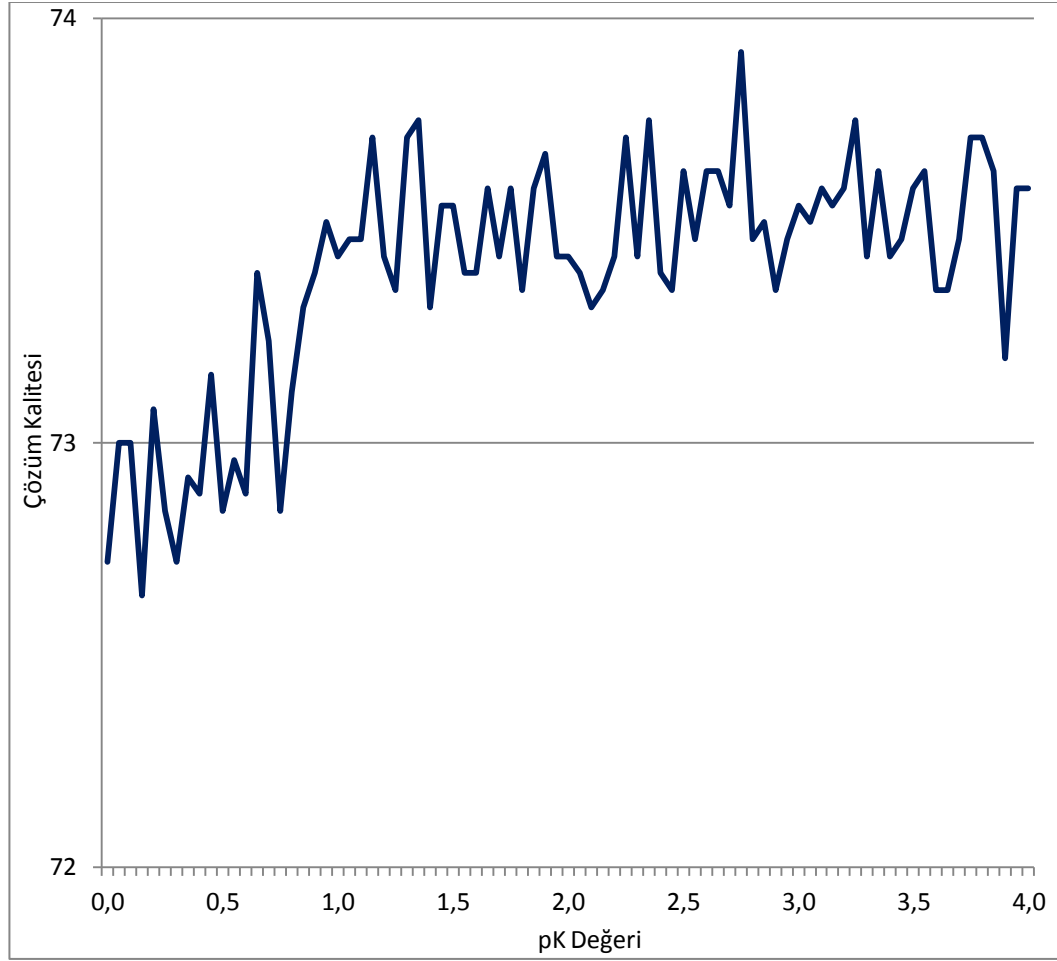
Şekil 6.3. pKS - Çözüm Kalitesi Grafiği (MKP)

Farklı pKS değerleri ile yapılan test sonuçları Şekil 6.4.'te gösterilmiştir. pKS değeri yükseldikçe çökelmenin hızlandığı ve çözüm kalitesinin düştüğü görülür. pKS değeri çok küçük seçildiğinde de belli bir tur sayısında çökeltme sağlanmadığı için çözüm kalitesi düşük kalmaktadır.



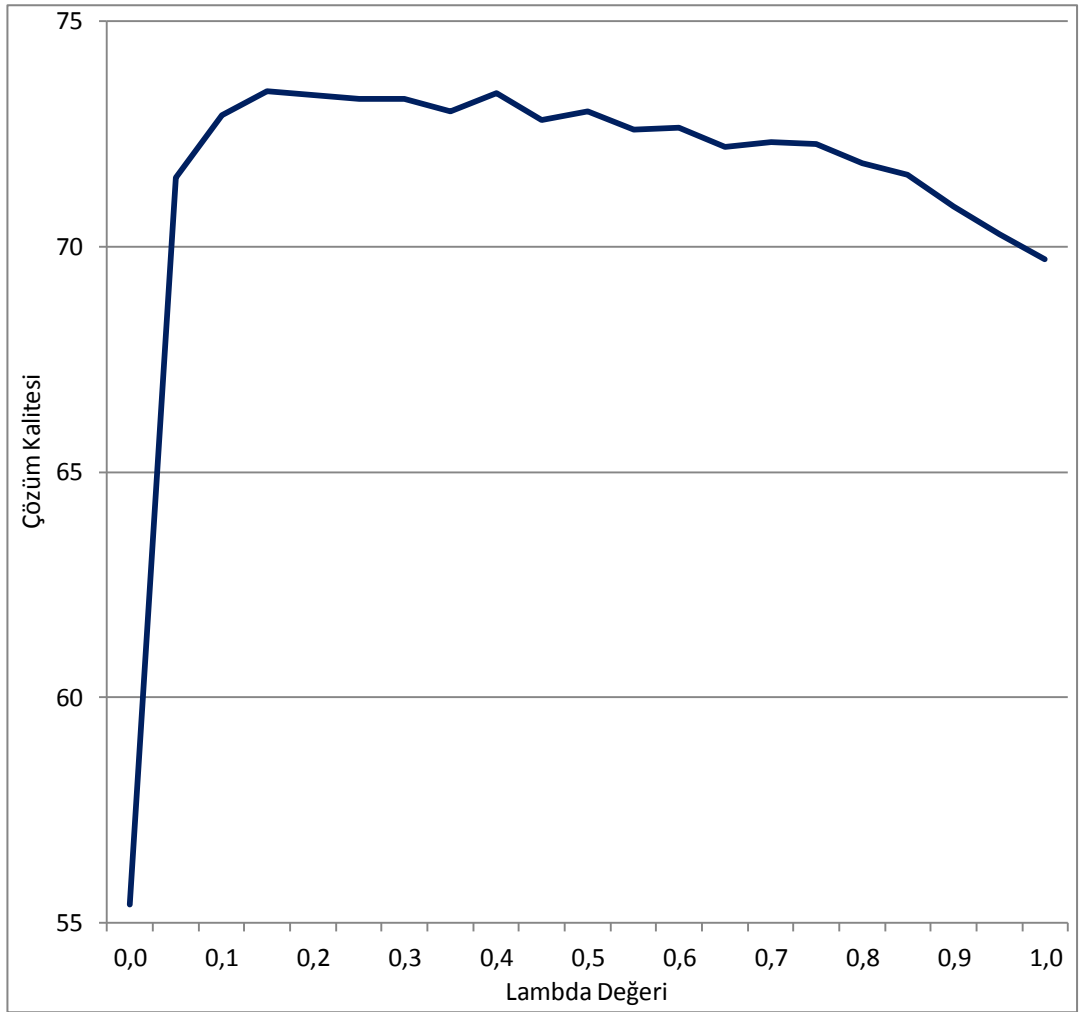
Şekil 6.4. Farklı pKS Değerleri İle Yapılan Test Sonuçları (MKP)

pK 'nin çözüm kalitesine etkisi Şekil 6.5.'te gösterilmiştir. pK 'nin pS 'den daha büyük değer alması sürüdeki bir parçacığın kişisel tecrübesinden daha fazla etkilenecek dolaşmasına yol açar. Bunu PSO'daki eylemsizlik(inertia) katsayısına benzetebiliriz. Bu sayının büyük olması farklılaşmayı artırır. Bu şekle göre pK 'nin 1 ile 3 arasında değerler almasının çözüm kalitesini arttırdığı gözlemlenmiştir. Testlerde 2.6 tercih edilmiştir.



Şekil 6.5. pK - Çözüm Kalitesi Grafiği (MKP)

λ 'nın çözüm kalitesine etkisi Şekil 6.6. ve Tablo 6.3.'te gösterilmiştir. λ , 0 değerini aldığı anda algoritmanın parçacıklar çözümlerine hiçbir bozma uygulamadıkları için ilk buldukları rastsal çözüm ile algoritmayı sonlandırırlar. Bu yüzden bu durumdaki çözüm kalitesi çok düşüktür. λ çok artınca da farklılaşma gereğinden fazla artacağı için çökelme zorlaşır. Bu problemde λ 'nın küçük değerleri için çözüm kalitesinin daha iyi olduğu gözlenmiş ve bütün problemlerin testinde 0.30 olarak seçilmiştir.

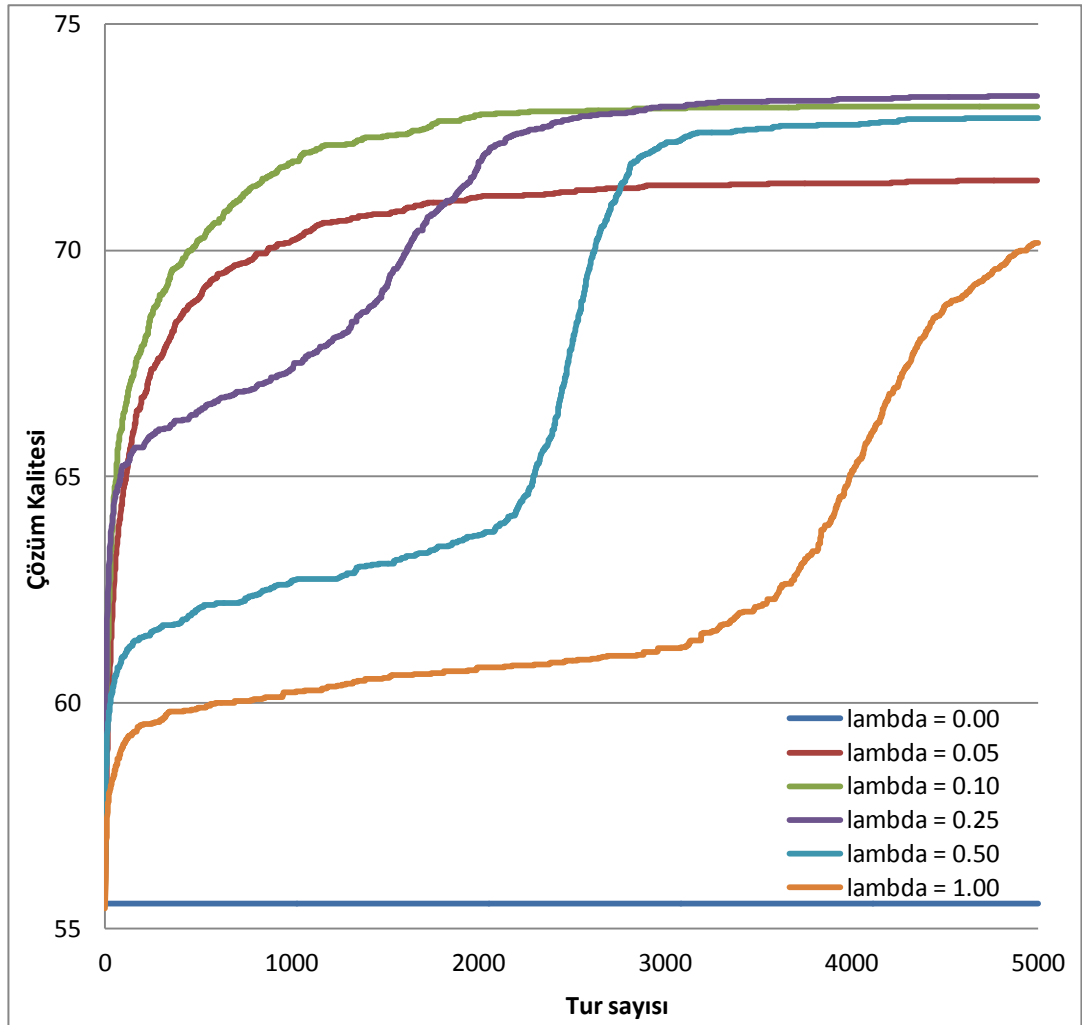


Şekil 6.6. Lambda - Çözüm Kalitesi Grafiği (MKP)

Tablo 6.3. Lambda - Çözüm Kalitesi Tablosu (MKP)

