ÇOK AMAÇLI KARAR VERME PROBLEMLERİNDE GENETİK ALGORİTMA TEMELLİ ÇÖZÜM YÖNTEMLERİ

GENETIC ALGORITHMS APPROACHES TO MULTIPLE OBJECTIVE OPTIMIZATION PROBLEMS

FERSİN KESKİN

Hacettepe Üniversitesi

Lisansüstü Eğitim – Öğretim ve Sınav Yönetmeliğinin

İstatistik Anabilim Dalı İçin Öngördüğü

DOKTORA TEZİ

olarak hazırlanmıştır.

2009

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından **İSTATİSTİK ANABİLİM DALI 'nda DOKTORA TEZİ** olarak kabul edilmiştir.

Başkan	:….. Prof. Dr. Ayşen Apaydın
Üye	:….. Prof. Dr. Hasan Bal
Üye	:….. Prof. Dr. Süleyman Günay
Üye (Danışman)	:….. Prof. Dr. Gülsüm Hocaoğlu
Üye	:….. Doç. Dr. Meral Sucu

ONAY

Bu tez ...../...../..... tarihinde Enstitü Yönetim Kurulunca kabul edilmiştir.

Prof.Dr. Erdem Yazgan

Fen Bilimleri Enstitüsü Müdürü

ÇOK AMAÇLI KARAR VERME PROBLEMLERİNDE GENETİK ALGORİTMA TEMELLİ ÇÖZÜM YÖNTEMLERİ

Fersin Keskin

ÖZ

Bu çalışmanın amacı literatürde bulunan kısıtlı ve kısıtsız çok amaçlı karar verme problemlerinin çözümünde kullanılan genetik algoritma temelli yöntemleri incelemek ve her iki durum için yeni algoritmalar önermektir.

Bu amaçla çok amaçlı karar verme problemleri için kullanılan temel tanımlar ve bilgiler verilmiştir. Ayrıca çok amaçlı problemlerin çözümünde kullanılan klasik yöntemler, Genetik Algoritma ve bu algoritma temelli kısıtsız çok amaçlı karar verme problemleri için önerilen algoritmalar üzerinde durulmuştur. Kısıtlı durum için literatürde önerilen teknikler ve ceza fonksiyonları incelenmiştir.

Kısıtsız çok amaçlı karar verme problemleri için genetik algoritma temelli yöntem ve kısıtlı çok amaçlı karar verme problemleri için iki genetik algoritma temelli yöntem önerilmiştir. Önerilen algoritmalar literatürden alınan problemlere uygulanmış ve sonuçlar tartışılmıştır.

Anahtar Kelimeler: Çok amaçlı karar verme, genetik algoritmalar, kısıtlı ve kısıtsız optimizasyon, ceza fonksiyonu.

Danışman: Prof. Dr. Gülsüm Hocaoğlu, Hacettepe Üniversitesi, İstatistik Bölümü, Yöneylem Araştırması Anabilim Dalı.

GENETIC ALGORITHMS BASED APPROACHES TO MULTIPLE OBJECTIVE OPTIMIZATION PROBLEMS

Fersin Keskin

ABSTRACT

The objective of this study is to discuss genetic algorithms based methods to solve both constraint and non-constraint multi-objective optimization problems in literature and to propose new algorithms for both cases.

For this goal, definitions and some knowledge related to the multi-objective optimization are provided. In addition the basic genetic algorithm, some classical methods and genetic algorithm based multi objective algorithms are introduced. Also for the constrained case in multi-objective optimization, some techniques and types of penalty functions are discussed.

A genetic algorithm based method for non-constraint multi-objective problems and two genetic algorithm based methods for constraint cases are suggested. Eventually, offered algorithms are applied over problems which were taken from literature, and results are argued.

Keywords: Multi-objective decision making, genetic algorithms, constraint and non-constraint optimization, penalty functions.

Advisor: Prof. Dr. Gülsüm Hocaoğlu, Hacettepe University, Department of Statistics, Operations Research Section.

TEŞEKKÜR

Doktora çalışması süresince sağladığı katkı ve önerileri için danışmanım sayın Prof. Dr. Gülsüm Hocaoğlu'na,

Destek ve önerilerini esirgemeyen hocalarım sayın Prof. Dr. Süleyman Günay 'a ve sayın Doç. Dr. Meral Sucu'ya,

Bölüm içinde bana destek olan tüm değerli hocalarıma ve çalışma arkadaşlarıma,

Sağladıkları manevi destek ile çalışmama büyük katkıda bulunan aileme sonsuz teşekkür ederim.

İÇİNDEKİLER

SİMGELER ve KISALTMALAR DİZİNİ

GA	Genetik Algoritma
VEGA	Vektör Değerlendirmeli Genetik Algoritma (Vector Evaluated Genetic Algorithm,)
MOGA	Çok Amaçlı Genetik Algoritma (Multi-objective Genetic Algorithm)
NSGA	Sıralı Alt Edilmeyen Genetik Algoritma (Non-dominated Sorting Genetic Algorithm)
NPGA	Uygun Pareto Genetik Algoritma (Niched Pareto Genetic Algorithm)
NSGA II	Sıralı Seçkin Alt Edilemeyen Genetik Algoritma (Elitist Non-dominated Sorting Genetic Algorithm)
SPEA	Güçlü Pareto Evrimsel Algoritma (Strength Pareto Evolutionary Algorithm)
PAES	Arşivlenmiş Pareto Evrimsel Strateji (Pareto Archived Evolutionary Strategy)

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Günümüzde sistemlerin iç içe girmesi, disiplinler arasında ilişkilerin fazlalaşması ve ihtiyaçların artması karar verme problemlerinde tek amaç yerine daha fazla amacın düşünülmesi gereksinimini ortaya çıkarmıştır. Amaçlar çoğunlukla birbirleriyle çelişmektedir. Bir amaç iyileştiğinde diğer amaç veya amaçlar kötüleşebilir. Çok amaçlı karar verme problemlerinde çeşitli nedenlerle oluşan kısıtlar altında, amaç fonksiyon değerleri bakımından birbirine alternatif olabilecek en iyi çözümler bulunmaya çalışılır.

Çok amaçlı karar verme problemleri ve çözüm yöntemleri son 20 yılda büyük bir ilgi ile çalışılmaktadır. Literatürde hemen her disipline ait çok amaçlı karar verme problemi örneği görülmektedir. Sadece üretim, yönetim, işletme, pazarlama, ulaştırma, finans gibi uygulaması fazla olan konularda değil, temel bilimler olan kimya, matematik, istatistik ([1], [2]) gibi alanlarda da çok amaçlı optimizasyon problemleri modellenmiş ve çözülmüştür.

Çalışmanın İkinci Kesim'inde çok amaçlı karar verme problemleri hakkında temel bilgi ve tanımlar gözden geçirilmiş, klasik çözüm yöntemleri ve bu yöntemlerin uygulanamadığı durumda başvurulan Genetik Algoritma tabanlı yöntemler, son olarak kısıtlı çok amaçlı problemlerin çözümü için uygulanan yöntemler hakkında literatür incelemesi ve işleyiş biçimleri verilmiştir. Üçüncü Kesim'de kısıtsız çok amaçlı karar verme problemlerin çözümü için Genetik Algoritma temelli bir, kısıtlı problemler için iki algoritma önerilmiştir. Dördüncü Kesim'de, önerilen algoritmaların uygulamaları yapılarak sonuçları elde edilmiş ve yorumlanmıştır. Beşinci Kesim'de ise önerilen algoritmalar genel olarak değerlendirilmiş ve gelecek çalışmaların neler olabileceği tartışılmıştır.

2. TEMEL BİLGİ VE TANIMLAR

Karar verme problemleri tek ya da çok amaçlı olarak ikiye ayrılır. Tek amaçlı problemlerde, kısıtlar altında en iyi uygun çözümü bulmak temel hedeftir. Çok amaçlı olanlarda ise amaç fonksiyon değerleri bakımından birbirlerine üstünlük sağlayamayan birden çok uygun çözümün bulunması hedeflenmektedir. Tek amaçlı optimizasyon problemleri II. Dünya Savaşı sırasında çalışılmaya başlanmış ve bunlara çözüm bulmak için çeşitli algoritmalar geliştirilmiştir. Doğrusal Programlama problemlerinin çözümünde Simpleks Algoritmasından, Doğrusal Olmayan Programlama problemlerinin çözümünde Kuhn-Tucker teoreminden faydalanılarak geliştirilen ve en iyi çözümü garantileyen teknikler bulunmuştur. Bu tekniklerin uygulanmasında bazı varsayımların sağlanması gerekliliği ve uygulamada değişken sayısı arttıkça optimal çözüm bulunmanın çok zor hale gelmesi, hem tek hem de çok amaçlı problemlerin çözümünde sezgisel (heuristic) yöntemlerin yaygınlaşmasını sağlamıştır [3]. Sezgisel yöntemlere ait algoritmalarda işleyiş kuralına uyularak en iyi çözümler araştırılır. Hızla gelişen bilgisayar teknolojisi ile sezgisellerin problemlere uygulanması kolaylaşmış ve önerilen yöntem sayısı artmıştır. Bu yöntemlerin çoğu, üç ana sezgisel yöntemden geliştirilmiştir. Bunlar Genetik Algoritma, Tabu Araması ve Tavlama Benzetimi yöntemleridir. Genetik Algoritma, araştırmacıların en çok üzerinde durduğu ve uygulamalarda iyi sonuçlar veren yöntem olarak literatürde yerini almaktadır. Çok amaçlı karar verme problemlerinin çözümünde Genetik Algoritma temelli algoritmalar önerilmiştir. Bu algoritmalara ait bilgiler bu kesimde yer almaktadır.
İsmi geçen üç sezgisel yöntem dışında diğer sezgiseller de önerilmiştir. Sinir Ağları Yöntemi, Karınca Koloni (ant colony) Arama Yöntemi, Saçılım (scatter) Araması önemli yöntemlerdir [4].

Tek amaçlı problemler ile çok amaçlı problemler arasındaki temel fark; tek amaçlı problemlerde tek bir optimal çözüm ya da onun alternatifi olan optimal çözümler söz konusu iken, çok amaçlı problemlerde aynı anda her amacın en iyi değerini veren bir çözümün olmamasıdır. Bu problemlerde tüm amaçlar önemli olduğundan sadece bir amaca göre çözüm aramak yanlıştır. Amaçlar birbirleriyle çeliştiği için birindeki iyileşme diğer amaçlardan en az birinde kötüleşmeye yol açar. Çelişen amaçlar nedeniyle bulunan çözümler arasında amaç fonksiyon değerlerine

bakılarak ödünleşim miktarları elde edilir. Kabul edilebilir ödünleşim miktarları karar vericilere bağlıdır. Bu durumda çok amaçlı problemlerde genelde kesin olarak bir optimal çözüm yoktur. Elde edilen ve aralarında ödünleşim olan çözümlerden bazıları en iyi çözümler olarak karar vericiye sunulur. Bu çözüm kümesindeki her çözüm aşağıda verilen tanıma uygun çözümlerdir.