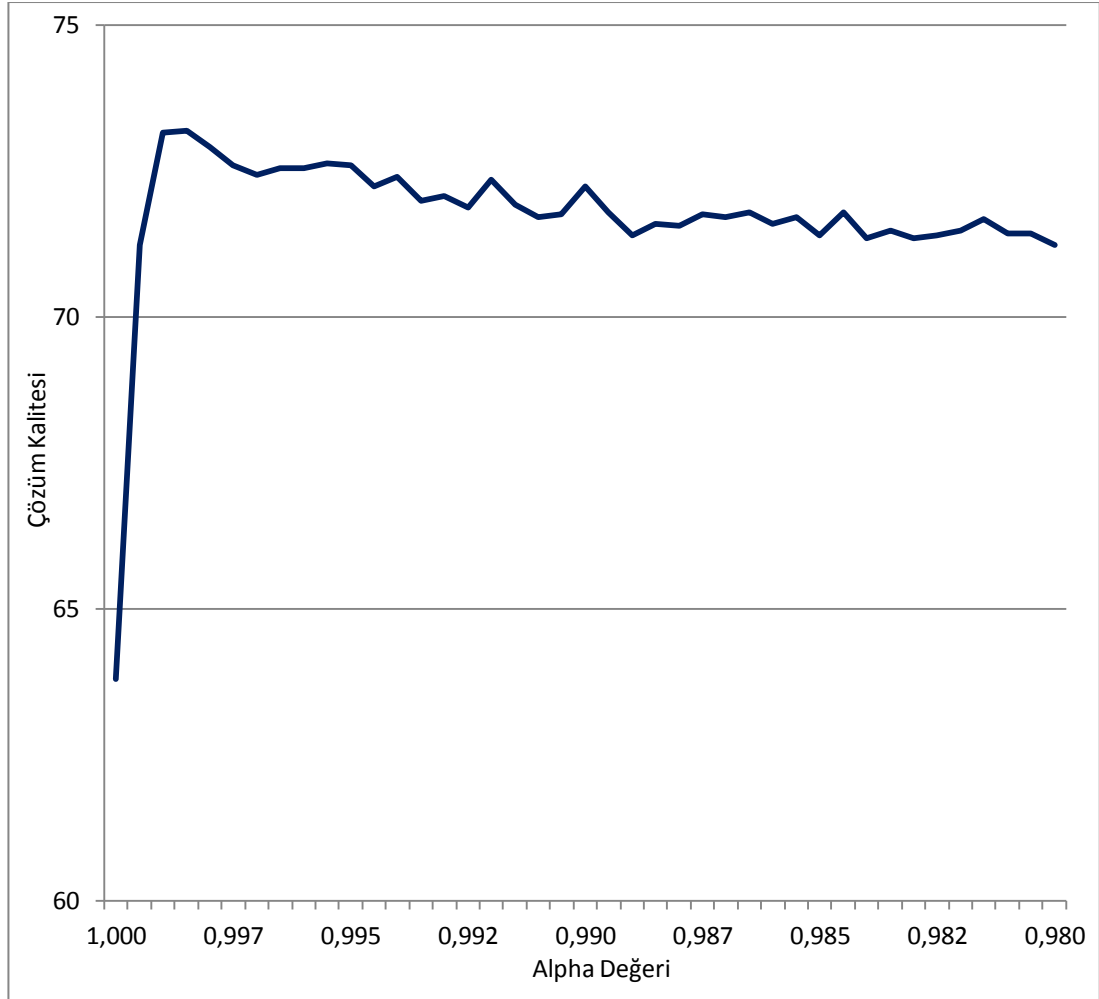
λ	En İyi	Ortalama	Std.	Süre
0,00	59	55,4	1,41	20,77
0,05	74	71,52	1,05	22,476
0,10	75	72,92	0,86	22,622
0,15	76	73,44	0,87	22,619
0,20	74	73,36	0,64	22,639
0,25	76	73,28	1,02	22,696
0,30	76	73,28	1,06	22,919
0,35	75	73	1,08	23,154
0,40	76	73,4	1,15	23,503
0,45	75	72,8	1,12	24,006
0,50	75	73	1,12	24,808
0,55	74	72,6	0,82	25,526
0,60	74	72,64	0,91	26,646
0,65	74	72,2	0,96	27,958
0,70	75	72,32	1,25	29,685
0,75	75	72,28	1,46	31,957
0,80	74	71,84	0,9	34,876
0,85	74	71,6	1,08	38,541
0,90	73	70,88	1,01	43,001
0,95	73	70,28	1,21	48,568
1,00	73	69,72	1,28	55,725

Farklı λ değerleri ile yapılan testler Şekil 6.7.'de gösterilmiştir. En alttaki düz çizgi λ 'nın 0 olduğu durumdur. Bu durumda bozma işlemi yapılmadığı için ilk turda bulunan çözümler geliştirilmeden algoritma sonlanır. λ küçük değerler aldığında çökelmenin çok erken gerçekleştiği görülür. 0.10 ve 0.25 değerlerinde algoritmanın çözüm kalitesinin en yüksek değerlere ulaştığı görülmüştür. Değeri büyötmeye devam ettiğimizde çökelme süresinin arttığı ve belirli tur sayısı içerisinde çözüm kalitesinin düştüğü görülmüştür.



Şekil 6.7. Farklı Lambda Değerleri İle Yapılan Test Sonuçları (MKP)

α 'nın çözüm kalitesine etkisi Şekil 6.8. ve Tablo 6.4'te gösterilmiştir. α , 1 değerini aldığı anda p_0 hiç küçülmeyeceği için algoritma rastsal davranmış ve iyi bir çözüm bulması mümkün olmamıştır. α 'nın küçülmesi p_0 'ın daha hızlı küçülmesi anlamına geleceği için farklılaşma azalacak ve algoritma daha çabuk çökelecektir. Sonuçlardan, bu durumun çözüm kalitesini düğürdüğü görülmektedir. Bütün testler yapılırken α , 0.9992 olarak seçilmiştir.

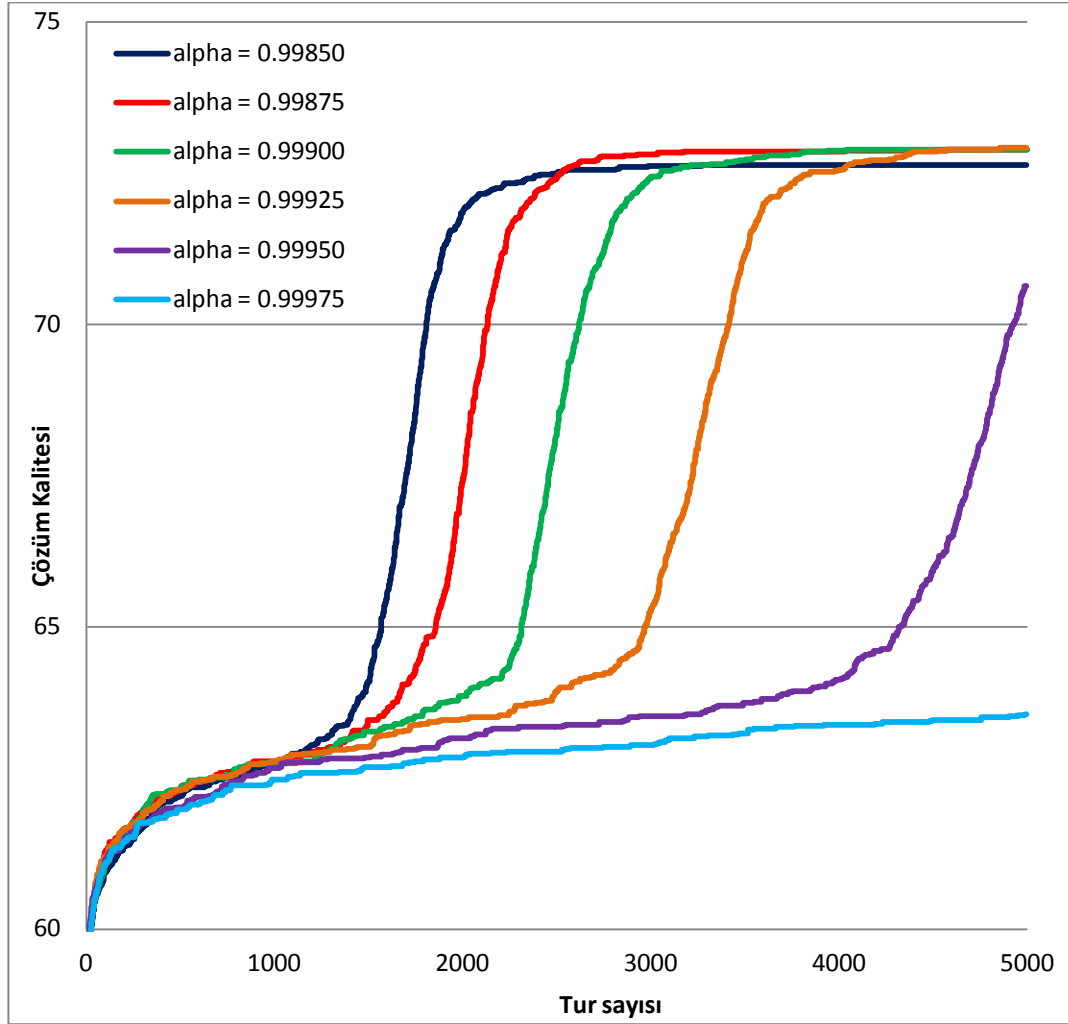


Şekil 6.8. Alpha – Çözüm Kalitesi Grafiği (MKP)

Tablo 6.4. Alpha - Çözüm Kalitesi Tablosu (MKP)

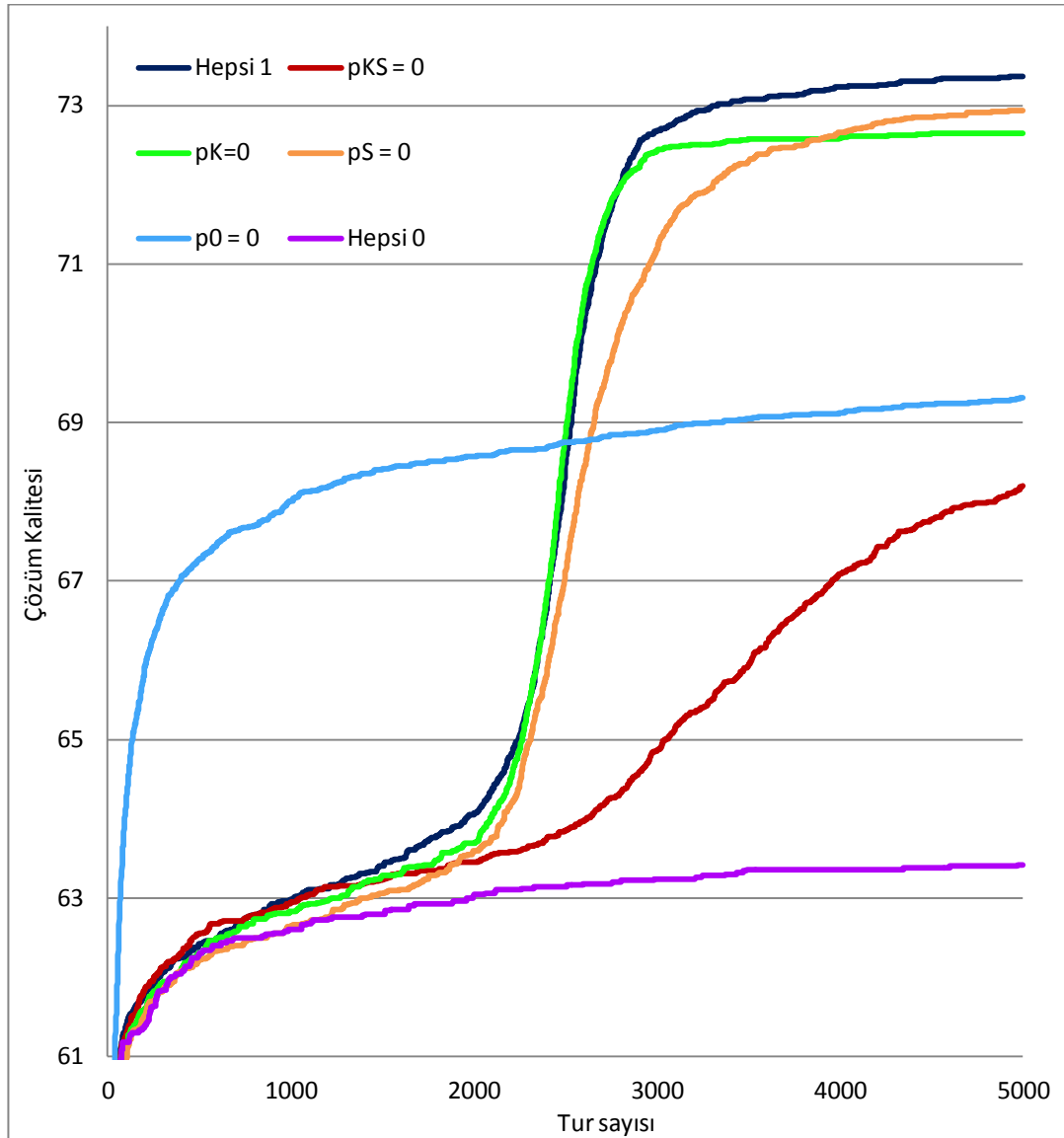
α	En İyi	Ortalama	Std.	Süre
1,0000	65,0000	63,8000	0,7071	53,2869
0,9995	74,0000	71,2400	1,3626	52,6762
0,9990	75,0000	73,1600	0,9866	52,1644
0,9985	76,0000	73,2000	0,9129	51,8100
0,9980	75,0000	72,9200	0,9539	51,1625
0,9975	75,0000	72,6000	0,9574	52,0519
0,9970	74,0000	72,4400	0,6506	50,9475
0,9965	74,0000	72,5600	0,8699	50,9606
0,9960	74,0000	72,5600	0,8206	50,8975
0,9955	74,0000	72,6400	0,9522	50,8844
0,9950	74,0000	72,6000	0,8165	50,7862
0,9945	75,0000	72,2400	1,0520	50,9644
0,9940	74,0000	72,4000	0,8660	48,3348
0,9935	75,0000	72,0000	1,0408	50,7594
0,9930	74,0000	72,0800	0,9539	50,8656
0,9925	73,0000	71,8800	0,6658	50,8400
0,9920	75,0000	72,3600	1,2871	51,0125
0,9915	75,0000	71,9200	1,0770	50,9581
0,9910	73,0000	71,7200	0,8907	50,8888
0,9905	74,0000	71,7600	0,8307	50,4212
0,9900	74,0000	72,2400	1,0116	50,5044
0,9895	74,0000	71,8000	0,7638	50,5469
0,9890	73,0000	71,4000	0,8165	50,6213
0,9885	73,0000	71,6000	0,7638	50,4700
0,9880	75,0000	71,5600	0,9165	50,3950
0,9875	75,0000	71,7600	1,1648	50,6862
0,9870	75,0000	71,7200	1,1373	51,0881
0,9865	73,0000	71,8000	0,7071	50,7225
0,9860	73,0000	71,6000	0,9129	50,6800
0,9855	75,0000	71,7200	1,1000	50,7006
0,9850	73,0000	71,4000	1,0000	50,6800
0,9845	74,0000	71,8000	1,0000	50,5856
0,9840	74,0000	71,3600	0,9950	50,6100
0,9835	73,0000	71,4800	0,8226	50,7338
0,9830	74,0000	71,3600	1,1504	50,6406
0,9825	73,0000	71,4000	0,8660	50,6256
0,9820	73,0000	71,4800	0,8226	50,7294

Farklı α değerleri ile yapılan testler Şekil 6.9.'da gösterilmiştir. Değer 1'e yaklaştıkça çökelmenin geciktiği ve çözüm kalitesinin arttığı şekilde açıkça görülmektedir. Belli bir değeri aştıktan sonra belli tur sayısında çökelleme sağlanamadığı için çözüm kalitesi düşük kalacaktır.



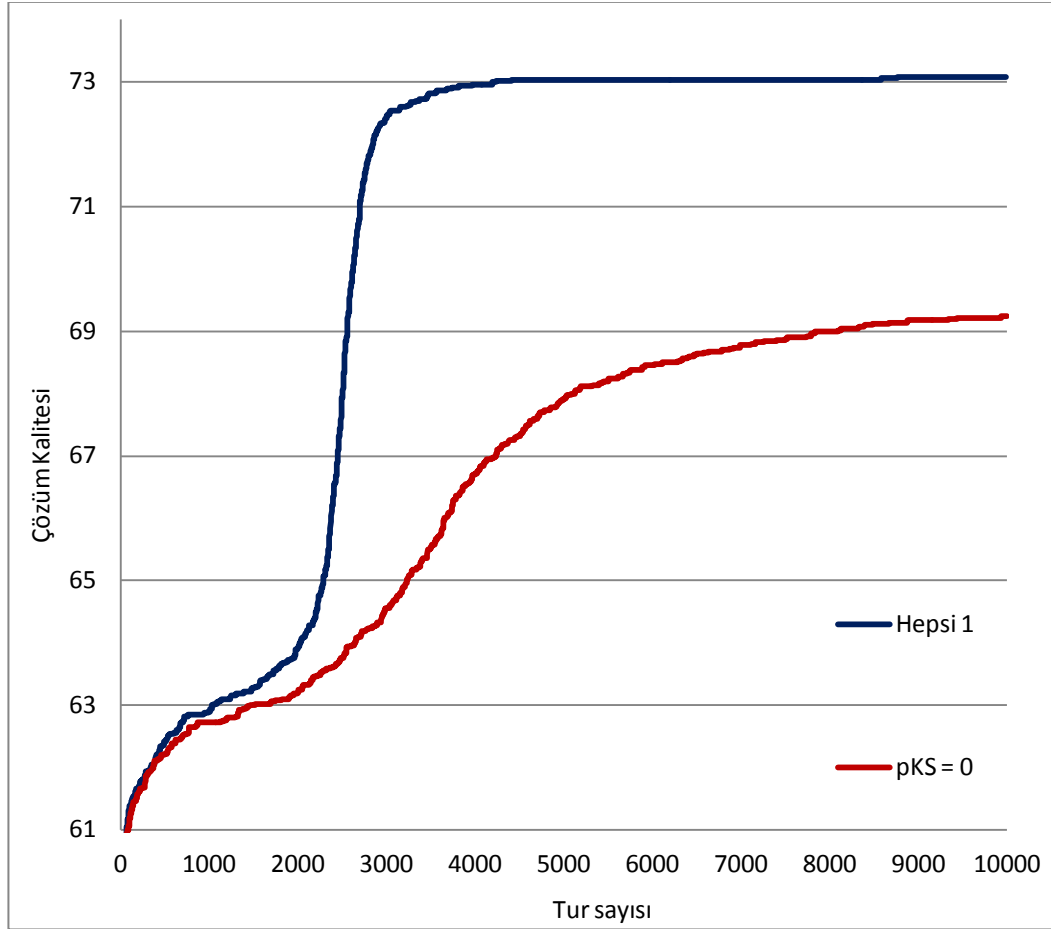
Şekil 6.9. Farklı Alpha Değerleri İle Yapılan Test Sonuçları (MKP)

Seçim oranları testinin sonuçları Şekil 6.10.'da gösterilmiştir. Hepsi bir seçilen durumda çözüm kalitesi en yüksek çıkmıştır. pK ve pS 0 olduğu durumlarda çözüm kalitesi ve algoritmanın çökelme zamanı çok farklı değildir ancak çözüm kalitesi biraz daha düşüktür. Bu durum sürü zekasının etkili olduğunun göstergesidir. $p0$ 'ın 0 olduğu durumda farklılaşma çok az olduğu için erken çökelme ve çözüm kalitesinin çok düşük olduğu görülmüştür. pKS 'nin 0 olduğu durumda ise çökelme çok gecikmiş hatta hiç gerçekleşmemiştir. Hepsi 0 durumunda çözüm kalitesinin diğer durumların çok gerisinde kaldığı görülmüştür.



Şekil 6.10. Seçim Oranları Testi (MKP)

Seçim oranları testi sonuçlarına bakıldığında pKS 'nin 0 olduğu durumun grafiğinin çıkışta olduğu görülmektedir. Buna dayanarak bu durumun tur sayısı artırıldığında daha yüksek çözümlere ulaşabileceği tahmin edilebilir. Ancak durumun böyle olmadığı tur sayısını artırarak yapılan testte görülmüştür. Sonuçları Şekil 6.11.'de görülen bu testte bütün parametrelerin bir olduğu ve sadece pKS 'nin 0 olduğu durumların karşılaştırması yapılmıştır.



Şekil 6.11. Hepsi 1 ve $pKS=0$ Karşılaştırması (10000 tur) (MKP)

Farklı seçim oranları değerleriyle çalıştırılan algoritmaların son durumdaki çözüm değerleri ortalamaları Tablo 6.5.'te verilmiştir.

Tablo 6.5. Seçim Oranlarının Karşılaştırılması (MKP)

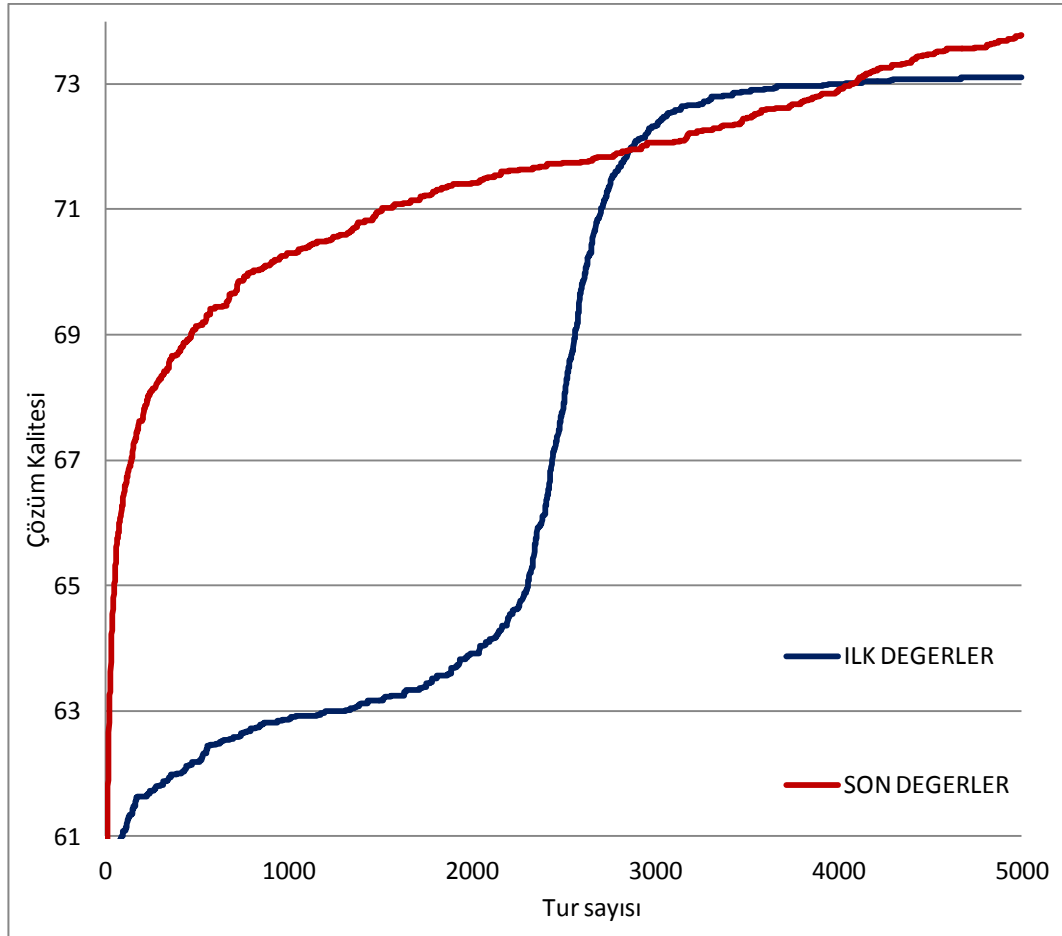
Hepsi 1	$pKS = 0$	$pK = 0$	$pS = 0$	$p0 = 0$	Hepsi 0
73,37	68,20	72,65	72,94	69,31	63,42

Başlangıçta seçilen parametre değerleri (ilk değerler) ile testler sonucunda tercih edilen değerler (son değerler) Tablo 6.6.'da verilmiştir.

Tablo 6.6. Parametrelerin İlk ve Son Değerleri

Değişken	İlk Değerler	Son Değerler
SB	25	80
MTS	5000	6000
pKS	1.0	0.15
pK	1.0	2.60
pS	1.0	1.00
$p0$	1.0	1.00
λ	0.5	0.14
α	0.999	0.9992

Parametrelerin ilk ve son değerlerinin çözüm kalitesine etkisi Şekil 6.12.'de gösterilmiştir.



Şekil 6.12. Parametrelerin İlk ve Son Değerlerinin Karşılaştırması (MKP)

Karşılaştırma problemleri ile yapılan testlerin sonuçları Tablo 6.7.'de verilmiştir. Bu tabloda V, çizgenin düğüm sayısını, C çizgede şu ana kadar bulunan en büyük kliğin büyüklüğünü göstermektedir. Eğer '*' işareti varsa, bulunan klik büyüklüğünün optimal olduğu kanıtlanmıştır. En iyi değer, ortalama değer ve parantez içinde standart sapma değerleri verilmiştir. Eğer standart sapma 0 ise o bölüm boş bırakılmıştır. Süreler saniye cinsindendir. Koyu gösterilen sayılar PSO-A'nın diğer algoritmalarından daha yüksek çözüm bulduğunu gösterir.

Tablo 6.7. PSO-A'nın MKP'ye Uygulanışının Karşılaştırmalı Sonuçları

			PSO-A			GLS				ANT			EA/G		
Çizgenin Adı	V	C	En iyi	Ort. (Std.)	Sr	En iyi	Ort. (Std.)	Sr	v.	En iyi	Ort. (Std.)	Sr	En iyi	Ort. (Std.)	Sr
C125.9	125	34*	34	34,00	0	34	34.00	0	I	34	34.00	0	34	34.00	1
C250.9	250	44*	44	44,00	1	44	43.00 (0.6)	0	I	44	44.00	1	44	44.00	2
C500.9	500	57	57	56,84 (0,3)	27	55	52.70 (1.4)	0	I	57	55.90 (0.6)	8	56	55.20 (0.9)	5
C1000.9	1000	68	68	66,40 (0,5)	68	66	61.60 (2.1)	0	G	68	66.20 (0.7)	49	67	64.40 (1.4)	21
C2000.9	2000	80	75	73,96 (0.6)	148	70	68.70 (1.2)	0	I	78	74.30 (1.4)	238	72	70.90 (1.0)	45
DSJC500.5	500	14	13	13,00	0	13	12.20 (0.4)	0	G	13	13.00	1	13	13.00	4
DSJC1000.5	1000	15	15	14,68 (0.4)	14	14	13.50 (0.5)	0	I	15	14.30 (0.4)	7	15	14.50 (0.3)	12
C2000.5	2000	16	16	15,64 (0.4)	34	15	14.20 (0.4)	0	I	16	15.30 (0.5)	40	16	14.90 (0.7)	28
C4000.5	4000	18	17	16,80 (0.4)	99	16	15.60 (0.5)	0	I	18	16.80 (0.6)	257	17	16.10 (0.3)	61
MANN_a27	378	126*	126	125,96 (0.2)	103	126	126.00	0	I	126	126.00	44	126	126.00	12
MANN_a45	1035	345*	338	337,28 (0.4)	819	345	343.10 (0.8)	2	I	344	342.90 (0.6)	749	345	343.70 (0.7)	80
brock200_2	200	12*	12	12,00	0	12	12.00	0	M	12	12.00	0	12	12.00	1
brock200_4	200	17*	17	16,92 (0.2)	1	17	15.70 (0.9)	0	M	17	16.80 (0.4)	1	17	16.50 (0.5)	2
brock400_2	400	29*	29	25,32 (1.1)	14	25	23.20 (0.7)	0	I	25	24.80 (0.4)	3	25	24.70 (0.4)	3
brock400_4	400	33*	33	27,24 (3,6)	11	25	23.60 (0.8)	0	G	33	27.10 (4.0)	5	33	25.10 (2.6)	3
brock800_2	800	24*	21	20,68 (0.4)	25	20	19.30 (0.6)	0	G	24	20.10 (0.6)	11	21	20.10 (0.4)	8
brock800_4	800	26*	21	20,56 (0,5)	25	20	19.00 (0.4)	0	I	26	20.00 (0.8)	11	21	19.90 (0.5)	8
gen200_p0.9_44	200	44*	44	44,00	3	44	39.70 (1.6)	0	G	44	44.00	0	44	44.00	25
gen200_p0.9_55	200	55*	55	55,00	0	55	50.80 (6.4)	0	G	55	55.00	0	55	55.00	30
gen400_p0.9_55	400	55	55	55,00	15	55	49.70 (1.2)	0	G	53	52.20 (0.4)	6	55	51.80 (0.7)	45
gen400_p0.9_65	400	65	65	65,00	6	65	53.70 (7.4)	0	G	65	65.00	2	65	65.00	47
gen400_p0.9_75	400	75	75	75,00	2	75	62.70 (12.3)	0	I	75	75.00	2	75	75.00	46
hamming8-4	256	16*	16	16,00	0	16	16.00	0	G	16	16.00	0	16	16.00	2
hamming10-4	1024	40	40	40,00	15	40	38.20 (1.2)	0	I	40	39.30 (0.9)	29	40	39.80 (0.6)	16
keller4	171	11*	11	11,00	0	11	11.00	0	G	11	11.00	0	11	11.00	1
keller5	776	27	27	27,00	5	27	26.30 (0.6)	0	I	27	27.00	12	27	26.90 (0.3)	10
keller6	3361	59	57	55,52 (0,7)	146	56	52.70 (1.8)	1	I	57	55.10 (1.3)	549	56	53.40 (1.2)	63
p_hat300-1	300	8*	8	8,00	0	8	8.00	0	G	8	8.00	0	8	8.00	2
p_hat300-2	300	25*	25	25,00	0	25	25.00	0	I	25	25.00	0	25	25.00	2
p_hat300-3	300	36*	36	36,00	0	36	35.10 (0.8)	0	I	36	36.00	0	36	36.00	2
p_hat700-1	700	11*	11	11,00	0	11	9.90 (0.7)	0	I	11	11.00 (0.1)	2	11	11.00	6
p_hat700-2	700	44*	44	44,00	0	44	43.60 (0.7)	0	I	44	44.00	4	44	44.00	8
p_hat700-3	700	62	62	62,00	4	62	61.80 (0.6)	0	I	62	62.00	7	62	62.00	13
p_hat1500-1	1500	12*	12	11,60 (0,5)	17	11	10.80 (0.4)	0	G	12	11.10 (0.2)	9	12	11.10 (0.3)	19
p_hat1500-2	1500	65	65	65,00	13	65	63.90 (2.0)	0	I	65	65.00	35	65	65.00	28
p_hat1500-3	1500	94	94	94,00	85	94	93.00 (0.8)	0	I	94	94.00	54	94	93.70 (0.5)	34

Karşılaştırma problemleri ile yapılan testlerde PSO-A'nın 8 tip problemde sadece birinde (*MANN*) diğer yöntemlerden daha kötü performans gösterdiği görülmüştür. Özellikle zor klik problemleri olan *brock* ve rastsal oluşturulan ve büyük çizgeler içeren *Cx.y* tipi çizgelerde PSO-A'nın karşılaştırılan algoritmalarından daha üstün olduğu görülmüştür. PSO-A, 36 değerlendirme çizgesinin 21'inde bütün çalışmalarda tam sonucu bulmuştur. 11 problemde diğer üç algoritmanın hepsinden de daha iyi bir sonuç bulunmuştur.