Tanım 1. Alt Eden (dominate) Çözüm: a, b $\in$ X çok amaçlı bir problemin iki ayrı çözüm vektörü olsun. Amaç fonksiyonları maksimizasyon tipinde olan bir problemde; her i için

$f_i(a) \geq f_i(b)$, $i \in (1,2,...,n)$

ve en az bir j için

$f_j(a) > f_j(b)$, $j \in \{1,2,...,n\}$

ise a çözümü b çözümünü alt eder ve $a \succ b$ şeklinde gösterilir [5].

Alt etme simetrik değildir. $a \succ b$ ise $b \succ a$ olamaz. Alt etme geçişli (transitive) özelliktedir. Örneğin a,b ve c farklı çözümler olsun. $a \succ b$ ve $b \succ c$ ise $a \succ c$ olur.

Alt etme zayıf ve güçlü alt etme olmak üzere ikiye ayrılır. Güçlü alt etme a çözümüne ait amaç fonksiyon değerlerinin hepsinin b çözümüne ait olanlardan daha iyi olması durumunda ortaya çıkar; diğer bir deyişle eşit olan amaç fonksiyon değerleri söz konusu değildir. Zayıf alt etme ise a çözümüne ait amaç fonksiyon değerlerinden en az biri b çözümünde kendine karşılık gelene eşit ve aynı zamanda yine a çözümüne ait amaç fonksiyon değerlerinden en az biri b çözümünde kendine karşılık gelenden daha iyi olması durumunda ortaya çıkar.

Tanım 2. Alt Edilemeyen (non-dominated) Çözüm: a, b $\in$ X çok amaçlı bir problemin iki ayrı çözüm vektörü olsun. Amaç fonksiyonları maksimizasyon tipinde olan bir problemde; en az bir i için

$f_i(a) < f_i(b)$, $i \in (1,2,...,n)$

ve en az bir j için

$f_j(a) > f_j(b)$, $j \in \{1,2,...,n\}$

ise, a ve b çözümleri alt edilemeyen çözümdür. Başka bir deyişle, çözümlere karşılık gelen amaç fonksiyonu değerlerinden bazıları a çözümünde, bazıları da b çözümünde daha iyidir [5],[6].

Bu tanımlardan sonra çok amaçlı karar verme problemlerinde karar vericilere söz konusu probleme ait çözümlerin oluşturduğu çözüm kümesi tanımı verilebilir.

Tanım 3. Pareto Optimal Küme: Çok amaçlı bir problemde elde edilen çözümler arasında diğer çözümleri alt eden, ancak birbirini alt edemeyen çözümlerin oluşturduğu kümeye Pareto optimal küme denir [5].

Pareto optimal olmayan çözümlerin her biri Pareto optimal çözümlerden en az biri tarafından alt edilir. Bu durumda Pareto optimal çözümler alt eden, Pareto optimal olmayanlar ise alt edilen çözümlerdir.

Çok amaçlı problemlerde çeşitli nedenlerle oluşan kısıtlar altında, amaç fonksiyonlarının en iyi değerlerini veren Pareto optimal çözüm kümesinin bulunması istenilir. Çok amaçlı karar verme problemlerinde amaçların tümü maksimum, tümü minimum ya da bir kısmı maksimum geri kalanları minimum olabilir. Problemin kısıtları da eşitlik ya da eşitsizlik biçiminde yazılabilir. Çok amaçlı bir optimizasyon probleminin matematiksel ifadesi, amaç fonksiyonları

minimum / maksimum $f_1(x), f_2(x), ..., f_M(x)$ ve

$g_i(x) \geq 0$, $i = 1,2,...,n$

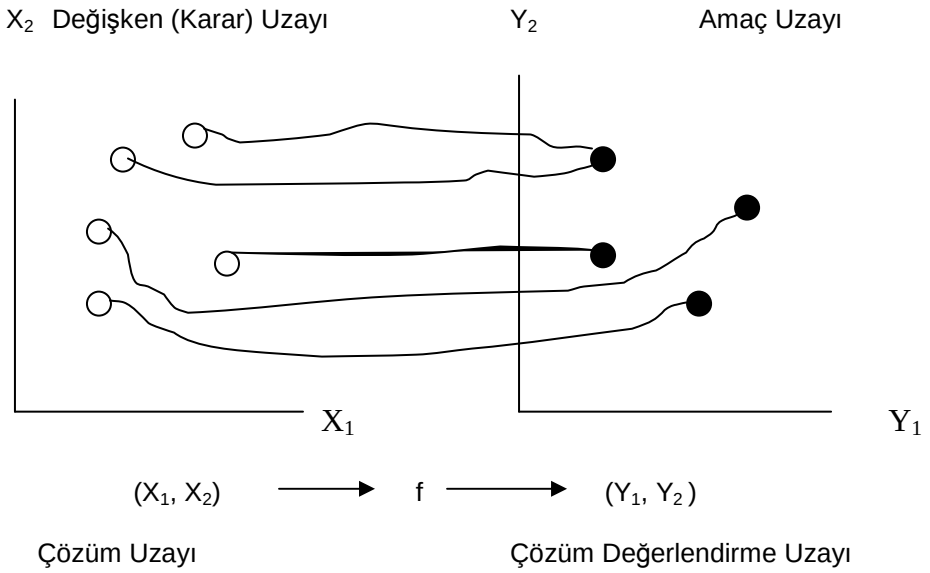
$h_j(x) = 0$, $j = 1,2,...,n$

şeklinde kısıtları göstermek üzere yazılabilir.

Amaç fonksiyonları arasında çelişkiler ve çözüm için araştırılan bölge, yani çözüm uzayının büyük ve karmaşık olması çok amaçlı problemlerin tipik özellikleridir.

Çok amaçlı bir optimizasyon probleminde eğer tüm amaç fonksiyonları ve kısıtlar doğrusal ise bu problem birçok amaçlı doğrusal programlama problemidir. Eğer amaç fonksiyonlarından ya da kısıtlardan en az biri doğrusal değilse, problem doğrusal olmayan çok amaçlı bir problemdir. Çok amaçlı problemlerin büyük çoğunluğu doğrusal değildir [5].

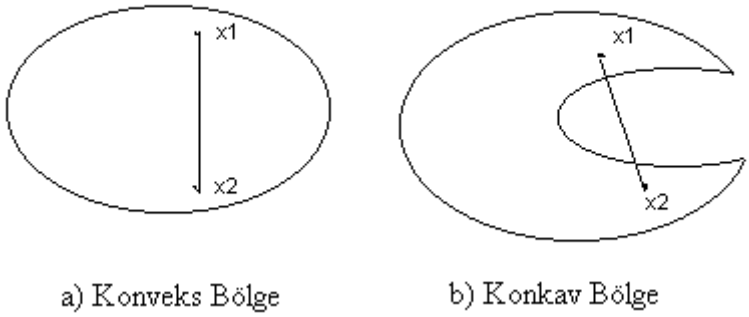
Çok amaçlı ve tek amaçlı problemlerde önemli bir fark araştırılan uzaylardır. Tek amaçlı problemlerde sadece karar uzayı varken, Şekil 1'de görüldüğü gibi çok amaçlı problemlerde hem karar uzayı hem de amaç uzayı vardır. Bu durumda üzerinde çalışılması gereken iki uzay vardır. Karar uzayındaki her noktaya amaç uzayında en az bir nokta karşılık gelmektedir.



Şekil 1. İki değişkenli iki amaçlı bir problemde karar ve amaç uzayları

Çok amaçlı karar verme problemlerinde amaç uzayı konveks ya da konkav olabilmektedir. Bu tip uzaylara basit bir örnek Şekil 2'de verilmiştir. Çok amaçlı bir problemin eğer tüm amaç fonksiyonları ve uygun çözüm bölgesi konveks ise problem çok amaçlı konveks bir problemdir. Bunu sağlamayan yani amaç fonksiyonları ve çözüm bölgesi konveks olmayan problemler ise çok amaçlı konkav problemlerdir. Hem konveks hem de konkav problemlerin çözümü zor olabilir ama konkav problemlerin çözümünde daha dikkatli olunmalıdır. Bunun sebebi amaç uzayının parçalı ya da birbirinden ayrık olabilmesidir. Çözüm arama sırasında sadece bir bölgede takılı kalmamak, diğer bölgelerden de araştırmalar yapmak önemlidir. Bu yapılırken kendisinden yararlanılacak algoritma uygun olmayan çözümler bulabilir. Uygun olmayan çözümlerden yararlanmak, aslında başka bölgelere sıçranmasını ve belki de daha iyi sonuçların bulunmasını sağlayabilecektir. Çok fazla uygun olmayan çözüm ile ilgilenmek ise çözüm arayışında gereksiz oyalanmaya yol açabilir. Bu sebeple uygun olmayan

çözümlerden yararlanma kısıtlı yapılmalıdır. Özetle, konveks problemlere göre konkav problem çözümlerini araştırmak zordur. Hem karar hem de amaç alanlarının ikisinin ya da en azından birinin konkav olması probleme optimal çözüm bulmayı güçleştirir [4].



Şekil 2. Konveks ve konkav bölgeler.

Çok amaçlı problemlerinin çoğu için optimal çözüm bulmak çok zordur ve gelişmiş bilgisayarlara rağmen zaman alan bir iştir. Bunun sebebi çok amaçlı problemlerin çoğunun NP-zor (NP-hard) olması yani; en iyi çözüm kümesini bulmak için kullandığımız algoritmanın üreteceği çözüm sayısının, problemin büyüklüğünün artması ile aşırı olarak fazlalaşması ve bir sınırlamanın yapılamamasıdır. Böylece Pareto optimal küme makul bir zamanda bulunamaz. Çözüm zamanı problem büyüklüğünün üstel bir fonksiyonu olarak artar. Örneğin çok amaçlı gezgin satıcı probleminde her şehre yalnızca bir kez uğranmak koşulu ile gidilen toplam uzaklığın minimum yapılması istenirken, satıcının gezdiği yerlere vardığı zamana göre değişen kar miktarları da maksimum yapılmak istenir. 10 şehir için yaklaşık 181,000 çözüm sayısı olacak ve bu bilgisayarda 3 dakikada çözülebilecektir. 20 şehir olduğunda ise 10^{16} tane çözüm olacak ve bilgisayar ile 320,000 yılda çözülebilecektir. Şehir sayısındaki % 50 artış, çözüm sayısını yaklaşık 55 milyar kat arttırmıştır ve 20 şehir için bile problemin çözümü imkânsız hale gelmiştir. Bunun sebebi çözüm sayısının, n şehir sayısı olmak üzere $n(n-1)!/2$ ile üstel olarak büyümesidir. Buna göre çok amaçlı gezgin satıcı probleminin NP-hard olduğu ve en iyi çözüm kümesini bulmanın çok zor olduğu görülmektedir [4]. Bu tip problemlerde optimal çözüm kümesi tam olarak elde edilememesine rağmen, bu çözüm kümesine yakınsayan başka çözümlerin oluşturacağı bir kümeye ulaşmak imkânsız değildir. Bu kümedeki çözümlere kabul edilebilir bir zamanda

ulaşılabiliyorsa, elde edilen yaklaşık optimal çözümler karar vericiye problemin çözümü olarak verilebilir. Böylece problemin optimal çözümü karar vericinin kabul edebileceği amaç fonksiyon değerlerinin bulunması ile sağlanır. Yaklaşık optimal kümenin bulunması için kullanılabilecek yöntemler sezgisel yöntemlerdir [3],[4],[5].