PSO-A'nın 36 değerlendirme çizgesi ile yapılan testler dikkate alınarak, ortalama klik büyüklüğü açılarından GLS, ANT ve EA/G ile karşılaştırmaları Tablo 6.8.'da gösterilmiştir ; '+' PSO-A'nın sonucunun iyi olduğu, '=' sonuçların eşit olduğu, '-' ise diğer çözümün iyi olduğu problem sayısını göstermektedir. Örneğin, PSO-A ile GLS karşılaştırıldığında, 36 problemin 28'inde PSO-A, 2'sinde GLS daha iyi sonuç vermiştir. 6 problemde ise sonuçlar eşittir.

Tablo 6.8. Test Sonuçlarının Karşılaştırması (MKP)

GLS			ANT			EA/G		
+	=	-	+	=	-	+	=	-
28	6	2	13	20	3	17	17	2

6.2. ÇSP ile Yapılan Deneyler

PSO-A modeli ÇSP üzerinde uygulanarak OR-Kütüphanesindeki [Beasley, 1990] karşılaştırmalı değerlendirme problemlerinin 9'u üzerinde test edilmiştir. Sonuçlar aynı problem üzerinde uygulanan bir genetik algoritma olan GENESYs [Khuri et al, 1994] ile karşılaştırılmıştır. Aşağıda GENESYs hakkında bilgi verilmiştir.

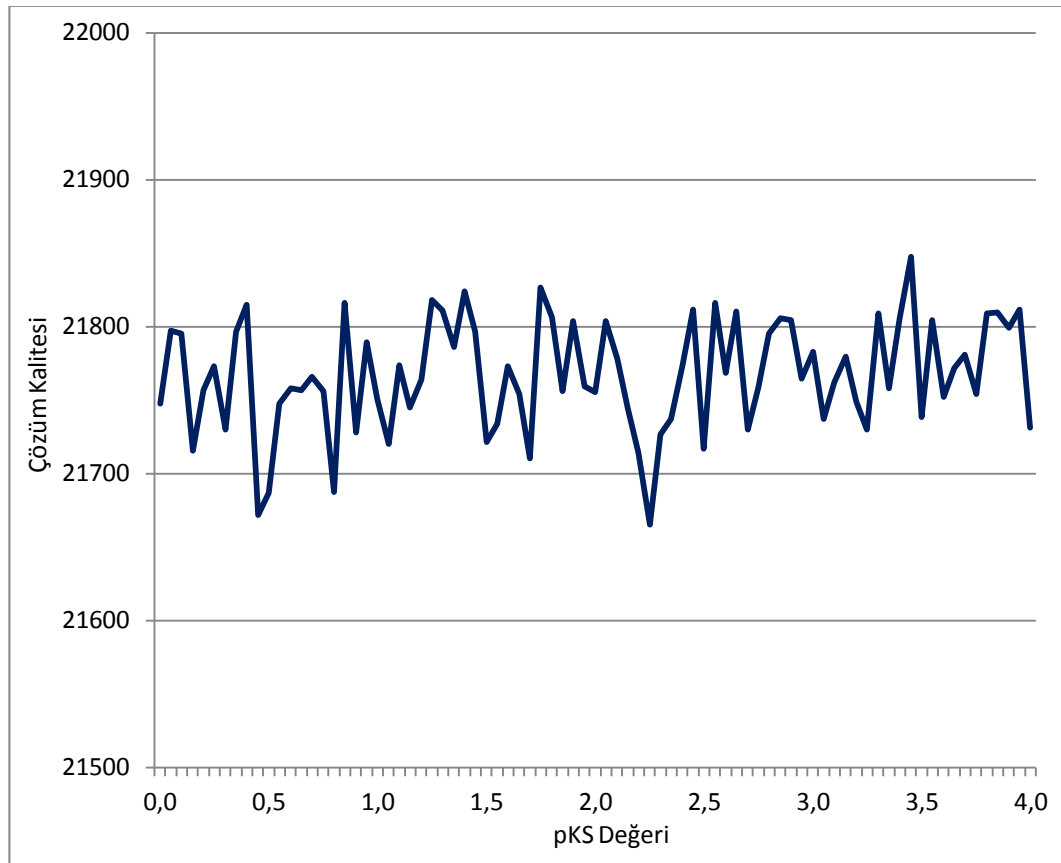
GENESYs [Khuri et al, 1994] : Problem, GA için şu şekilde kodlanmıştır. Her bir nesnenin çözümde olup olmaması bir x dizisi ile ifade edilmiştir. Algoritmada olursuz çözümlere bir ceza fonksiyonu ile izin verilmiştir. Bunun dışında standart GA adımları izlenmiştir.

Karşılaştırma Problemleri :

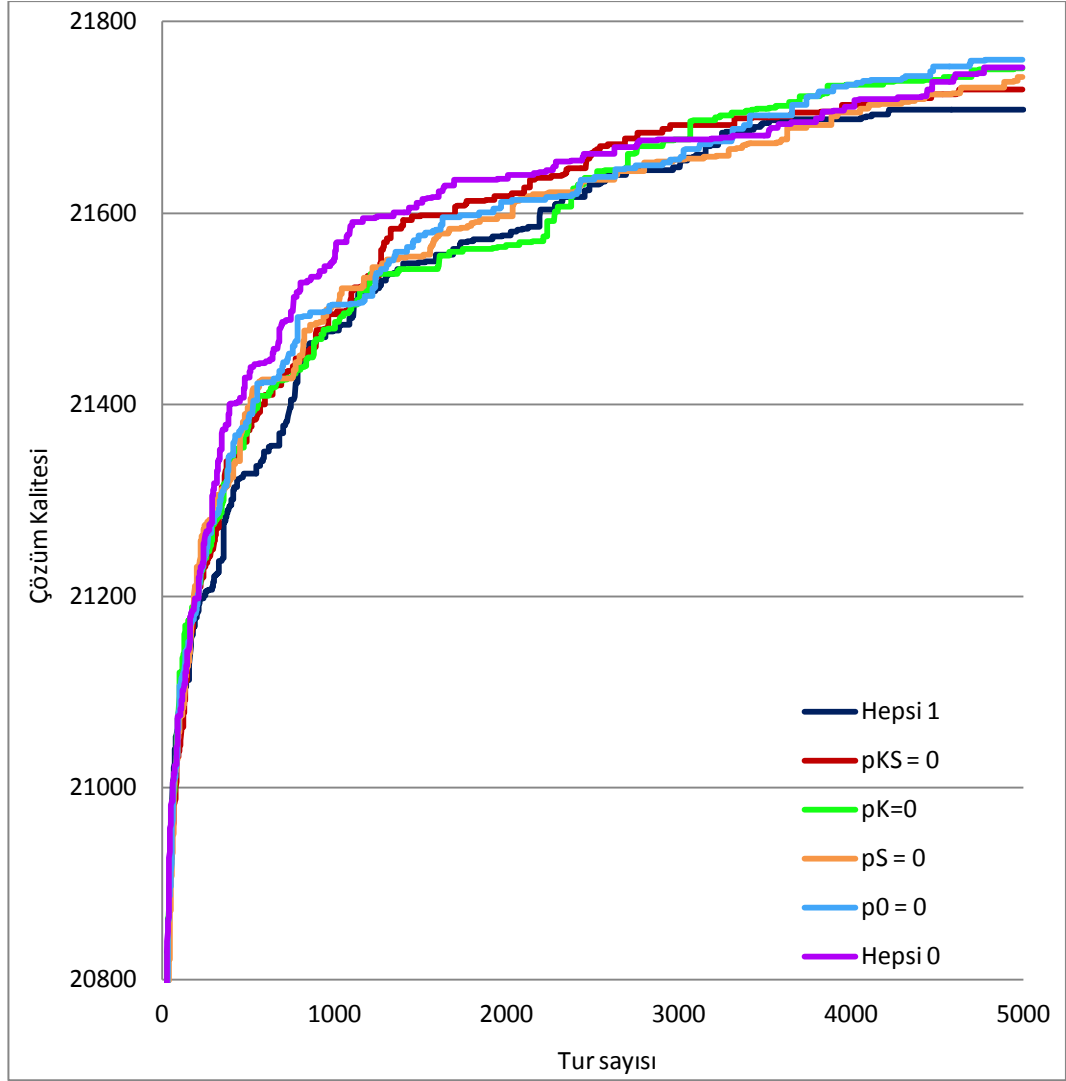
BHOSLIB kenar kapsama probleminden seçilen *WEIGN*, *SENTO* ile *mkncapcb* problemlerinden toplam 9 problem ile testler yapılmıştır. ÇKP’de temsili problem olarak OR-Kütüphanesindeki *mkncapcb1* isimli dosyadaki problemlerden ilki olan 100 nesneli, 5 boyutlu problem kullanılmıştır.

Test Sonuçları :

MKP yapılan testlerin tamamı ÇSP’de de yapılmıştır. Ancak ÇSP’de alınan sonuçlar MKP’de olduğu gibi anlamlandırılabilir görülmeyip daha çok rastsal bir davranışın sonuçları gibi görünmektedir. Örneğin Şekil 6.13.’de *pKS* değişiminin çözüm kalitesine etkisi görülmektedir. Diğer parametre testlerinde de benzer grafikler ortaya çıkmıştır.



Şekil 6.13. pKS - Çözüm Kalitesi Grafiği (ÇSP)



Şekil 6.14. Seçim Oranları Testi (ÇSP)

Tablo 6.9. Seçim Oranlarının Karşılaştırılması (ÇSP)

Hepsi 1	pKS = 0	pK = 0	pS = 0	p0 = 0	Hepsi 0
21708,7	21729,3	21751,5	21741,9	21759,9	21752,7

Seçim oranları testinin sonuçları Şekil 6.14. ve Tablo 6.9’da görülmüştür. Bu testlerin sonucunda, bu problem üzerinde seçim oranlarının çözüm kalitesini etkilemediği hatta rastsal nokta seçimi durumu ile (hepsi 0 durumu) aynı karaktere sahip olduğu görülmüştür.

Başarılı bir parametre analizi yapılamadığı için MKP için tercih edilen parametre değerleri aynen kullanılarak testler yapılmıştır.

Geliştirilen algoritmanın karşılaştırma çizgeleri ile yapılan test sonuçları ve GENESys algoritması ile karşılaştırması Tablo 6.10.'da gösterilmiştir. Bu tabloda n, nesne sayısını, m boyut sayısını, opt. problemin optimal çözüm değerini, #opt algoritmanın testler sırasında optimal değeri kaç kez bulunduğunu gösterir. Testler PSO için 20 kez, GENESys için 100 kez çalıştırılmıştır. Küçük problemlerde PSO-A, GENESys'den daha iyi sonuç bulduğu ancak problem büyüdükçe PSO-A'nın performansının hızla düştüğü görülmüştür.

Tablo 6.10. PSO-A'nın ÇSP'ye Uygulanışının Karşılaştırmalı Sonuçları

				PSO-A					GENESys		
	n	m	opt.	En İyi	Ort.	Sd.	Süre	#opt	En İyi	Ort.	#opt
knap15	15	10	4015	4015	4015	0	0.04	20	4015	4012	83
knap20	20	10	6120	6120	6109	9	0.97	5	6120	6102	33
knap28	28	10	12400	12400	12295	80	1,80	5	12400	12374	33
knap39	39	5	10618	10588	10567	15	2,15	0	10618	10536	4
knap50	50	5	16537	16436	16361	45	3,13	0	16537	16378	1
SENT01	60	30	7772	7598	7291	129	8,21	0	7772	7626	5
SENT02	60	30	8722	8525	8448	43	12,31	0	8722	8685	2
WEING7	105	2	1095445	1088522	1086018	1121	73,53	0	1095382	1093897	0
WEING8	105	2	624319	565627	536931	10513	50,73	0	624319	613383	6

6.3. KKP ile Yapılan Deneyler

PSO-A modeli KKP üzerinde uygulanarak ikinci BHOSLIB karşılaştırmalı değerlendirme çizgelerinden 24 çizge ile test edilmiştir [Xu and Li, 2000]. Sonuçlar aynı problem üzerinde uygulanan bir karınca kolonisi algoritması olan CKACS [Gilmour and Dras, 2006] ile karşılaştırılmıştır. Aşağıda bu algoritma hakkında bilgi verilmiştir.

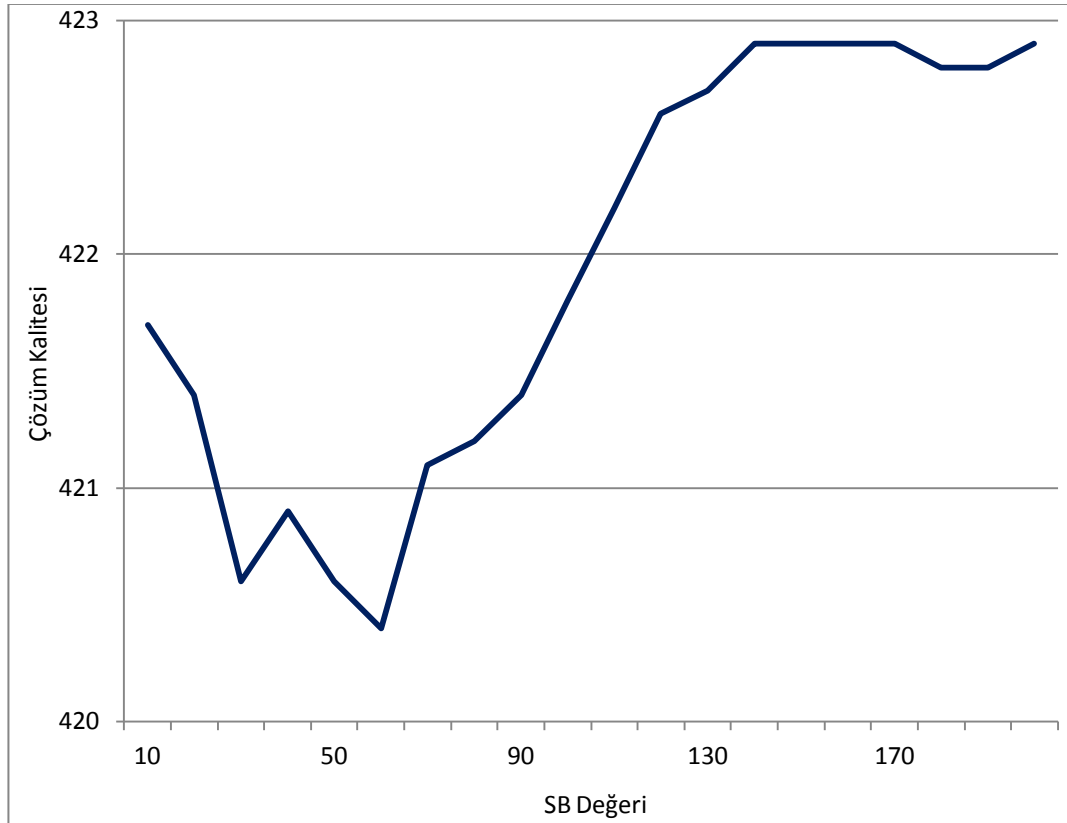
CKACS [Gilmour and Dras, 2006] : Özçekirdekleme yöntemi (kernelization), problem uzayını polinom zamanda bazı parametrelerden bağımsız olarak daha küçük bir problem özüne dönüştürmeyi amaçlayan bir yöntemdir. Bu algoritmada, KKP üzerinde özçekirdekleme yöntemi ile birlikte karınca kolonisi algoritması kullanılmıştır. Bu çalışma kapsamında sadece öz problemdeki düğümleri dikkate almak, öz problemdeki düğümlere farklı feromon ağırlıkları vermek, özde ve öz dışarısında farklı karınca kolonileri kullanmak gibi altı farklı yöntem denenmiştir. Bu yöntemler BHOSLIB problemleri üzerinde test edilmiştir. Bu testler sonucunda en iyi performans gösteren yöntemin, özçekirdek bilgisini kullanarak feromon güncellemesi yapan “CycleKernelized Ant Colony System (CKACS)” olduğu görülmüştür. PSO-A’nın değerlendirme problemleri ile yapılan karşılaştırmalı sonuçları verilirken CKACS sonuçları kullanılmıştır.

Karşılaştırma Problemleri :

BHOSLIB kenar kapsama problemi örnekleri, optimal çözümün bulunması zor olması için geliştirilmiş bir algoritma ile yapay olarak oluşturulmuş zor problemlerdir. KKP’de temsili problem olarak ikinci BHOSLIB problemlerinden olan 450 düğümlü, optimal kapsayan küme büyüklüğü 420 olan frb30-15-1 isimli çizge kullanılmıştır.

Test Sonuçları :

Sürü boyutunun (SB) çözüm kalitesine etkisi Şekil 6.15. ve Tablo 6.11.'de gösterilmiştir. Çözüm kalitesinin, sürü boyutu arttıkça belli bir noktadan sonra kötüleşmesinin nedeni tur sayısının SB arttıkça düşmesi ve algoritmanın çökelme sağlayamadan sonlanmasıdır. Bu testler sonucunda SB, 80 olarak seçilmiştir.

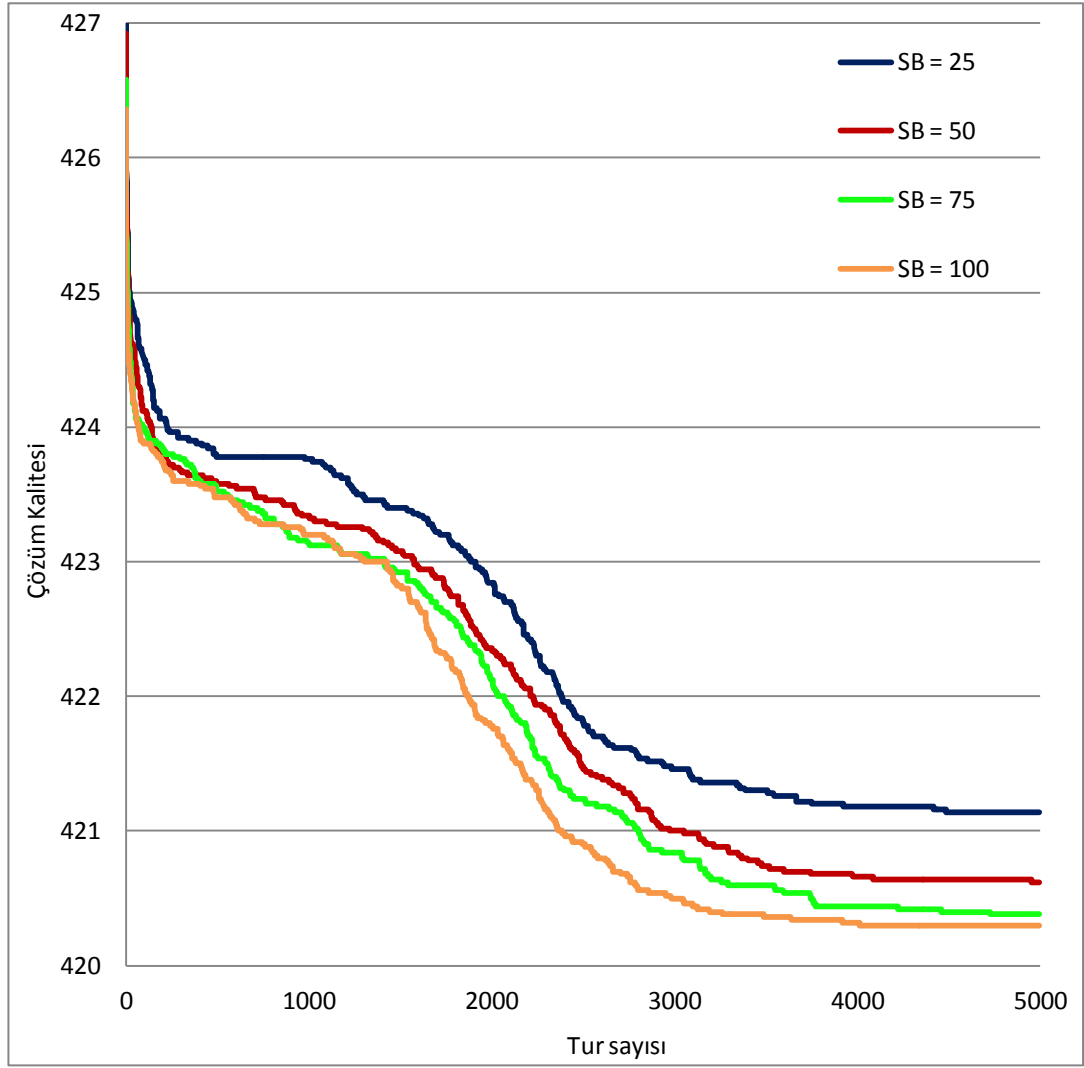


Şekil 6.15. SB - Çözüm Kalitesi Grafiği (KKP)

Tablo 6.11. SB - Çözüm Kalitesi Tablosu (KKP)

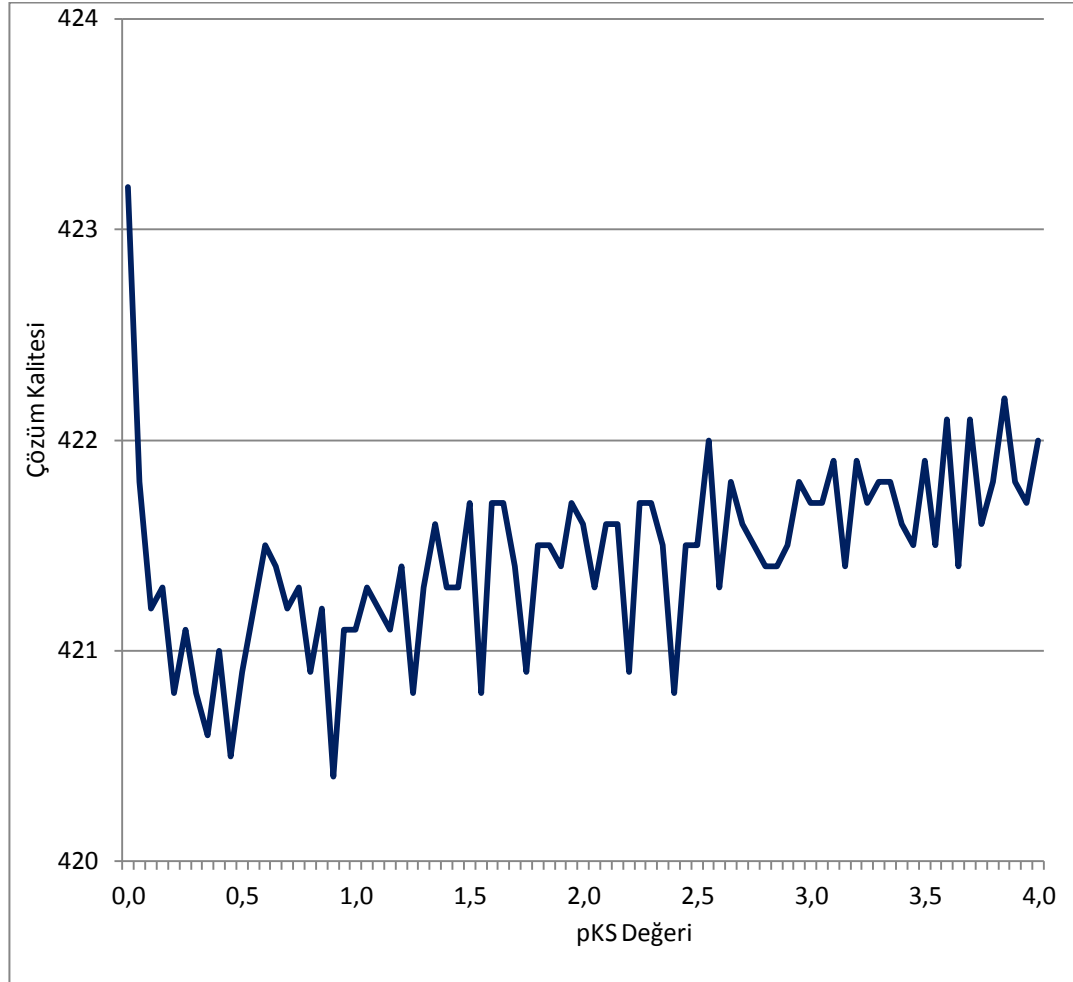
SB	En İyi	Ortalama	Std.	Süre
10	420	421,700	0,670	386,497
20	420	421,400	0,700	415,395
30	420	420,600	0,700	361,899
40	420	420,900	0,990	378,265
50	420	420,600	0,840	356,319
60	420	420,400	0,700	377,360
70	420	421,100	0,740	395,355
80	420	421,200	0,920	394,722
90	420	421,400	0,700	481,974
100	420	421,800	1,030	453,269
110	421	422,200	0,630	427,284
120	422	422,600	0,520	413,308
130	421	422,700	0,670	411,444
140	422	422,900	0,320	406,886
150	422	422,900	0,320	414,438
160	422	422,900	0,570	409,304
170	422	422,900	0,320	409,461
180	421	422,800	0,630	408,974
190	422	422,800	0,420	409,835

Farklı SB değerlerinin çözüm kalitesine etkisi Şekil 6.16.'da gösterilmiştir. Tur sayısı sabit olarak yapılan bu testlerde, sürü boyutu arttıkça çözüm kalitesinin bir miktar arttığı, sürü boyutunu artırmaya devam ettirtikçe bu artışın küçüldüğü görülmüştür.



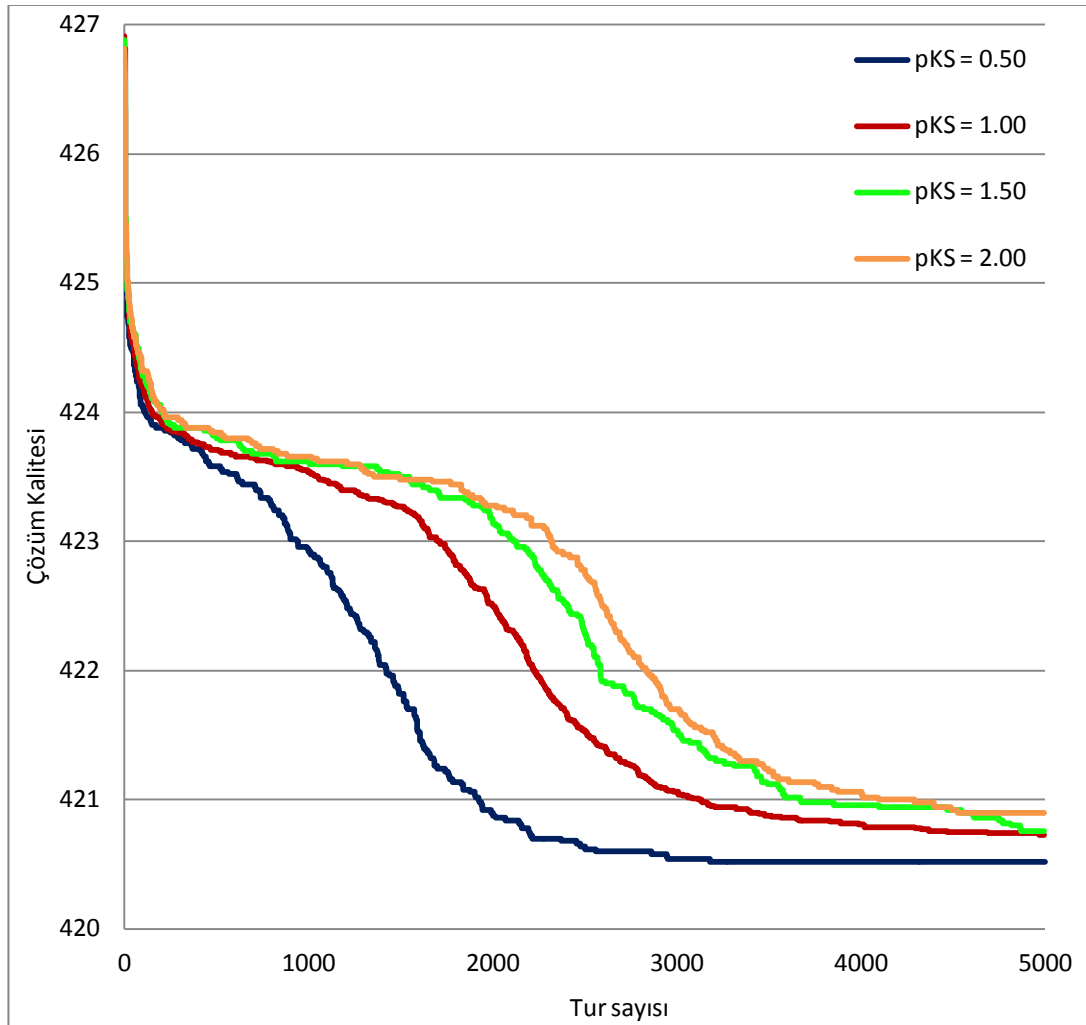
Şekil 6.16. Farklı SB Değerleri İle Yapılan Test Sonuçları (KKP)

pKS 'nin çözüm kalitesine etkisi Şekil 6.17.'de gösterilmiştir. Grafikler benzer karakter taşıdığı için MKP'de pKS değişiminin çözüm kalitesine etkisi ile ilgili söylenenlerin hepsi ÇSP için de söylenebilir. Bütün problemler için testler yapılırken pKS , 0.65 olarak seçilmiştir.