Çok amaçlı problemlerin çözümünde elde edilecek çözümlerde bazı özelliklere dikkat edilmelidir. Elde edilecek son çözüm kümesinin gerçek Pareto optimal kümeye yakınsaması ya da karar vericinin kabul edebileceği çözümlere ulaşması istenir. Bunun olması için çözüm algoritmasının, çözüm arama sırasında uyguladığı seçme ve seçtiği çözümlerin değerlendirilme şekli büyük önem taşımaktadır.

Çok amaçlı karar verme problemlerinde bazı çözümler özel isimler almaktadır. N tane amacı olan çok amaçlı bir optimizasyon probleminde her amaç fonksiyonu verilen kısıtlar altında çözüldüğünde elde edilen amaç fonksiyon değerlerinin oluşturduğu vektöre “ideal çözüm vektörü” denir. Bu vektör genelde ütopik bir çözüm olup problem için uygun olmayan bir çözümdür. İdeal çözüm vektörü amaçların hepsi maksimum ya da minimum olduğu durumlarda uygun çözüm olabilir. Genelde uygun olmamasına rağmen ideal çözüm vektörü çözüm algoritmasının başlangıç noktası olarak alınabilir [5], [7].

Nadir amaç vektörü (nadir objective vector), amaç fonksiyonlarının tüm Pareto optimal küme içinde üst sınırını verir. Pratikte hesaplanması ya da belirlenmesi zordur. Ancak ödemeler (payoff) tablosu oluşturularak ve ideal çözüm vektöründen de yararlanılarak nadir amaç vektörü elde edilebilir. İdeal ve nadir amaç vektörleri her bir amaç fonksiyonunun Pareto optimal bölgede normalleştirilmesi için kullanılmaktadır. Normalleştirme işlemi f_i pareto optimal bölgedeki amaç fonksiyon değeri olmak üzere

$$f_i^{norm} = \frac{f_i - f_i^{ideal}}{f_i^{nadir} - f_i^{ideal}}$$

formülüyle yapılır [8] .

2.1. Çok Amaçlı Problemlerin Çözümünde Kullanılan Klasik Yöntemler

Bu kesimde çok amaçlı optimizasyon problemlerinde çözüm bulmak amacıyla başvurulan yöntemler verilmiştir. Çözümü aranan çok amaçlı karar verme probleminin biçimi uygunsa bu yöntemler etkilidir ve eğer problemin en iyi çözümü mevcutsa bu çözümü bulabilir.

2.1.1. Amaçların Ağırlıklandırılması

Bu yöntemde amaçlara ağırlıklar verilerek toplanır ve ağırlıklı tek bir amaç fonksiyonu elde edilir. Ağırlıklar karar vericinin amaçlara verdiği öneme göre belirlenir. Ağırlıklı amaçların oluşturduğu fonksiyon kullanılarak optimal sonuç bulunabilir. Yöntemin uygulanması sırasında problem hakkında çok fazla bilgi yoksa ağırlıkların belirlenmesi zor olabilir. Amaçlara ağırlıklar verilmesiyle çok amaçlı problem

$$\text{Minimum } Z(x) = \sum_{i=1}^{N} w_i f_i(x) \ ;$$

$$g_i(x) \le 0 \ , \ i = 1,2,\ldots,I$$

$$h_j(x) = 0 \ , \ j = 1,2,\ldots,J \qquad (2.1)$$

$$x_m \in X \ , \ m = 1,2,\ldots,M$$

şeklinde yazılabilir. Ağırlıklar $w_i \in [0,1]$'dir. Uygulamada genelde $\sum_{i=1}^{N} w_i = 1$ olur. Eğer problem konveks ve ağırlıklar pozitif ise Pareto optimal çözüm bulunur [8]. Bunun dışındaki durumlarda, yani; problemin konkav olması ya da ağırlıkların pozitif olmaması durumunda, bu yöntem ile çözüm aramak yanlış olmaktadır.

2.1.2. Є-Kısıt Yöntemi

Є kısıt yöntemi özellikle ağırlıklı yöntemin çözemediği konveks olmayan problemler için kullanılmaktadır. Haimes vd. [9] amaç fonksiyonlarından birinin amaçta kalması ve diğerlerinin $Є_i$ üst sınırlı kısıtlar olarak alınması ile çok amaçlı problemlerin yeniden formüle edilip çözülebileceğini göstermiştir. Buna göre formülasyon

Minimum $f_N(x)$

$f_i(x) \le \varepsilon_i$, i = 1,2,…,N-1

$g_j(x) \le 0$, j = 1,2,…,J

$h_k(x) = 0$, k = 1,2,…,K (2.2)

$x_m \in X$, m = 1,2,…,M

şeklinde yazılabilir.

ϵ_i terimleri amaç fonksiyonları için üst sınırdır ve sıfırdan farklı küçük sayılar olmak zorunda değildir. Üst sınırları belirleme önemlidir. Miettinen [8] tarafından ispat edilen teoreme göre verilen herhangi (ϵ_1, ϵ_2 ,…,ϵ_{N-1}) üst sınır vektörüne göre bulunan tek çözüm, ilgilenilen ϵ-kısıt problemine ait Pareto optimal çözümdür. Buna göre problemin çözümü üst sınırlara bağlıdır. Bu sınırların konulması için problem hakkında çok iyi bilgi edinilmelidir. Sınırlardan biri olması gerektiğinden küçük alınırsa problem için uygun çözüm bile bulunamayabilir. Farklı ϵ değerleri ile çeşitli Pareto optimal çözümler elde edilebilir. Çok amaçlı optimizasyon problemlerinin büyük çoğunluğunda olduğu gibi çözüm bölgesi konveks olmayan problemlerde, ϵ kısıtlamasıyla global optimal noktanın olduğu bölge kesilip atılabilir. Böylece en iyi çözüm hiçbir zaman bulunamaz [5].

2.1.3. Ağırlıklı Metrik Yöntemi

Amaçların ağırlıklandırılması yönteminde olduğu gibi, ağırlıklı metrik yönteminde de amaçlar tek fonksiyona indirgenir. Problem;

Minimum $$l_p(x) = \left(\sum_{i=1}^{N} w_i \left| f_i(x) - f_i^{ideal} \right|^p \right)^{1/p}$$

$g_i(x) \le 0$, i = 1,2,…,I

$h_j(x) = 0$, j = 1,2,…,J (2.3)

$x_m \in X$, m = 1,2,…,M

şeklinde yazılabilir [3]. Bu formülasyonda $1 \le p \le \infty$ şeklindedir. Eğer p = 1 olursa yöntem ağırlıklı yönteme dönüşür. p = 2 olursa, amaç fonksiyonu ideal vektörden, ağırlıklı Öklit uzaklığın minimize edilmesi şekline döndüğü görülür. p = ∞

olduğunda ise problem en büyük $\left|f_i - f_i^{ideal}\right|$ ifadesinin minimize edilmesine dönüşür. Bu durumda elde edilen problem ağırlıklı Tchebycheff problemidir. Eğer ağırlıklar pozitif ise Pareto optimal çözüm bulunabilir. Yöntem ideal çözümlere gerek duyduğundan probleme başlamadan önce her amaç fonksiyonu tek başına düşünülerek optimal değerleri elde edilmelidir. Ayrıca amaç fonksiyon değerlerinin de normalleştirilmesi önerilmektedir [8].

2.1.4. Hedef Programlama

İlk olarak Charnes vd. [10] tarafından tanıtılmış ancak yoğun kullanımı Ignizio [11] ve Lee [12]'nin çalışmalarından sonra başlamıştır. Hedef programlamada ana fikir her amaç için önceden belirlenen hedeflere ulaşmaktır. Eğer bu mümkün değil ise bu hedeflerden sapmalar minimize edilmelidir. Dört değişik hedef çeşidi vardır. Bunlar küçük eşit $(f(x) \le t)$, büyük eşit $(f(x) \ge t)$, eşit $(f(x) = t)$ ve aralık $(f(x) \in [t^l, t^u])$ şeklinde tanımlanır. Bu hedeflere ulaşmak için fark değişkenleri kullanılmaktadır. Problemin amaç fonksiyonu fark değişkenlerinin toplamını minimum yapacak biçimde kurulur. Hedef programlama fark değişkenlerinin amaç fonksiyonunda kullanım şekline göre çeşitli gruplara ayrılır. Bunlardan en çok kullanılanlar ağırlıklı, ardışık (lexicographic) ve min-maks hedef programlamadır [13]. Ağırlıklı hedef programlama problemi matematiksel olarak;

$$\text{Minimum} \quad \sum_{i=1}^{N} (w_i d_i^+ + v_i d_i^-)$$

$$f_i(x) - d_i^+ + d_i^- = t_i \qquad (2.4)$$

$$d_i^+, d_i^- \ge 0 \ , \ i = 1,2,\ldots,N$$

$$x_m \in X, \ m = 1,2,\ldots,M$$

şeklinde yazılabilir. d_i^+ değişkenleri hedeflerden pozitif sapmaları, d_i^- değişkenleri ise negatif sapmaları göstermektedir. w_i ve v_i değişkenleri ise bunlara verilen ağırlıklardır [13] .

Diğer hedef programlama çeşidi ardışık hedef programlamadır. Burada hedefler arasında öncelikler varsa kullanılır. Önceliği olan hedef ile kısıtlar düşünülerek probleme çözüm bulunur, diğer hedefler sonradan eklenir. Buna göre birbirini

izleyen hedef programlama problemleri çözülür ve tek çözüme ulaşıncaya kadar devam edilir. Genelde bir tek Pareto optimal çözüme ulaşılır.