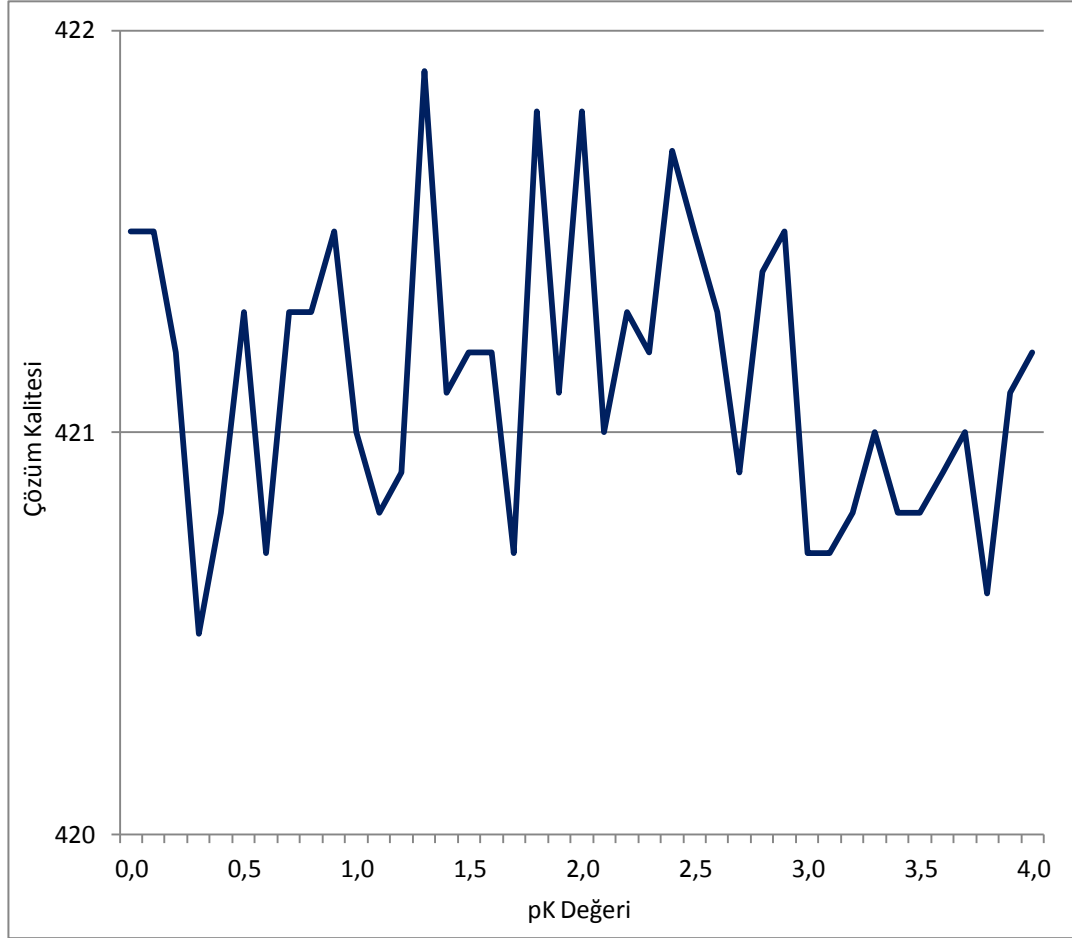
Şekil 6.17. pKS - Çözüm Kalitesi Grafiği (KKP)

Farklı pKS değerleri ile yapılan test sonuçları Şekil 6.18.'de gösterilmiştir. Bu şekilde, yukarıda yapılan tahmin daha net bir şekilde izlenmektedir.



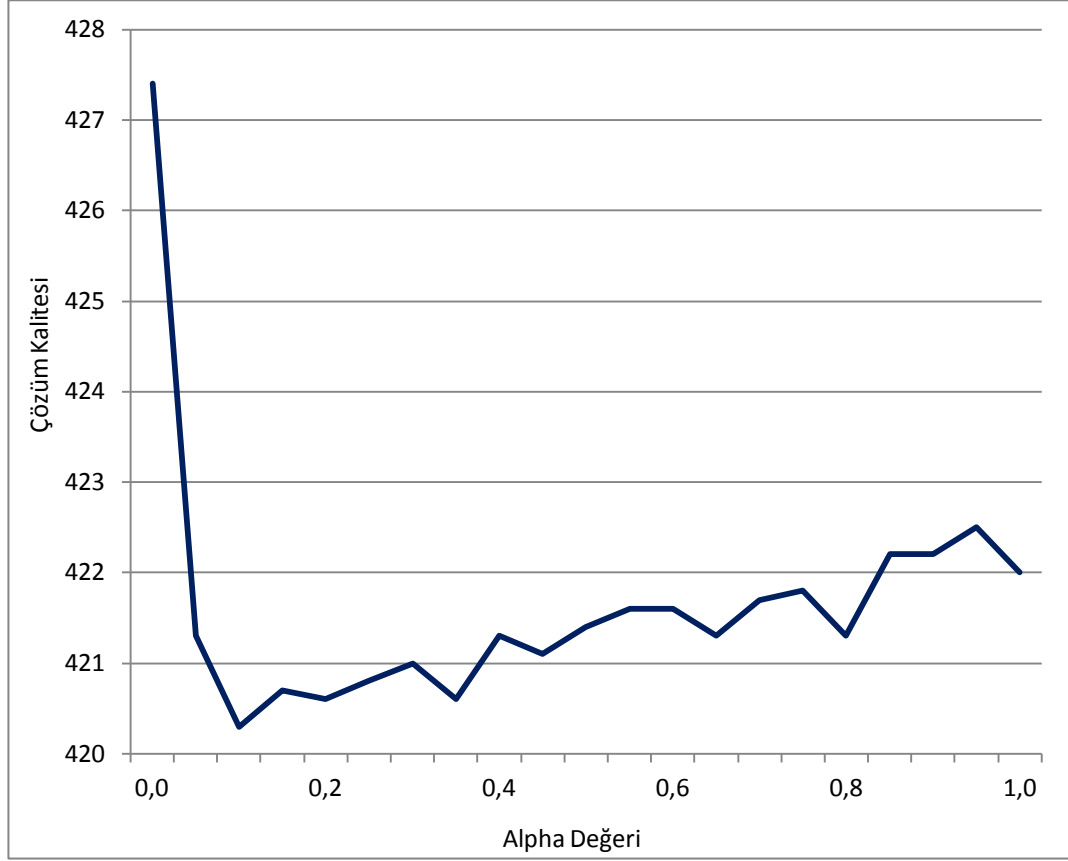
Şekil 6.18. Farklı pKS Değerleri İle Yapılan Test Sonuçları (KKP)

pK 'nin sürü çözüm kalitesine etkisi Şekil 6.19.'da gösterilmiştir. Bu şekle göre pK 'nin 0.2 ile 1.2 arasında değerler almasının çözüm kalitesini arttırdığı gözlemlenmiştir. Testlerde 0.3 tercih edilmiştir.



Şekil 6.19. pK - Çözüm Kalitesi Grafiği (KKP)

λ 'nın çözüm kalitesine etkisi Şekil 6.20. ve Tablo 6.12.'de gösterilmiştir. Burada da parametrenin çözüme etkisi MKP'de olduğu gibidir. Bütün testlerde değer 0.125 olarak alınmıştır.

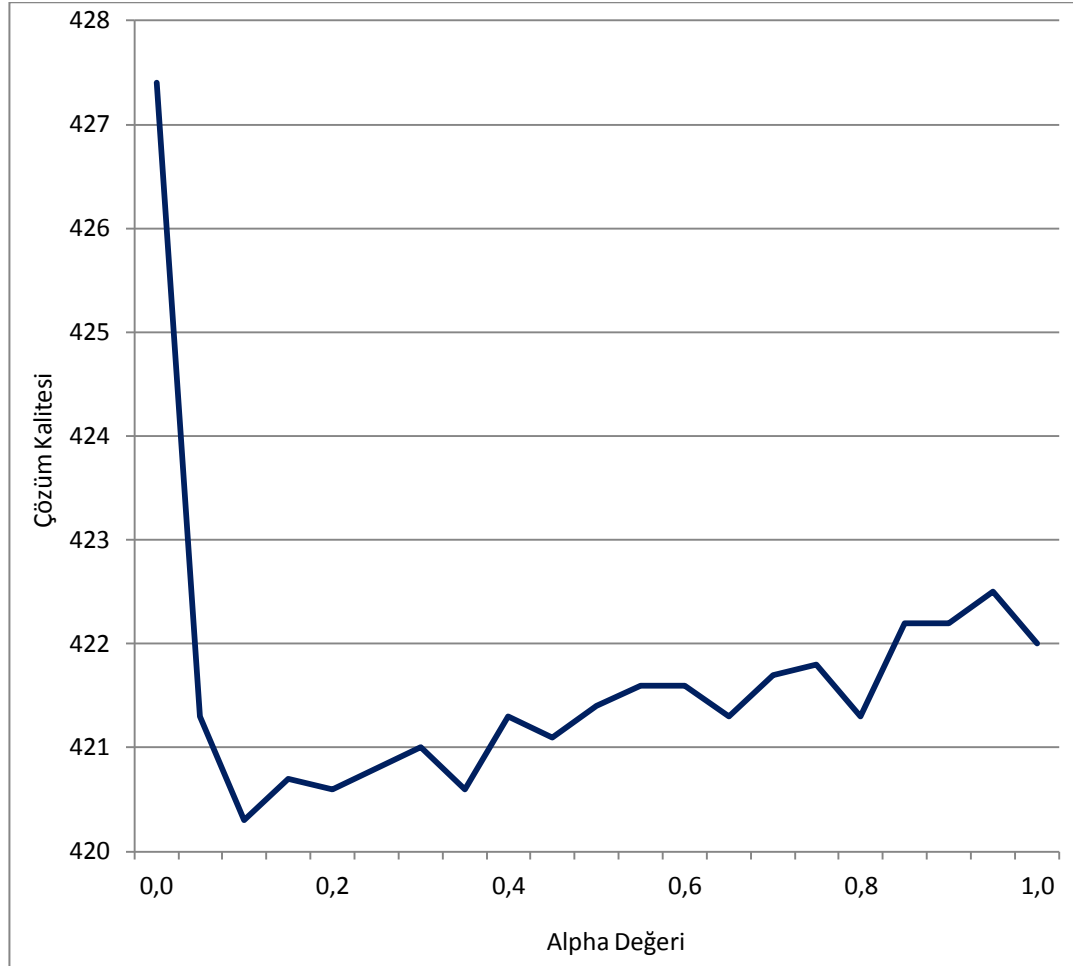


Şekil 6.20. Lambda - Çözüm Kalitesi Grafiği (KKP)

Tablo 6.12. Lambda - Çözüm Kalitesi Tablosu (KKP)

λ	En İyi	Ortalama	Std.	Süre
0,000	426	427,400	0,970	103,946
0,050	420	421,300	0,820	80,695
0,100	420	420,300	0,480	50,829
0,150	420	420,700	0,820	93,358
0,200	420	420,600	0,970	70,233
0,250	420	420,800	0,920	99,181
0,300	420	421,000	0,820	116,295
0,350	420	420,600	0,840	120,396
0,400	420	421,300	0,950	144,532
0,450	420	421,100	0,740	162,753
0,500	420	421,400	0,840	185,611
0,550	420	421,600	0,700	206,779
0,600	420	421,600	0,970	212,050
0,650	420	421,300	0,820	224,967
0,700	420	421,700	0,670	262,284
0,750	420	421,800	0,790	260,717
0,800	420	421,300	1,060	250,358
0,850	421	422,200	0,630	280,680
0,900	422	422,200	0,420	336,572
0,950	422	422,500	0,530	393,580
1,000	420	422,000	0,940	406,118

α 'nın çözüm kalitesine etkisi Şekil 6.21. ve Tablo 6.13.'te gösterilmiştir. Sonuçlar yine MKP ile büyük benzerlikler taşımaktadır ve aynı argümanlar burada da söylenebilir. Bütün testler yapılırken α , 1.001 olarak seçilmiştir.

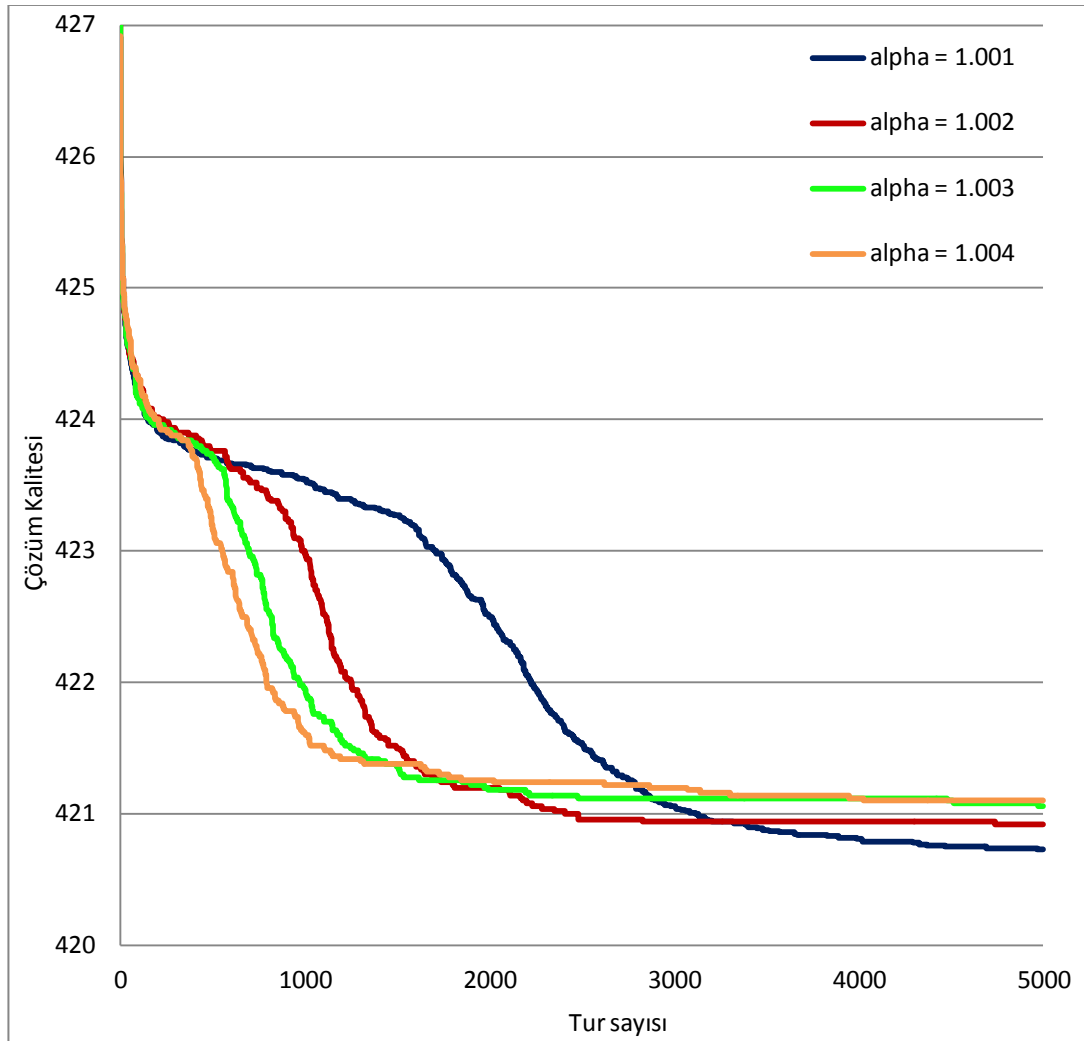


Şekil 6.21. Alpha - Çözüm Kalitesi Grafiği (KKP)

Tablo 6.13. Alpha - Çözüm Kalitesi Tablosu (KKP)

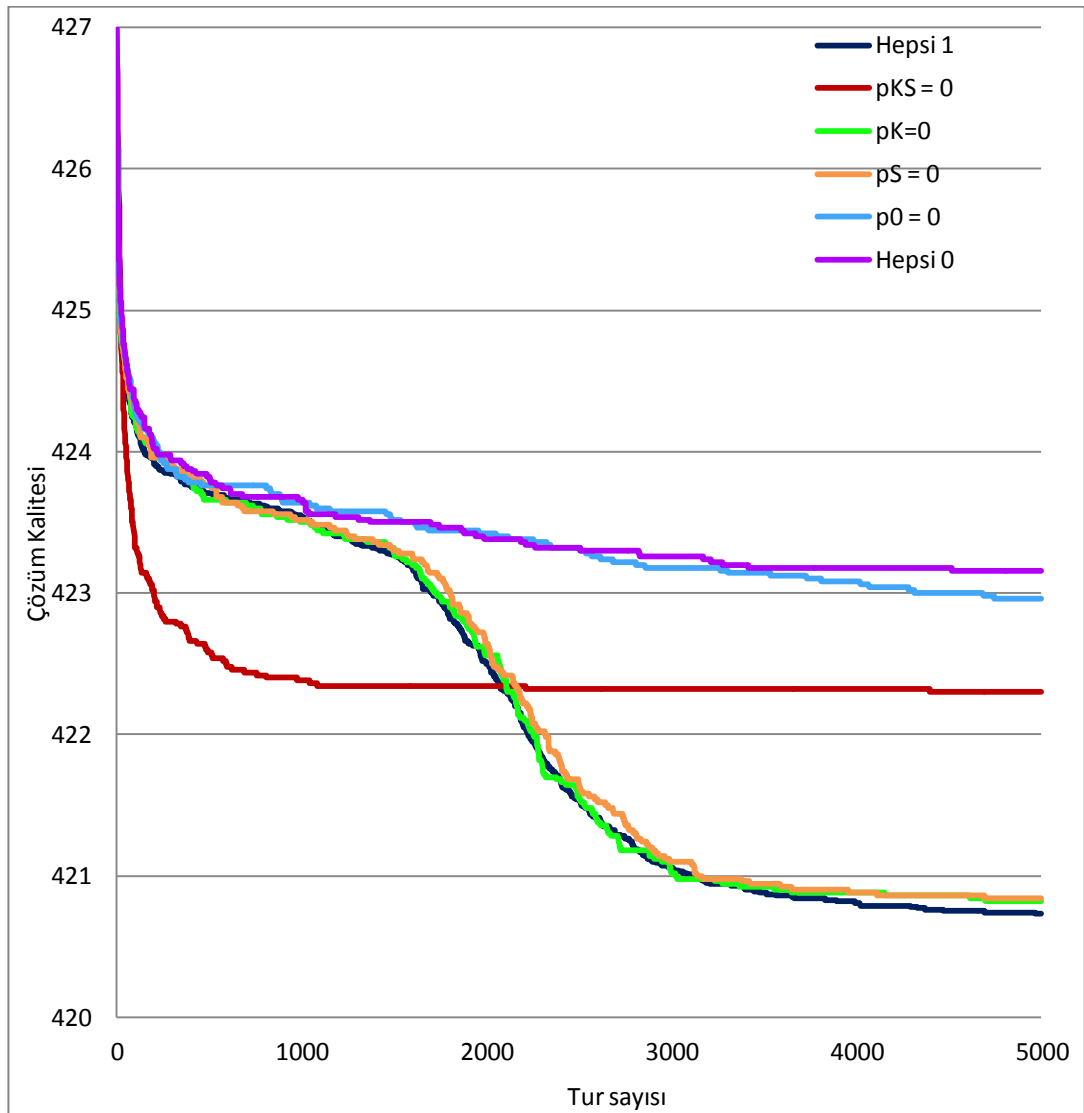
α	En İyi	Ortalama	Std.	Süre
1,000	423	423,400	0,520	167248,000
1,001	420	421,000	0,940	148780,000
1,002	421	421,700	0,480	156983,000
1,003	421	421,900	0,740	151486,000
1,004	420	421,900	0,880	127988,000
1,005	420	421,600	0,700	120260,000
1,006	421	421,700	0,480	149738,000
1,007	420	421,800	0,790	130452,000
1,008	421	421,800	0,630	142435,000
1,009	421	421,800	0,630	142904,000
1,010	420	421,800	0,920	128268,000
1,011	421	421,700	0,480	142294,000
1,012	420	421,900	0,740	138529,000
1,013	421	421,900	0,570	141502,000
1,014	422	422,100	0,320	142188,000
1,015	422	422,400	0,520	142954,000
1,016	420	422,000	0,820	134954,000
1,017	420	421,700	0,670	132624,000
1,018	420	421,700	0,820	133586,000
1,019	421	422,000	0,670	140892,000
1,020	421	421,900	0,320	141506,000

Farklı α değerleri ile yapılan testler Şekil 6.22.'de gösterilmiştir. Değer 1'e yaklaştıkça çökelmenin geciktiği ve çözüm kalitesinin arttığı şekilde görülmektedir.



Şekil 6.22. Farklı Alpha Değerleri İle Yapılan Test Sonuçları (KKP)

Seçim oranları testinin sonuçları Şekil 6.23.'te gösterilmiştir. Bu testin sonuçları da MKP ile büyük benzerlik göstermektedir. Hepsi bir seçilen durumda çözüm kalitesi en yüksek çıkmıştır. pK ve pS 0 olduğu durumlarda çözüm kalitesi ve algoritmanın çökelme zamanı çok farklı değildir ancak çözüm kalitesi biraz daha düşüktür. Bu durum sürü zekasının etkili olduğunun göstergesidir. $p0$ 'ın 0 olduğu durumda farklılaşma çok az olduğu için erken çökelme ve çözüm kalitesinin çok düşük olduğu görülmüştür. pKS 'nin 0 olduğu durumda ise çökelme çok gecikmiş hatta hiç gerçekleşmemiştir ve çözüm kalitesi çok düşük kalmıştır. Hepsi 0 durumunda da beklenildiği gibi çözüm kalitesi çok düşük kalmıştır.



Şekil 6.23. Seçim Oranları Testi (KKP)

Farklı seçim oranları değerleriyle çalıştırılan algoritmaların son durumdaki çözüm değerleri ortalamaları Tablo 6.14.'te verilmiştir.

Tablo 6.14. Seçim Oranlarının Karşılaştırılması (KKP)

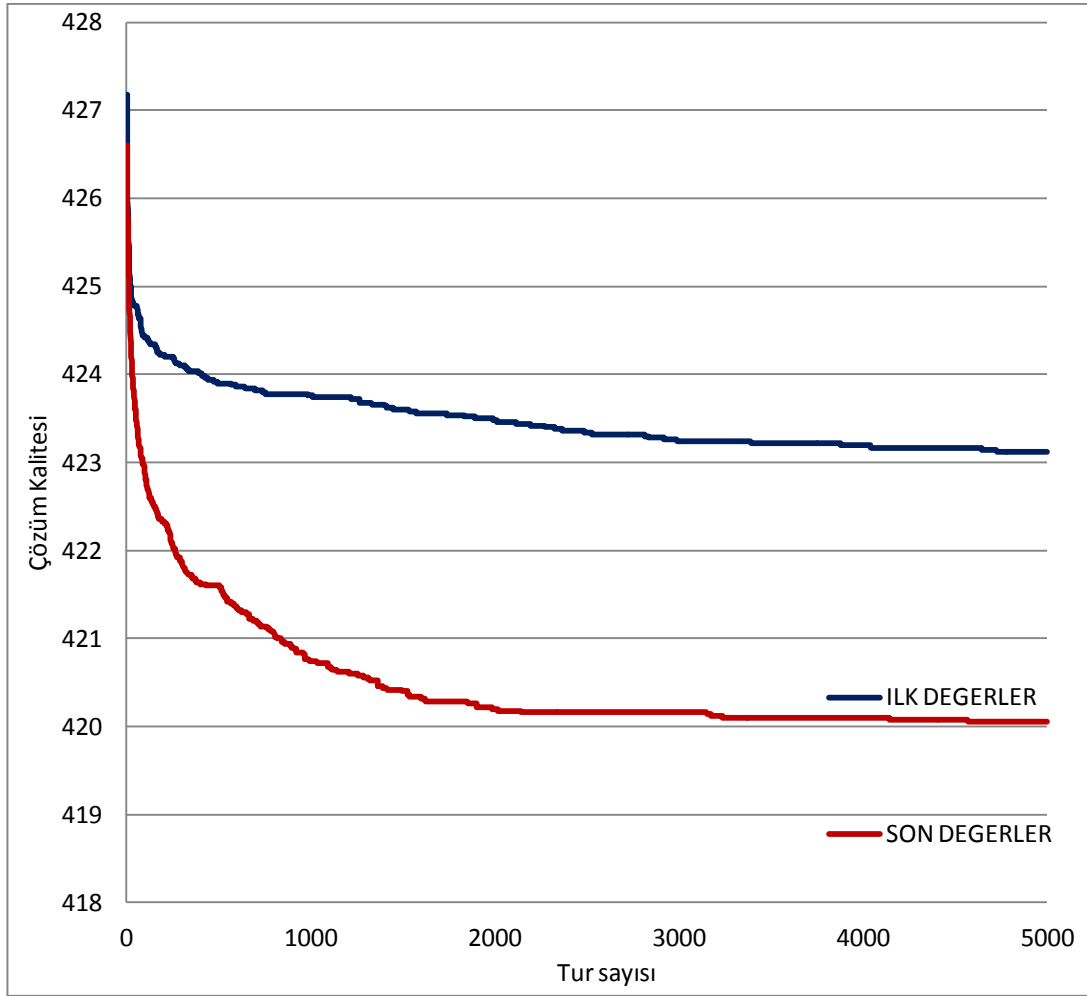
Hepsi 1	$pKS = 0$	$pK = 0$	$pS = 0$	$p0 = 0$	Hepsi 0
420,73	422,30	420,82	420,84	422,96	423,16

Başlangıçta seçilen parametre değerleri (ilk değerler) ile testler sonucunda tercih edilen değerler (son değerler) Tablo 6.15.'te verilmiştir.

Tablo 6.15. Parametrelerin İlk ve Son Değerleri (KKP)

Değişken	İlk Değerler	Son Değerler
SB	25	80
MTS	5000	6000
pKS	1.0	0.65
pK	1.0	0.3
pS	1.0	1.00
$p0$	1.0	1.00
λ	0.5	0.125
α	0.1001	1.001

Parametrelerin ilk ve son değerlerinin çözüm kalitesine etkisi Şekil 6.24.'te gösterilmiştir.



Şekil 6.24. Parametrelerin İlk ve Son Değerlerinin Karşılaştırması (KKP)

Karşılaştırma problemleri ile yapılan testlerin sonuçları Tablo 6.16.'da verilmiştir. Bu tabloda “|V|”, çizgenin düğüm sayısını, “|C|” optimal çözümünü göstermektedir. Süreler saniye cinsindendir. Koyu renkle gösterilen bölgeler PSO-A'nın performansının daha iyi olduğu yerlerdir. CKACS algoritmasının test sonuçları ile ilgili fazla detay verilmemesinin nedeni algoritmanın anlatıldığı makalede bu bilgilerin bulunmamasıdır. Ortalama değerler, bu değerleri kullanan [Richter et. al., 2007]'den alınmıştır.