Sık kullanılan hedef programlama yöntemlerinden biri Min-Maks hedef programlamadır. Bu yöntem ağırlıklı hedef programlama yaklaşımına benzemektedir. Ancak burada hedeflerden ağırlıklı sapmaların toplamını minimize etmek yerine, en büyük olan sapma minimum yapılır. Buna göre problem

$$\begin{aligned} &\text{minimum} \quad d \\ &w_i d_i^+ + v_i d_i^- \le d \\ &f_i(x) - d_i^+ + d_i^- = t_i, \quad i = 1,2,\ldots,N \\ &d_i^+, d_i^- \ge 0 \\ &x_m \in X, \quad m = 1,2,\ldots,M \end{aligned} \tag{2.5}$$

şeklinde ifade edilir. w_i ve v_i değişkenleri ağırlıklardır. d hedefler arasındaki en büyük sapmayı, t_i değerleri ise hedefleri göstermektedir [13] .

Hedef programlamada karar verici amaçlara ulaşmak için birçok hedef tanımlamak zorundadır. Bu hedeflerin matematiksel olarak ifade edilmesi gerekmektedir. Problemlerde amacın hedefe dönüşümü zordur ya da tam olarak belirlenemeyebilir. Eğer istenilen hedefler tam olarak belirlenirse ve bunlar uygun çözüm bölgelerinde ise hedef programlama etkili bir yöntemdir. Ancak belirlenen amaçlar doğrusal ve arama uzayı konveks değilse hedef programlama yöntemleri optimal çözüm ya da çözümleri bulmada yetersiz kalabilir [5],[13].

2.2. Klasik Yöntemlerin Yetersiz Kaldığı Çok Amaçlı Karar Verme Problemleri ve Genetik Algoritma

Klasik yöntemler konveks çok amaçlı karar verme problemlerde ve amaçların tek amaca indirgenmesinde bir sakınca olmayan problemlerde en iyi çözümü garanti etmektedir. Bu özelliği ile bu yöntemlerin problemlere uygulanmaları günümüzde devam etmektedir. Hatta Є-Kısıt Yöntemi ve Ağırlıklı Tchebycheff Yöntemi kesin olmamakla birlikte konkav problemlerde de global optimal çözümü bulabilmektedir [4],[5],[14].

Çok amaçlı karar verme problemlerinin çoğu için Pareto optimal çözüm kümesine ulaşmak klasik yöntemlerle hemen hemen imkânsızdır. Bunun sebebi çok amaçlı problemlerinin çoğu NP-zor özelliği göstermesidir [3]. Değişkenlerin birkaçı ve hatta bazen hepsi sadece tamsayılı değerler alabilir. Bu tip kesikli değişkenli problemlerin çözüm uzayı dahi belirlenemeyebilir. Tamsayılı problemler genelde çok karışık ve konkav arama uzayına sahiptir ve en iyi çözüm kümesini bulmak zordur. Ayrıca çok sayıda matematik ve mühendislik uygulamalarında karşılaşılan çok amaçlı problemler NP-zor problemlerdir ve NP-zor sınıfı problemlerinin çözümü için kullanılan geçerli bir yöntem yoktur. Bunlar için optimal çözümleri araştırmak ancak sezgisel yöntemlerle olabilmektedir [4].

Klasik yöntemlerin Pareto optimal çözümlere yakınsaması çok yavaştır. Çok sayıda değişkene sahip büyük problemlerde uygulamaları sıkıntı yaratabilmektedir. Ayrıca klasik yöntemlerin özel problemler için geliştirildikleri görülmektedir. Örneğin Fletcher Reeves Algoritması (conjugate gradient) karesel (quadratic) amaç fonksiyonlarına sahip olan problemlerde optimal çözüm bulmak için kullanılmaktadır [15].

Bu tezde sezgisel algoritmalar arasında önemli yer tutan Genetik Algoritmanın NP-zor sınıfındaki çok amaçlı problemlere uygulanması incelenecektir. Bu amaçla, Genetik Algoritma ile ilgili temel bilgiler kısa olarak bu kesimde verilmiştir. Daha çok bilgi için verilen referanslara başvurulabilir.

İlk olarak 1973 yılında John Holland tarafından düşünülen genetik algoritma (GA) çeşitli bilimsel araştırma problemlerinde optimizasyon aracı olarak yoğun biçimde kullanılmaktadır. Holland'dan sonra GA çalışmaları hızlanmıştır. Bu konuda ilk çalışmalar uluslararası konferanslar sonunda basılan yayın kataloglarında çıkmış ve daha sonra Genetik Algoritma konusunda kitaplar yayımlanmıştır. En çok başvurulan kaynak kitaplar Holland [16], Goldberg [17], Michalewicz [18], Gen ve Cheng [19], Mitchell [20] ve Vose [21]'dur. En kapsamlı referans kaynak ise Back, Fogel ve Michalewicz tarafından yazılan "Handbook on Evolutionary Computation" adlı kitaptır. Ayrıca GA'dan faydalanılarak yapılan bilimsel araştırmalar günümüzde çok sayıda bilimsel makalede karşımıza çıkmaktadır [6].

GA, genetik ile ilgili mekanizmaların algoritma olarak optimizasyon problemlerine uygulanmasıdır. Genetikte yeniden oluşturma (reproduction), mutasyon (mutation) ve çaprazlama (crossover) ile olur. Bunlar GA operatörleridir. Elde edilen birey (çözüm), kalıtımını devam ettirebilmesi için iyi bir uyum değerine (fitness value) sahip olmalıdır. Böylece güçlü bireyler hayatta kalmayı başarırlar ve gelecek kuşaklara kendi genlerini aktarabilirler. Buna göre iyi sonuç veren çözümlere ait bilgiler, sonraki çözümlere aktarılarak daha iyi çözümlere ulaşılabilir [22].

Genetik Algoritmaların genel işleyiş şekli aşağıda özetlenmiştir;

1. Rasgele seçilmiş çözümler ile N büyüklükteki başlangıç popülasyonu oluşturulur.
2. GA operatörlerinin uygulanacağı uyumu büyük olan çözümler seçilir.
3. Seçilen çözümler eşlenerek, çaprazlama ve mutasyon operatörleri ile yeni popülasyon oluşturulur.
4. Yeni popülasyonu oluşturan çözümlerin uyumları belirlenir.
5. Durdurma kriterine ulaşılmamışsa 2.adıma dönülür. Durdurma kriteri Algoritmanın önceden belirlenmiş kuşak sayısı kadar devam etmesi ya da belli sayıda kuşak içinde daha iyi çözüm bulunamaması vb. olabilir.
6. Algoritmanın durması ile o zamana kadar bulunan çözümlerden en iyisi problemin çözümü olarak değerlendirilir [3].

GA'nın bilgisayarda uygulanabilmesi için, problemlere en uygun kodlama şekli kullanılmalıdır. İkilik düzende (binary) kodlama, permütasyon kodlaması ve reel sayı kodlaması en çok kullanılanlardır.

Kendinden sonraki kuşakta kullanılacak çözümlerin seçimi, GA'nın işleyişinde önemli adımlardan biridir. Araştırmalar sonucunda çeşitli seçim yöntemleri önerilmiştir. Uyuma orantılı seçim [17], rulet çemberi seçimi [17], sıralama seçim [23], turnuva seçimi [24] ve yerine koymadan seçim [17] uygulamada sıklıkla kullanılan seçim yöntemleridir. Bu yöntemlerin ortak noktası, elde edilen çözümlerin uyum değerlerine göre olasılıksal olarak seçilmesidir. İyi uyum değerine sahip olan çözümlerin gelecek kuşakta olma olasılığı her zaman yüksektir.

Seçilen çözümlerden yararlanarak yeni çözümler elde etmek için, GA operatörleri olan çaprazlama ve mutasyondan faydalanılır. Çaprazlama, mutasyon operatörüne göre daha sık kullanılır. Tek nokta, iki nokta, çok nokta ve tekbiçimli çaprazlama teknikleri vardır. Bunlardan en çok tek nokta ve iki nokta çaprazlamanın kullanıldığı görülmektedir. Mutasyon ise çaprazlamadan sonra elde edilen çözümlere uygulanan bir operatördür. Önceden belirlenen küçük bir mutasyon olasılığı ile çözümlere uygulanır ve bunlardan değişik çözümler elde edilir. Mutasyon operatörü gelecek kuşakların oluşmasında çeşitlilik sağlar. Çeşitleri ve olasılıkları önceden belirlendiği için kontrol parametreleri olan çaprazlama ve mutasyon ile iyi çözümlere ulaşılabileceği gibi kötü çözümler de bulunabilir; ama bu çözümler seçim aşamasında elenirler. GA'da çaprazlama ve mutasyon kontrol parametrelerine ek olarak, popülasyon genişliğine de karar verilmelidir. Böylece Genetik Algoritma'da üç kontrol parametresi mevcuttur. Problemlerin çoğu bu parametrelere çok duyarlıdır ve iyi çözüm bulma bunların doğru seçilmesi ile mümkün olmaktadır [3] .

2.3. Çok Amaçlı Genetik Algoritma Tabanlı Sezgisel Yöntemler

Çok amaçlı optimizasyon problemlerin çözümü için uygulanan, Genetik Algoritma tabanlı ve seçkin çözümlerin izlenmediği sezgisel yöntemler bu kesimde verilmiştir. Bu algoritmalar çok kullanılan ve uygulamalarda en iyi sonuçlar veren temel yaklaşımlardır [5].

2.3.1. Vektör Değerlendirmeli Genetik Algoritma

Schaffer [25] tarafından geliştirilmiş olan Vektör Değerlendirmeli Genetik Algoritma (Vector Evaluated Genetic Algorithm, "VEGA") alt edilemeyen çözümleri bularak çalışan ilk algoritmadır. Bu algoritmada popülasyon her nesilde amaç fonksiyon sayısı kadar alt popülasyonlara bölünür ve her alt popülasyona rastgele çözümler atanır. İlk alt popülasyonda bulunan çözümlere uyum değeri olarak ilk amaç fonksiyon değerleri verilir. Bu şekilde devam edilerek k tane amaç fonksiyonu varsa, son alt küme olan k. alt kümeye en son amaç fonksiyonunun değerleri uyum değeri olarak verilir ve k tane alt popülasyon elde edilir. N ana popülasyondaki çözüm sayısı olmak üzere, her alt popülasyonda N/k tane çözüm

olur. Elde edilen k grubun her birine diğer gruplardan bağımsız olarak, seçildiği amaç fonksiyonunun değerine göre uyuma orantılı seçim yöntemi uygulanır. Bu yöntemde alt popülasyona ait çözümlerin amaç fonksiyon değerleri ortalama amaç fonksiyon değerine bölünerek üreme havuzuna çözümlerden kaç tane girmesi gerektiği belirlenir. Bu işlemden sonra havuzdaki çözümlerden rastgele seçim ile eşleştirme yapılır. Bu çiftlere GA operatörleri uygulanarak yeni çözümler elde edilir ve bu çözümler karıştırılarak algoritmanın başına dönülür. Aynı işlemler yeni popülasyona uygulanarak devam edilir. Durdurma kriterine kadar algoritma sürdürülür.

VEGA yöntemi basit ve klasik Genetik Algoritma için gerekli olan hesaplama yükü dışında ek yük getirmeyen bir yöntemdir. Ancak alt popülasyonların sadece bir amaç fonksiyonuna göre seçilmesi ile çözümler diğer amaçlara göre test edilememektedir. Bunun sonucunda çeşitlilik sağlanamamakta ve algoritma çoğu kez sadece bir amaç için çok iyi olan çözümlere yönelmektedir [17].

2.3.2. Çok Amaçlı Genetik Algoritma

Çok Amaçlı Genetik Algoritma (Multi-objective Genetic Algorithm "MOGA") Fonseca ve Fleming [26] tarafından geliştirilmiştir. Bu algoritmada popülasyonda bulunan her çözümün, diğer çözümleri alt edebilme durumuna bakılır. n_i, i.çözümü alt eden çözüm sayısı olmak üzere, her çözüme $r_i = 1 + n_i$ şeklinde bir sıralama (rank) sayısı verilir. Buna göre alt edilmeyen çözümlerin sıralama sayısı 1 olur. Elde edilen sıralama sayılarına göre çözümler gruplanır. İlk grupta alt edilemeyen çözümler, ikincisinde bir kez alt edilen çözümler olmak üzere devam edilerek gruplar oluşturulur.

$$F_i = N - \sum_{k=1}^{r_i} n(k) - 0{,}5\ [n(i) - 1] \qquad (2.6)$$

formülü ile her çözüm için uyum fonksiyon değerleri hesaplanır. Bu formülde n(i), i sıra sayısına sahip çözüm sayısını göstermektedir. N popülasyon büyüklüğüdür.
Sonraki adımda, her grupta çözümlerin sadece kendi gruplarında bulunan çözümlerden uzaklıkları

$$d_{ij} = \sqrt{\sum_{k=1}^{M} \left(\frac{f_k^{(i)} - f_k^{(j)}}{f_k^{max} - f_k^{min}} \right)^2} \qquad (2.7)$$

formülü ile hesaplanır. Burada

$f_k^{(i)}$ = k. amaç fonksiyonunun i. çözümde aldığı değer,

$f_k^{(j)}$ = k. amaç fonksiyonunun j. çözümde aldığı değer,

$f_k^{(max)}$ = k. amaç fonksiyonunun maksimum değeri ve

$f_k^{(min)}$ = k. amaç fonksiyonunun minimum değeridir.

Uzaklıklar hesaplandıktan sonra probleme bağlı olarak küçük bir sayı olması gereken ve uygunluk (niche) parametresi olarak bilinen σ_{share} belirlenir. Goldberg ve Richardson [27] tarafından geliştirilen paylaşım fonksiyonu

$$Sh(d_{ij}) = \begin{cases} 1 - \dfrac{d_{ij}}{\sigma_{share}} & , \quad d_{ij} \le \sigma_{share} \\ 0 & , \quad \text{öteki durumlar} \end{cases} \qquad (2.8)$$

kullanılarak her grupta bulunan iki çözüm arasındaki paylaşım değerleri hesaplanır. Paylaşım fonksiyonu, 0 ile 1 arasında değer alabilmektedir. 1 değerini alması demek, çözümün aynı çözüm olması; yani uzaklığın sıfır olmasıdır. Eğer $d_{ij} \ge \sigma_{share}$ ise, fonksiyon sıfır değerini alır. Bunun anlamı, iki çözüm birbirinden σ_{share} parametresinden daha fazla uzaktadır ve iki çözüm birbirinden farklıdır. $d_{ij} \le \sigma_{share}$ olduğunda, iki çözüm birbirine σ_{share} parametresi kadar ya da daha yakındadır. Dolayısıyla iki çözüm birbirine önemli derecede yakındır.

Paylaşım fonksiyon değerleri kullanılarak çözümlere karşılık gelen uygunluk sayılarına

$$nc_i = \sum_{j=1}^{n(i)} Sh(d_{ij}) \qquad (2.9)$$

formülüyle ulaşılır. Çözümler için paylaşım uyum değeri

$$F_j^{'} = F_j / nc_j \tag{2.10}$$

formülüyle hesaplanır. Son olarak ölçekli paylaşım uyum değerleri

$$\frac{F_j X n(i)}{\sum_{k=1}^{n(i)} F_k^{'}} x F_j^{'} \tag{2.11}$$

şeklinde elde edilir. Ölçekli paylaşım uyum değerleri kullanılarak stokastik evrensel seçim [23] yöntemi ile eşleştirme yapılarak çözümler belirlenir. Tek nokta çaprazlama ile gen takası yapılır ve mutasyon uygulanır. Bu şekilde durdurma kriterine kadar algoritma devam eder.

MOGA yönteminde çözümlerin uyum değerlerinin hesaplanması kolaydır. Ayrıca uygunluk parametresi ile çok sayıda optimizasyon problemine uygulanabilir. Ancak bu teknikte aynı tarafta (front) bulunan farklı çözümlere aynı uyum değerinin verilmesi algoritmanın yanlış tarafta çözümler aramasına neden olabilir. Uygulamalarda, MOGA'nın çalışılan çözüm alanı şekline çok duyarlı olduğu ortaya konulmuştur. Ayrıca algoritma bir çözümün kendisinden kötü bir çözümden her zaman daha iyi uyuma sahip olmasını garantilememektedir [28].

2.3.3. Sıralı Alt Edilmeyen Genetik Algoritma

Sıralı Alt Edilmeyen Genetik Algoritma (Non-dominated Sorting Genetic Algorithm, "NSGA") Srivas ve Deb [29] tarafından geliştirilmiştir. Yöntem rastgele elde edilen çözümlerin alt edilip edilmeme durumlarına göre sıralanmasıyla başlar. P_1 popülasyonuna alt edilemeyen çözümler alınır. P_2'ye P_1'deki çözümler çıkartıldığında kalan çözümler arasında alt edilemeyen çözümler alınır ve bu şekilde devam edilerek tüm popülasyondaki çözümler alt popülasyonlara ayrılır. Her alt popülasyonda bulunan q tane çözüme, $q \in P_j$ olmak üzere $F_j^{(q)} = F_{min} - \varepsilon$ formülü ile uyum değerleri atanır. F_{min} ve ε kullanıcı tarafından belirlenen parametrelerdir. ε küçük bir sayı ve N popülasyon genişliği olmak üzere $F_{min} = N + \varepsilon$ olarak alınır. Burada önemli olan P_1'de bulunan alt edilemeyen en iyi çözümlere büyük uyum değeri vermek ve ondan sonra gelen P_2 çözümlerine daha küçük uyum değeri vermektir. Bu ise F_{min} parametresinin alt kümeler için küçültülmesi ile sağlanır. Bu şekilde devam edilerek, en kötü çözümlere en küçük

uyum değerleri verilir. MOGA yönteminde olduğu gibi her alt popülasyondaki çözümler arasındaki uzaklıklar Eşitlik 2.7. formülü ile hesaplanır. Bu uzaklıklar ile çözümlere ait paylaşım fonksiyon değerleri, uygunluk sayıları ve paylaşım uyum değerleri MOGA yönteminde kullanılan formüller yardımıyla hesaplanır. Rulet çemberi yöntemi ile çözümler seçilir. Bu seçimde alt edilemeyen 1. alt popülasyondaki (P_1) çözümlerin seçilme şansı daha çok olmaktadır. Seçilen çözümlere çaprazlama ve mutasyon uygulanarak yeni çözümler elde edilir ve durdurulma kriteri oluşuncaya kadar algoritma çözüm aramaya devam eder.

NSGA yöntemi alt edilme durumlarına göre iyi şekilde çözümleri sınıflandırmakta ve uyum değerleri atamaktadır. MOGA yöntemine benzer şekilde F_{min}, ε ve σ_{share} parametrelerinin belirlenmesi gerekmektedir. Bu parametreler algoritmanın iyi işlemesi açısından önemlidir ve dikkatli seçilmesi şarttır. Algoritmanın özellikle σ_{share} parametresine çok duyarlı olduğu yapılan çeşitli uygulamalardan bilinmektedir [29].

2.3.4. Uygun Pareto Genetik Algoritma

Uygun Pareto Genetik Algoritma (Niched Pareto Genetic Algorithm, “NPGA”) Horn vd. [30] tarafından geliştirilmiştir. Buraya kadar bahsedilen yöntemlerin aksine NPGA’da ikili turnuva seçim yöntemi uygulanmaktadır. Algoritma için öncelikle rastgele çözümler seçilerek P popülasyonu elde edilir. Yine rastgele seçilen çözümlerle, t_{dom} büyüklüğünde bir alt popülasyon seçilir. P popülasyonundaki çözümlerden ilk iki sıradaki çözümler seçilerek turnuva yapılır. Buna göre seçilen çözümler ile alt popülasyondaki çözümler karşılaştırılır. Alt popülasyondaki çözümler P’den seçilenleri alt ediyorsa, alt popülasyonda en çok sayıda bireyi alt eden çözüm, birinci ana çözüm olarak alınır. Eğer P’den seçilenler, alt popülasyondaki çözümleri alt edebiliyorsa, P’den en çok sayıda bireyi alt eden çözüm, birinci ana çözüm olarak alınır. Karşılaşılacak bir başka durum ise her iki gruptaki çözümlerin birbirini alt edememeleridir. Bu durumda MOGA yöntemindeki gibi P’den seçilen çözümler ile alt popülasyondaki çözümler arasında uzaklıklar hesaplanır ve buradan paylaşım fonksiyonu değerleri bulunarak yine σ_{share} parametresi ile karşılaştırılır. Gerekli olan uygunluk sayıları hesaplanarak, uygunluk sayısı büyük olan çözüm birinci ana çözüm olarak alınır. İkinci ana

çözüm de yukarıdaki karşılaştırma ve gerekirse hesaplamalardan elde edilir. Popülasyondaki tüm çözümler için turnuva yapılmalıdır. İki ana çözüm elde edilince çözümler arasında çaprazlama ve mutasyon yapılarak yeni çözümlere ulaşılır. N büyüklüğüne sahip P popülasyonu için yeteri kadar çözüm bu şekilde üretilir. Durdurma kriterine kadar algoritma kendini tekrar eder.