Yapılan testler sonucunda 40 kenar kapsama problemi örneği içeren BHOSLIB karşılaştırma problemlerinin hepsinde PSO-A'nın çözüm kalitesinin CKACS'dan daha iyi olduğu görülmüştür.

Tablo 6.16. PSO-A'nın KKP'ye Uygulanışının Karşılaştırmalı Sonuçları

	V	C	PSO-A				CKACS
			En İyi	Ortalama	Std.	Süre	Ortalama
frb30-15-1	450	420	420	420,0	0,00	109	424.0
frb30-15-2			420	420,1	0,32	153	424.5
frb30-15-3			420	420,7	0,48	298	424.6
frb30-15-4			420	420,0	0,00	77	424.0
frb30-15-5			420	420,8	0,42	416	423.6
frb35-17-1	595	560	561	561,7	0,48	721	565.5
frb35-17-2			560	561,1	0,74	657	566.5
frb35-17-3			560	560,1	0,32	397	564.4
frb35-17-4			561	561,5	0,53	834	565.5
frb35-17-5			560	560,8	0,42	633	564.1
frb40-19-1	760	720	721	721,7	0,48	959	725.6
frb40-19-2			721	721,8	0,42	959	726.8
frb40-19-3			721	721,7	0,48	1028	727.6
frb40-19-4			721	721,7	0,48	981	726.1
frb40-19-5			721	721,8	0,42	937	725.3
frb45-21-1	945	900	902	902,5	0,53	1365	908.2
frb45-21-2			902	902,8	0,42	1452	908.5
frb45-21-3			902	902,7	0,48	1459	908.3
frb45-21-4			901	902,2	0,63	1436	908.4
frb45-21-5			901	902,1	0,57	1418	909.1
frb50-23-1	1150	1100	1102	1103,0	0,47	2202	1110.4
frb50-23-2			1103	1103,2	0,42	1965	1109.7
frb50-23-3			1102	1102,8	0,63	1964	1108.3
frb50-23-4			1102	1103,0	0,47	1923	1109.6
frb50-23-5			1102	1102,8	0,42	1917	1110.3
frb53-24-1	1272	1219	1221	1222,6	0,70	2124	1229.9
frb53-24-2			1221	1221,9	0,57	2072	1229.3
frb53-24-3			1221	1222,2	0,63	2096	1231.6
frb53-24-4			1222	1222,2	0,42	2175	1230.5
frb53-24-5			1221	1222,4	0,84	2276	1231.8
frb56-25-1	1400	1344	1347	1347,8	0,42	2470	1356.8
frb56-25-2			1348	1348,1	0,32	2762	1355.7
frb56-25-3			1347	1347,7	0,48	2239	1355.6
frb56-25-4			1347	1347,6	0,70	2297	1354.8
frb56-25-5			1346	1347,5	0,71	2179	1354.6
frb59-26-1	1534	1475	1478	1479,1	0,88	2909	1486.8
frb59-26-2			1478	1478,8	0,63	3108	1486.4
frb59-26-3			1478	1479,0	0,82	2626	1487.8
frb59-26-4			1479	1479,2	0,42	3060	1487.3
frb59-26-5			1478	1479,0	0,67	2886	1487.3

7. SONUÇLAR

Bu çalışma kapsamında alt küme tabanlı ayırık optimizasyon problemlerinin büyük bir bölümünü kapsayan büyüme ve küçülme amaçlayan alt küme problemlerinin tanımı yapılmış ve bu tip problemler üzerinde çalışacak bir PSO modeli geliştirilmiştir. Modelin sadece bu tipteki problemler üzerinde çalışmasının nedeni modelin iklendirme, bozma ve onarma aşamalarında sürekli olarak olurlu çözümler oluşturmalarının ve onarma aşaması tamamlandığında en iyi amaç fonksiyonu değerine sahip olmasının beklenmesidir.

Geliştirilen model iki büyüme ve bir küçülme amaçlayan problem için gerçekleştirilmiş ve bu problemler üzerinde test edilmiştir. Büyüme amaçlayan problemler maksimum klik problemi (MKP) ve çok boyutlu sırt çantası problemidir (ÇSP). Küçülme amaçlayan problem ise kenar kapsama problemidir (KKP).

Modelin belirtilen problemler üzerinde parametre analizleri yapılmıştır. Bu analizler, özellikle pKS , λ ve α parametrelerinin çözüm kalitesi, farklılaşma ve çökme zamanı gibi algoritmanın davranış özelliklerine büyük bir etkisi olduğunu göstermiştir.

Bu problemler üzerinde gerçekleştirilen algoritmalar parametre analizleri sonucunda bulunan uygun parametreler kullanılarak karşılaştırma problemleri ile test edilmiştir. Yapılan testler sonucunda modelin MKP ve KKP üzerindeki sonuçlarının çözüm kalitesi açısından diğer sürü tabanlı metasezgisellerden daha başarılı olduğu görülmüştür. ÇSP üzerinde yapılan testlerden ise hem parametre analizleri hem de çözüm kalitesi bakımından iyi sonuçlar alınamamıştır.

PSO-A'nın MKP ve KKP üzerinde başarı ile çalışırken, ÇSP üzerinde başarısız olmasının nedeni şöyle yorumlanabilir. Bu modelde iki ayrı iyi çözümden yararlanarak yeni ve iyi bir çözüm oluşturmak esastır. Bu çözümler kişisel ve sosyal tecrübelerdir. Her turda parçacıklar bir önceki çözümlerini belli bir oranda bozar ve bu iki çözümden yararlanarak onarırlar. Bu iki çözümün birleştirilmesi ile yeni bir

özümün oluşması modelin temelini oluşturan sezgisel bir fikirdir. Bu fikir bazı problemlerde başarılı olurken bazı problemlerde başarılı olmayabilir.

MKP ve KKP problemlerinde özümü oluşturan noktalar (düğümler) arasında ilişkiler (kenarlar) mevcuttur. SP’de ise noktalar (nesneler) arasında bir ilişki mevcut değildir. Her nesne birbirinden bağımsızdır. Noktaların birbiri ile ilişkili olmasının iki iyi özümün birleşerek iyi bir özüm oluşturmaları sezgiselinin başarılı olması konusunda etkili olduğu düşünülebilir. Bu nedenle model MKP ve KKP’de başarı ile çalışırken, SP’de başarısız olmuş olabilir.

PSO-A modelinin MKP problemi üzerinde uygulanması ile geliştirilen algoritmayı anlatan “Maksimum Klik Problemi için Yeni Bir Hibrid Evrimsel Algoritma” bildirisi yayınlanmıştır [Cebeci ve Sevilgen, 2012]. Benzer şekilde modelin KKP üzerindeki sonuçlarının yayınlanması düşünülmektedir.

Ayrıca PSO-A’nın SP üzerinde başarısız olmasının nedenleri daha derinlemesine irdelenerek, modelin hangi tip alt küme problemlerinde başarılı olabileceği daha detaylı olarak araştırılabilir.

KAYNAKLAR DİZİNİ

1. Beasley J. E., OR-Library, Multidimensional Knapsack Problem,
[http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapi
nfo.html](http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapi
nfo.html)
2. Cebeci S., E. Sevilgen, "Maksimum Klik Problemi için Yeni Bir Hibrid
Evrimsel Algoritma", *Inista*, 2012.
3. Chu P.C., Beasley J.E., "A Genetic Algorithm for the Multidimensional
Knapsack Problem"[J]. *Journal of Heuristic*, 1998.
4. Clerc M., *Particle Swarm Optimization*. ISTE Ltd, 2006.
5. Correa E.S., Freitas A.A., Johnson, C. G., "A new discrete particle swarm
algorithm applied to attribute selection in a bioinformatics data set",
*Proceedings of the 8th annual conference on Genetic and evolutionary
computation*, July 08-12, 2006.
6. Dorigo M., "Optimization, learning and natural algorithms", PhD thesis,
Politecnico di Milano, Italy, 1992.
7. Fenet S., Solnon C., Investigating ACO Capabilities for Solving the
Maximum Clique Problem, *Soumis à Journal of Heuristics*, Liris, 2004.
8. Fidanova S., "Evolutionary algorithm for multidimensional knapsack
problem", *PPSNVII-Workshop*, 2002.
9. Garey M.R., Johnson D.S., *Computers and Intractability - A Guide to the
Theory of NP-Completeness*, Freeman & Co., San Francisco, CA, 1979.
10. Geng X., Xu J., Xiao J., Pan L., "A simple simulated annealing algorithm for
the maximum clique problem", *Information Sciences*, Volume 177, Issue 22,
pg. 5064-5071, 2007.

11. Gilmour S., Dras M., "Kernelization as heuristic structure for the vertex coverproblem", In: *Proceedings of the Third Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS 2006)*, Brussels, Belgium, 2006.
12. Glover F., Kochenberger G.A., "Critical Event Tabu Search for Multidimensional Knapsack Problems", *Metaheuristics: The Theory and Applications*, pg: 407-427. Kluwer Academic Publisher, 1996.
13. Goldberg D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.
14. Holland J.H., *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.
15. Johnson D.S. and Trick M.A. (Eds.), "Clique, Coloring, and Satisfiability: Second DIMACS Implementation Challenge," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 26, American Mathematical Society, Providence, RI, 1996.
16. Karp R.M., "Reducibility among combinatorial problems", *Complexity of Computer Computations* (R.E. Miller and J.W. Thatcher, eds.), Plenum Press, 1972, pg. 85–103.
17. Kennedy J., Eberhart R., "Particle swarm optimization", *Proceedings of the 1995 IEEE International Conference on Neural Networks*, vol.4 Perth, Australia, 1995.
18. Khuri S., Baeck T., Heitkoetter J., "The Zero/One Multiple Knapsack Problem and Genetic Algorithms", *ACM Symposium on Applied Computing, SAC'94*, Phoenix, Arizona. sf. 188-193, 1994.
19. Kong M., Tian P., Kao Y., "A new ant colony optimization algorithm for the multidimensional knapsack problem", *Computers and Operations Research*, Volume 35, Issue 8, pg. 2672-2683, 2008.

20. Kotecha, K., Gambhava N., "A hybrid genetic algorithm for minimum vertex cover problem", *The First Indian International Conference on Artificial Intelligence*, pg. 904-913, 2003.
21. Liao C.J., Tseng C.T., Luarn P., "A discrete version of particle swarm optimization for flowshop scheduling problems", *Computers & Operations Research* Volume 34, Issue 10, October 2007.
22. Marchiori E., "A simple heuristic based genetic algorithm for maximum clique problem", in *Proc. ACM Symp. Appl. Comput.*, sf.366-373, 1998.
23. Marchiori E., "Genetic, iterated and multistart local search for the maximum clique problem", *Applications of Evolutionary Computing*, Springer, 2002.
24. Papadimitriou, Steiglitz, *Combinatorial Optimization Algorithms and Complexity*, Dover Publications, Inc., 31 East 2nd Street, Mineola, NY. 11501, 1998.
25. Puchinger J., Raidl G.R., Pferschy U., "The Multidimensional Knapsack Problem - Structure and Algorithms", *INFORMS Journal on Computing* Volume 22 Issue 2, pg. 250-265. Spring 2010.
26. Qian F., Ding R., "Simulated annealing for the 0/1 multidimensional knapsack problem," *Numerical Mathematics*, vol. 16, pg. 320-327, 2007.
27. Richter S., Helmert M., Gretton C., "A Stochastic Local Search Approach to Vertex Cover", *Advances in Artificial Intelligence, Lecture Notes in Computer Science*, Volume 4667/2007, pg. 412-426, 2007.
28. Solnon, C., Bridge, D. "An ant colony optimization meta-heuristic for subset selection problems" In N. Nédjah & L. Mourelle (Eds.), *System engineering using particle swarm optimization* (pg. 7–29). New York: Nova Science Publisher. 2006.

29. Soriano P., Gendreau M., "Tabu search algorithms for the maximum clique problem", *DIMACS Volume 26: Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge*, D. S. Johnson and M. A. Trick (Eds.), pg. 221-244, 1993.
30. Şevkli Z., Sevilgen F.E., Keleş Ç., "Particle swarm optimization for the orienteering problem", *Inista*, 2007.
31. Tasgetiren , Suganthan , Quan-Qe Pan, "A discrete particle swarm optimization algorithm for the generalized traveling salesman problem", *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, July 07-11, 2007.
32. Wang, Huang, Zhou, Pang, "Particle swarm optimization for traveling salesman problem", *Machine Learning and Cybernetics, 2003 International Conference on*, 2-5 Kasım 2003. pg. 1583 - 1585 Vol.3
33. Wang C., Zhang J., Yang J., Hu C., Liu J., "A Modified Particle Swarm Optimization Algorithm and its Applications for Solving Travelling Salesman Problem", *In Proceedings of the International Conference on Neural Networks and Brain, ICNN*, vol.2, sf. 689-694, 2005.
34. Xu K., Li W., "Exact phase transitions in random constraint satisfaction problems", *J. Ariff. Intell. Res. (JAIR)* 12:93-103, 2000.
35. Xu X., Ma J., "An efficient simulated annealing algorithm for the minimum vertex cover problem", *Neurocomputing - New Issues in Neurocomputing: 13th European Symposium on Artificial Neural Networks*, Volume 69, Issues 7-9, pg. 913-916, 2006.
36. Zhang Q., "Evolutionary algorithm with guided mutation for the maximum clique problem", *IEEE Trans. Evol. Comput.*, vol. 9, pg.192 , 2005.

ÖZGEÇMİŞ

1981 yılında İstanbul’da doğmuştur. İlk öğretimini 1988-1993 yılları arasında Nazmi Duhani İlkokulu, orta öğretimini 1993–1996 yılları arasında Altay Çeşme Ortaokulu, liseyi 1995-1999 yılları arasında Rezan Has Lisesinde tamamlamıştır. 1999 yılında girdiği Boğaziçi Üniversitesi Matematik Bölümü’nü 2006 yılında bitirmiştir. Yüksek lisans eğitimine GYTE Bilgisayar Mühendisliği bölümünde devam etmektedir, 2010 yılından beri aynı kurumda araştırma görevlisi olarak çalışmaktadır.