Bu algoritmada açık olarak uyum değerleri hesaplanmadığından VEGA, MOGA ve NSGA yöntemlerindeki gibi uyum hesaplama için zaman kaybedilmemektedir. Algoritmada seçim stratejisi olarak turnuva seçim tekniği kullanılmıştır. Yapılan çalışmalarla algoritma performansının, t_{dom} parametresine çok bağımlı olduğu bulunmuştur. Bunun yanı sıra belirlenen σ_{share} parametre değeri algoritmanın performansını etkilemektedir [5],[31].

2.4. Seçkin Çözümlerin İzlendiği Algoritmalar

Seçkin çözümlerin izlenmediği GA tabanlı algoritmalardan birkaç yıl sonra bu çözümlerin özel olarak nesiller boyunca takip edildiği algoritmalar önerilmiştir. Böylece ilk nesilde bulunan bir birey eğer sonraki nesillerde bulunan bireylerden daha iyi ise, son nesle kadar varlığını sürdürebilmektedir. Bütün nesillerde GA operatörleri ile bu çözümlerden yeni çözümler üretme şansı diğer çözümlere göre daha yüksek tutulmaktadır.

2.4.1. Sıralı Seçkin Alt Edilemeyen Genetik Algoritma

Sıralı Seçkin Alt Edilemeyen Genetik Algoritma (Elitist Non-dominated Sorting Genetic Algorithm, “NSGA II”) Deb ve Goel [32], [33] tarafından geliştirilmiştir. Bu algoritma NSGA yöntemine benzemektedir; ancak F_{min} ve ε parametreleri kullanılmamaktadır. Algoritma rastgele seçilen N tane çözümün bulunduğu P_1 popülasyonu ile başlar ve bu çözümlere çaprazlama ve mutasyon operatörleri uygulanarak N tane çözümü olan yeni Q_1 popülasyonu yaratılır. P_1 ve Q_1 birleştirilip R_1 birleşik popülasyonu elde edilir. Bütün çözümler birbirleriyle karşılaştırılıp, NSGA’daki gibi alt edilip edilmeme durumuna göre gruplanarak F_i alt kümeleri elde edilir. F_1’de alt edilemeyen, F_2’de ise F_1’deki çözümlerden sonra alt

edilemeyen çözümler vardır. Bu şekilde devam edilerek tüm çözümler gruplanır. Bulunan adımdaki orijinal popülasyon P_t olarak gösterilirse, N tane çözümün olması gereken ve bir adım sonrasına gerekli P_{t+1} popülasyonunun oluşturulması F_1 kümesindeki çözümlerle başlar. F_1'deki çözümlerin sayısı N'den az ise F_2'ye geçilir. Bu şekilde devam edilerek N çözümün bulunduğu kümedeki tüm çözümler seçilir. Son kümeden bütün çözümler seçildiği için istenilen miktardan çok çözüm elde edilmiştir, bu yüzden sonraki kümede bulunan çözümlere ihtiyaç yoktur. P_{t+1} popülasyonunda N tane çözüm olması gerektiği için, en son çözüm alınan kümeden, kalabalık uzaklık atama prosedürü (crowded distance assignment procedure) ile gerekli sayıda çözüm alınır. Bu teknikte, seçim yapılacak alt kümede bulunan çözümler, önce 1. amaca verdiği değere göre iyiden kötüye sıralanır ve çözümlere ait uzaklıklar hesaplanır. En iyi ve en kötü çözümlerin uzaklığı sonsuz (∞) olarak belirlenir ve arada kalan değişkenlerin uzaklıkları

$$d_i = \frac{f_1^{(i+1)} - f_1^{(i-1)}}{f_1^{max} - f_1^{min}} \qquad (2.12)$$

formülü ile hesaplanır. Bu oranın pay kısmında i. çözümden sonra ve önce gelen çözümlerin 1.amaç fonksiyonuna verdikleri değerler arasındaki fark vardır. Payda da ise 1. amaç fonksiyonuna ait maksimum ile minimum değerlerin arasındaki fark vardır. Daha sonra aynı kümedeki çözümler 2. amaç fonksiyonuna verdikleri değerlere göre sıralanır ve Eşitlik 2.12. formülü ile uzaklıklar bulunur. Çözümlerin 1. amaca göre bulunan uzaklık değerleri, ikinci amaca göre bulunanla toplanır ve böylece çözümün toplam uzaklık değeri bulunur. M tane amaç fonksiyonu varsa, bu işlem M kez tekrarlanır ve bulunan toplam uzaklıklar çözümlerin son uzaklık değerleri olur. P_{t+1} için gerekli olan çözüm sayısı kadar çözüm, bulunan uzaklık değerlerine göre seçilir. Öncelikle büyük uzaklık değerine sahip çözümler alınır.

Elde edilen P_{t+1} popülasyonunda en son alt kümeden seçilen çözümlerin uzaklık değerleri bilinmektedir ancak önceki seçilenlerin de hesaplanması gerekir. Bu uzaklıklar kalabalık turnuva seçimi (crowded tournament selection) için kullanılır. Bu seçime göre rastgele seçilen iki çözümün önce alt etme sıra sayılarına bakılır. Hangisi büyük ise, o çözüm seçilir. Eğer aynı ise, bulunan uzaklık değeri büyük olan çözüm seçilir ve böylece eşleme kümesi oluşturulur. Buradaki çözümlere mutasyon uygulanabilir. Sonuçta yeni Q_{t+1} popülasyonu oluşturulur. Durdurma kriteri sağlanıncaya kadar algoritma tekrar edilir [5].

NSGA II, σ_{share} uygunluk parametresine ihtiyaç duymamaktadır. Seçkin sonuçlara daha çok şans tanıyan algoritmada, bu çözümlerin kaybolması imkânsız gibidir. Dolayısıyla algoritma içinde bulunduğu adıma kadar bulduğu Pareto optimal çözümü hiçbir zaman kaybetmez. Çözüm seçme mekanizması, popülasyon büyüklüğünü kısıtlamak amacıyla kullanılmaktadır; ancak bu durumda algoritma optimal çözüme yakınlaşma özelliğini kaybedebilmektedir. Genellikle ilk alt edilmeyen çözümlerin olduğu F_1 alt kümesinde bulunan çözüm sayısı ana popülasyon sayısından büyük olmadığı için bu kümedeki tüm çözümler seçilmektedir. Böylece seçilen çözümler arasında çeşitlilik sağlanamamaktadır.

2.4.2. Güçlü Pareto Evrimsel Algoritma

Güçlü Pareto Evrimsel Algoritma (Strength Pareto Evolutionary Algorithm, "SPEA") Zitzler ve Thiele [34] tarafından geliştirilmiştir. SPEA yönteminde ilk önce N tane çözümün rastgele seçilmesiyle P_1 popülasyonu oluşturulur. Buna ek olarak N_1 seçkin çözümün konulacağı S_1 boş kümesi oluşturulur. P_t popülasyonu herhangi t.kuşakta elde edilen çözüm popülasyonu ise, P_t 'de alt edilemeyen en iyi çözümler S_t kümesine de kopyalanır. Bu popülasyonda elde edilen çözüm sayısı seçilen N_t parametresinden büyük ise kümeleme yöntemi uygulanarak sayı düşürülür. Kümeleme yönteminde S_t kümesinde bulunan her çözüm ayrı küme olarak alınır. Kümelerin arasındaki uzaklık

$$d_{c_k,c_l} = \frac{1}{n_{c_k} n_{c_l}} \sum_{\substack{i \in c_k \\ j \in c_l}} d(i,j) \qquad (2.13)$$

formülü ile hesaplanır. Formülde

n_{c_k} = k kümesinde bulunan çözüm sayısı,

n_{c_l} = l kümesinde bulunan çözüm sayısı,

$d(i,j)$ = kümelerde bulunan her çift çözüm arasındaki Öklit uzaklıkları

olarak yer almaktadır [35]. Buna göre iki sınıf arasındaki uzaklık sınıflarda bulunan çözümler arasındaki Öklit uzaklıklarının ortalamasıdır. Bu uzaklıklardan en küçük uzaklığa sahip iki sınıf birleştirilebilir. Bu şekilde S_t'de bulunan çözümlerden N_t tane sınıf elde edileninceye kadar kümeleme yöntemine devam edilir. En son elde

edilen N_t tane kümenin her birinden sadece bir tane çözüm seçilir. İçinde bir tane çözüm bulunan kümelerden eldeki tek çözüm otomatik alınır. Birden fazla çözüm bulunan kümelerden çözüm almak için, öncelikle bu çözümlere ait amaç fonksiyon değerlerinin ortalaması hesaplanır. Daha sonra kümede bulunan çözümler ile hesaplanan ortalama çözüm arasındaki Öklit uzaklıkları bulunur. Bu uzaklığı en küçük olan çözüm bu kümeden seçilir. Böylece S_{t+1} popülasyonu oluşturulmuş olur. P_t ve S_{t+1} popülasyonlarında bulunan çözümlere uyum değeri atanır. Önce S_{t+1}'deki çözümlere

$$U_i = \frac{n_i}{N+1} \tag{2.14}$$

formülü ile uyum değerleri verilir. Burada

U_i = i. çözüme karşılık gelen uyum değeri,

n_i = S_{t+1} popülasyonunda bulunan i. çözümün P_t popülasyonuna ait kaç tane çözümü alt ettiğini gösteren sayı

olarak tanımlanmıştır. P_t'de bulunan çözümlere

$$F_j = 1 + \sum_{i \in \bar{P}_{t+1}, i \le j} U_i \tag{2.15}$$

formülü ile uyum değeri verilir. Formülden de anlaşılacağı gibi P_t'de j. çözümü alt eden ve S_{t+1} popülasyonunda yer alan çözümlere ait güç toplamlarına 1 eklemek suretiyle j.çözümün uyum değeri bulunur. Bu formüller ile alt edilemeyen çözümlere daha küçük güç değerleri verilmektedir. Bundan sonra P_t ve S_{t+1} popülasyonları birleştirilir ve uyum değerlerine göre rasgele N çift çözüm alınır. Bunlar arasında ikili turnuva yapılarak en iyi çözümler belirlenir ve bu çözümlere çaprazlama ve mutasyon uygulanarak (t+1). kuşak bireyleri (P_{t+1} popülasyonu) oluşturulur.

Bu algoritmada kümeleme analizi ile Pareto optimal çözümlerin seçiminde çeşitlilik sağlanmaktadır. Uzaklıkların hesabı kolaydır ve fazladan parametreye ihtiyaç duyulmamaktadır. Algoritmada seçkin çözümlerin takip edileceği popülasyonların büyüklüğü olan N_t parametresinin belirlenmesi gerekmektedir. Ayrıca algoritmanın iyi sonuç verebilmesi için ana popülasyon büyüklüğü N ile N_t arasında denge sağlanmalıdır. N_t ne çok büyük ne de çok küçük olmalıdır. Çok büyük olursa algoritma seçkin çözümlerle çok vakit kaybeder ve başka çözümler üretemeyerek optimal çözümlere yakınsamayabilir. Çok küçük olursa seçkin kümedeki

çözümlerden az yararlanılır ve algoritmanın optimal çözümlerin olduğu alanın dışında çok arama yapmasına neden olabilir. Yapılan uygulamalar, $N=0,25N_t$ olarak alınırsa algoritmanın iyi işleyeceğini göstermektedir. Algoritmanın eleştirilen bir yönü, popülasyonların oluşumu sırasında ana popülasyonda bulunan çözümlere alt edilme kriterine uyum değeri verilmesidir. Ayrıca bulunan güç değerleri bazen aynı önemdeki alt edilmeyen çözümleri aynı değerde tutamamaktadır [5].

2.4.3. Arşivlenmiş Pareto Evrimsel Strateji

Arşivlenmiş Pareto Evrimsel Strateji (Pareto Archived Evolutionary Strategy, "PAES") Knowles ve Corne [36] tarafından geliştirilmiştir. Algoritma rastgele seçilen bir çözüm ile başlar ve mutasyon ile başka bir çözüm elde edilir. Bu çözümler karşılaştırılır. Hangisi iyi ise gelecek kuşak için ana çözüm olur. Herhangi t kuşakta ana çözüm (P_t) ve mutasyonla elde edilen yeni çözüme (C_t) ek olarak arşivlenen çözümler vardır. Arşivleme ile karşılaştığı çözümü alt eden bireyler kaydedilir. Eğer P_t çözümü C_t çözümünü alt ediyorsa C_t çözümü kabul edilmez ve mutasyon ile yeni çözüm aranır. Yeni C_t çözümü P_t çözümünü alt ediyorsa yeni çözüm kabul edilir ve C_t ana çözüm olmakla birlikte bir kopyası da arşive alınır. Böylece her alt eden çözüm ile karşılaşmada arşive alınan çözümler ile arşiv kalabalıklaşır. Eğer ana ve yeni çözüm birbirlerine üstünlük sağlayamıyorlarsa yeni çözüm olan C_t arşivdeki çözümler ile karşılaştırılır. Eğer arşiv çözümlerinden biri C_t çözümünü alt ediyorsa, C_t reddedilir ve P_t çözümüne mutasyon uygulanıp yeni çözüm aranır. Eğer C_t çözümü arşiv çözümlerinden bir ya da daha fazlasını alt ediyor ise, C_t arşive alınır. Alt ettiği çözüm ya da çözümler arşivden silinir. C_t çözümü bir sonraki kuşak için ana çözüm olarak alınır. C_t çözümü arşivdeki çözümleri alt edemiyor, arşivdeki çözümlerde C_t çözümünü alt edemiyorlarsa, yer varsa C_t arşive alınabilir. C_t'nin mi yoksa P_t'nin mi gelecek kuşak için ana çözüm olacağına ya da eğer arşivde yer yoksa hangisinin arşivde kalacağına karar vermek için komşuluk araştırması yapılır [36].

PAES algoritmasında komşuluk araştırması ile bir çözümün yoğunluk hesaplanması yapılmaktadır. Öncelikle her amaç fonksiyonunun alabileceği değer aralığı 2^d tane eşit parçaya bölünür. Derinlik (depth) parametresi olarak

adlandırılan d, algoritmayı uygulayan tarafından belirlenmektedir. Eğer M tane amaç fonksiyonu varsa, araştırılan alan, 2^{dM} tane aynı büyüklükteki bölgelere ayrılmış olur. Arşivlenmiş çözümlerin yerleri belirlenir. C_t ve P_t çözümlerinin yoğunluğu kendi komşuluklarında bulunan çözüm sayısıdır. Eğer C_t çözümü P_t çözümüne göre daha az kalabalık yerde ise C_t arşive alınır. Değil ise P_t arşivde kalır ve C_t çözümünden vazgeçilir. Arşivde yer varsa C_t arşive girer ve gelecek kuşakta hangi çözümün ana çözüm olarak alınacağına yine komşuluk araştırması ile karar verilir.

PAES algoritması çözüm alanını iyi bir şekilde bölerek çeşitliliği sağlamaktadır. Algoritma için d parametresinin seçimi önemlidir. Çok küçük olursa, elde edilen bölgeler büyük olacak ve çeşitlilik istenilen düzeyde olmayacaktır. Çok büyük olursa 2^{dM} ile bölge sayısı üstel artacak ve çok fazla küçük bölge olacaktır. Bu ise, algoritmanın çalışma zamanını kötü yönde etkileyecektir. Ayrıca arşiv büyüklüğünün seçimi de algoritmanın iyi çalışması için önemlidir [5].

PAES algoritmasının, (1+λ) PAES ve (μ,λ) PAES olarak iki uygulaması daha vardır. (1+λ) PAES algoritmasında bir ana çözüme λ kez mutasyon uygulanarak yeni çözümler bulunur. Bunlardan hangisinin ana çözüm ile karşılaşacağını bulmak için arşivlenmiş çözümlerle karşılaştırılmak suretiyle yeni çözümlere uyum değerleri atanır. En iyi uyuma sahip olan çözüm ana çözüm ile karşılaştırılır. Algoritmanın devamı PAES ile aynıdır. (μ,λ) PAES algoritmasında ise μ tane ana çözüm ile λ tane yeni çözüm arşiv çözümleri ile karşılaştırılır. Yeni çözümlerden arşive girecekler belirlenir ve arşiv güncelleştirilir. μ tane ana çözümün belirlenmesi için hem ana hem de yeni popülasyondaki tüm çözümler arşiv çözümleri ile karşılaştırılır. Bu çözümler içinde arşiv çözümlerinden birini ya da daha fazlasını alt edenlere 1 değeri, arşiv çözümlerinin alt ettiklerine -1 değeri ve arşiv çözümlerini alt edemeyen ama alt da olmayan çözümlere de 0 değeri verilerek, her çözüme uyum değeri atanır. Bu değerler kullanılarak ana ve yeni popülasyondaki çözümler arasında ikili turnuva yöntemi ile μ tane yeni çözüm gelecek kuşak için seçilir.

Bu iki yöntemin orijinal PAES yönteminden daha iyi sonuç vermediği uygulamalar ile bulunmuştur [5].

2.5. Kısıtlı Çok Amaçlı Problemlerde Genetik Algoritma ile Çözüm Yöntemleri

Kısıtlı optimizasyon problemlerinde, kısıtların sağlanması için bazı yöntemler önerilmiştir. Bu yöntemler öncelikle tek amaçlı problemler için geliştirilmiş ve sonra bunlardan yararlanılarak çok amaçlı problemler için öneriler yapılmıştır [37]. Bu yüzden tek amaçlı problemlerde uygun olmayan çözümlere yaklaşımlar üzerinde durulacaktır.

2.5.1. Kısıtlı Tek Amaçlı Problemlerde Genetik Algoritma ile Çözüm Yöntemleri ve Uygun Olmayan Çözümlere Yaklaşımlar

Tek amaçlı optimizasyon problemlerinde GA ile çözüm arama sırasında uygun olmayan çözümlerle karşılaşılabilir. Bu durumda bulunan uygun çözümler hemen elenerek hiçbir değerlendirmeye alınmayabilir. Bu basit yaklaşıma ölüm cezası (death penalty) yaklaşımı denir. Bu yöntemin sakıncaları vardır. Algoritma devam ederken bir süre sonra uygun bir çözüm bölgesine girecek ve hiçbir zaman buradan çıkamadan bu bölgede araştırmasına devam edecektir. Çözüm uzayı parçalı uygun bölgelere ayrılmış bir problemde algoritma başka bölgelere sıçrayamayacaktır. Bunun yerine uygun olmayan çözümler için ceza terimi hesaplanarak bu terimi amaç fonksiyonuna yansıtmak (maksimum ise eklemek, minimum ise çıkartmak) ve böylece uygun olmayan çözümlerden de faydalanmak daha iyi bir yaklaşımdır. Ceza teriminin hesaplanması için ceza fonksiyonuna ihtiyaç vardır. Ceza fonksiyonu yöntemi ile çözüm arama, ceza fonksiyonu türüne göre sınıflandırılmıştır. Ceza terimi statik, dinamik ya da uyarlı ceza fonksiyonları ile hesaplanabilir. Statik ceza fonksiyonu;

$$P_s(x) = \sum_{j=1}^{M} R_j f_j^2(x) \qquad (2.16)$$

şeklinde yazılır [38]. R_j j. kısıta ait ceza katsayısıdır ve kısıtlara özel olarak belirlenir. $f_j(x)$ j. kısıtın ihlal miktarıdır. Dinamik ceza fonksiyonu;

$$P_d(x) = (C * t)^{\alpha} \sum_{j=1}^{M} f_j^{\beta}(x) \qquad (2.17)$$

olarak yazılır [38]. Bu yöntem Joines ve Houck yöntemi olarak da bilinir. Burada $f_j(x)$ statik formüldeki gibi j. kısıtın ihlal miktarını göstermektedir. C, α ve β sabit ve α genelde 1 ya da 2 olarak alınmaktadır. t ise nesil sayısını göstermektedir. Buna göre algoritma ilerledikçe ceza büyümektedir.

Uyarlı ceza fonksiyonu olarak literatürde kabul gören önemli fonksiyonlar vardır. Bunlardan Smith, Tate ve Coit [39] yönteminde uygulanan uyarlı ceza fonksiyonu;

$$P_{STC}(x) = -\sum_{i=1}^{M}\left(\frac{\Delta b_i(x)}{\Delta b_i^{nef}}\right)^{\alpha}\left(f_{all}^{\ *} - f_{feas}^{*}\right) \qquad (2.18)$$

olarak verilmektedir. α ceza fonksiyonunun önemini belirten parametredir ve genelde 2 olarak alındığı gözlenmektedir. $\Delta b_i(x)$ i. kısıt ihlal miktarını, Δb_i^{nef} ise her problem için özel olarak belirlenen uygunluğa yakınlık değerini (near feasibility thrashold) göstermektedir. f_{feas}^{*} o zamana kadar bulunan en iyi uygun çözüme ait amaç fonksiyon değeridir. f_{all}^{*} o zaman kadar bulunan tüm çözümler içinde en iyi çözüme ait cezasız amaç fonksiyonu değeridir.

Gen ve Cheng [40] yönteminde ceza fonksiyonu;

$$P_{GC}(x) = 1 - \frac{1}{m}\sum_{i=1}^{M}\left(\frac{\Delta b_i(x)}{\Delta b_i^{maks}}\right)^{\alpha} \qquad (2.19)$$

olarak verilmektedir. Burada $\Delta b_i(x) = maks\{0, g_i(x)-b_i\}$ ve $\Delta b_i^{maks} = maks\{\varepsilon, \Delta b_i(x)\}$ olarak tanımlanmıştır. ε ise sıfıra bölümden kaçınmak amacıyla küçük pozitif bir sayı olarak alınmaktadır. Buna benzer bir yöntem olan Yokota, Gen, Ida ve Taguchi yönteminde ceza fonksiyonunda Δb_i^{maks} yerine sadece sağ yan değerleri olan b_i değerlerinin yazılması ile elde edilen ceza fonksiyonu uygulanır.

Bu yöntemlerden Smith, Tate ve Coit tarafından önerilen uyarlı ceza fonksiyonu yönteminin kullanıldığı uygulamalarda diğer yöntemlere göre daha iyi sonuç elde edildiği araştırmalardan anlaşılmaktadır.

2.5.2. Kısıtlı Çok Amaçlı Problemlerde Genetik Algoritma ile Çözüm Yöntemleri ve Uygun Olmayan Çözümlere Yaklaşımlar

Tek amaçlı optimizasyondaki gibi çok amaçlı problemlerde de bir çözüm herhangi bir kısıt ya da kısıtları ihlal ettiğinde hiç dikkate alınmadan atılabilir [41]. Önceden belirtilen sakıncalardan dolayı bu yaklaşım yerine alt etmeye ve ceza fonksiyonuna dayalı yöntemler ortaya çıkmıştır.

Ceza fonksiyonu yönteminde bütün amaç fonksiyonları minimum türüne çevrilir. Kısıtlar ≥ 0 biçimine dönüştürülür. Bir x çözümünün j. kısıt ihlali eğer $g_i(x) < 0$ ise;

$$w_j(x) = |g_i(x)| \tag{2.20}$$

şeklinde bulunur. $g_i(x) \geq 0$ ise kısıt ihlal edilmemektedir ve ceza sıfırdır. J tane kısıttan elde edilecek toplam kısıt ihlali ise;

$$\Omega(x) = \sum_{j=1}^{J} w_j(x) \tag{2.21}$$

şeklinde toplanarak bulunur. Toplam ihlal miktarı R_m ceza parametresi ile çarpılarak çözüme ait tüm amaç fonksiyonlarına eklenir. Yeni amaç fonksiyon değerleri;

$$F_m(x) = f_m(x) + R_m\Omega(x) \tag{2.22}$$

ile hesaplanır. Cezalı amaç fonksiyon değerleri ile bulunan çözüm, diğer çözümlerle karşılaştırılır [18], [42].

Ceza fonksiyonu yöntemi yanı sıra çok amaçlı optimizasyon da önemli yeri olan alt etme ölçütüne göre uygun olmayan çözümlere yapılan yaklaşımlar vardır.

Alt etme ölçütüne dayanan bir yöntem olan Jimenez-Verdegay-Gomez-Skarmeta [43] yönteminde uygun olmayan çözümler algoritmanın akışı içinde halledilmektedir. Burada turnuva yöntemine göre çözüm seçilmektedir. Yöntemde karşılaşılan iki çözüm için izlenen yol aşağıda ifade edilmiştir.

a. İki çözümde uygun çözüm ise izlenen yol NPGA ile aynıdır. rastgele oluşturulan uygun çözüm kümesi ile iki çözüm karşılaştırılır. Eğer biri bu küme tarafından alt edilir ama diğeri alt edilemez ise, alt edilemeyen seçilir. Eğer her ikisi alt ediliyorsa ya da alt edilemiyorsa, NPGA yöntemindeki gibi uygunluk sayıları hesaplanarak σ_{share} parametresi yardımı ile çözümlerden biri seçilir.

b. Eğer çözümlerden biri uygun ancak diğeri uygun değilse, uygun olan doğrudan seçilir ve uygun olmayan çözümden vazgeçilir.
c. İki çözümde uygun değilse, popülasyondan rasgele seçilen uygun olmayan çözümlerin bulunduğu küme ile karşılaştırılır. Çözümlerden biri en iyi uygun olmayan çözümden daha iyi ve diğeri daha kötü ise, iyi olan seçilir. Eğer her ikisi de en iyi uygun olmayan çözümden daha iyi ya da daha kötü ise, iki çözüm aynı derecede iyi ya da kötü olarak kabul edilir. Bu durumda (a)'daki gibi çözümler için uygunluk sayıları hesaplanır. Küçük uygunluk sayısına sahip olan çözüm turnuvayı kazanır [43].

Deb [44], kısıtları ihlal eden çözümlere alt etme ölçütüne dayalı yaklaşımları incelerken kısıtlı – alt eden çözüm tanımını vermiştir ve bu tanım yardımı ile kısıtlı turnuva yöntemini önermiştir.

Tanım 4. Kısıtlı- alt eden çözüm: Aşağıda verilen durumlara göre i çözümü j çözümüne göre kısıtlı-alt eden bir çözümdür.
a. i ve j çözümleri uygun ve i çözümü j çözümünü alt eder.
b. i uygun çözüm ancak j uygun çözüm değildir.
c. i ve j çözümleri uygun değil ancak iki çözümün kısıtlara verdiği sağ yan değerlerine göre i çözümü daha küçük kısıt ihlaline sahiptir.

Bu tanıma Coello [45] katkıda bulunmuştur. Kendisi tanımda Deb'in verdiği a ve b kısımları aynen almış, (c) kısmını değiştirerek bir madde daha eklemiştir. Coello'ya göre;
c. i ve j çözümleri uygun değil ancak i çözümü j çözümüne göre daha az sayıda kısıt ihlal etmektedir.
d. i ve j çözümleri uygun değil ve her ikisi de aynı sayıda kısıtı ihlal etmektedir, ancak iki çözümün kısıtlara verdiği sağ yan değerlerine göre i çözümü daha küçük kısıt ihlaline sahiptir.
d seçeneği için kısıt miktar katsayısı;

$$K(x) = \sum_{n=1}^{n_{maks}} maks(g_n(x),0) \qquad (2.23)$$

olarak hesaplanır. En küçük kısıt miktar katsayısına sahip çözüm tercih edilir.

Bu tanımdan sonra kısıtlı turnuva seçim yöntemi verilebilir. x_1 ve x_2 iki farklı çözüm olmak üzere;

a. x_1 çözümü x_2 çözümüne göre kısıtlı- alt eden çözüm ise, x1 seçilir.

b. x_1 ve x_2 çözümleri kısıtlı- alt etme ölçütüne göre birbirine üstünlük sağlayamıyorsa, çeşitliliği arttırmak amacıyla iki çözüm arasında daha az kalabalık olan komşulukta bulunan çözüm seçilir. Bunun için çeşitli ölçütler önerilmektedir. Bunlardan biri uygunluk sayısı ölçütüdür. MOGA yönteminde kullanılan paylaşım fonksiyonu yardımıyla uygunluk sayıları hesaplanır σ_{share} parametresinin kullanılmasından sonra küçük uygunluk sayısına sahip çözüm seçilir. Diğer bir ölçüt baş sayısı (head count metric) ölçütüdür. Bu ölçütte σ_{share} komşuluğunda bulunan çözüm sayısı bulunur ve çözüm sayısı daha küçük olan komşulukta yer alan çözüm seçilir. Komşuluklarda bulunan toplam çözüm sayısı aynı olabilir. Bu durumda uygunluk sayısı ölçütü daha iyi sonuç verebilir. Son olarak önerilen ölçüt NSGAII yönteminde uygulanan kalabalık turnuva seçimidir. Bu seçime göre rasgele seçilen iki çözümün alt etme sıra sayılarına bakılır. Hangisi küçük ise o çözüm seçilir. Eğer aynı ise yöntemde hesaplanan kalabalık uzaklık ölçütü kullanılır ve bu ölçütü büyük olan çözüm seçilir [6].

3. GENETİK ALGORİTMA TABANLI ÇOK AMAÇLI KARAR VERME PROBLEMLERİ İÇİN YENİ ALGORİTMALAR

Bu kesimde kısıtlı ve kısıtsız çok amaçlı problemlerin çözümü için kullanılabilecek algoritmalar önerilmiştir. Bunlardan kısıtlı durum için verilen algoritmanın iki değişik şekilde işlemesi düşünülmüştür. Önerilen kısıtlı çok amaçlı genetik algoritma, uygun olmayan çözümlerin değerlendirilmesi aşaması dışında kısıtsız durum için önerilen algoritma ile aynı işlemektedir.

Algoritmaların Pareto optimal bölgedeki çözümleri bulmada etkili olmasını sağlayan özelliği yeni çözümlerin oluşturulma aşamasında kullanılan yöntemlerdir. [5],[31] . Bu aşamada istatistiksel bir yöntem kullanma düşüncesiyle kısıtsız çok amaçlı GA oluşturulmuştur. Kullanılan istatistiksel yöntem kümeleme yöntemidir. Literatürde bu aşamada istatistiksel bir yöntem kullanan bir algoritma yoktur. Kümeleme yöntemi, önerilen algoritmaların bulduğu çözümlere çeşitlilik katmakta, aynı zamanda komşuluk inceleme özelliği sağlayarak kesikli çözüm bölgelerine sahip problemlerde, birbirinden kopuk bölgelerden çözüm örnekleri alma olasılığını arttırmaktadır.

Seçkin çözümlerin izlendiği algoritmalar iyi sonuç verdiği literatürden elde edilmiştir [3] [31]. Önerilen algoritmalarda seçkin çözümlere daha çok yeni çözüm üretme şansı verilmiştir. Kısıtlı durum için önerilen algoritmaların ikisinde de uygun olmayan çözümlerin seçimi seçkin çözümlerle ilişkilendirilmiştir. Literatürde araştırılan algoritmalarda seçkin çözümler sadece yeni çözüm üretmek için kullanılmıştır. Kısıtlı durum için önerilen ikinci algoritmada ceza fonksiyonu kullanılarak uygun olmayan çözümlere ceza verilmiş ve cezalı amaç fonksiyon büyüklükleri değerlendirilerek uygun ve uygun olmayan çözümler arasında karşılaştırmalar yapılmıştır. Böylece uygun olmayan çözümlere de seçilme şansı verilmiştir. Kısıtlı çok amaçlı genetik algoritmalarda ceza fonksiyonu kullanımı ilk olarak önerilen algoritma ile olmuştur.

Tüm bu eksiklikler belirlenerek GA temelli, seçkin çözümlerin izlendiği ve izlenmediği algoritmalardan yararlanarak, istatistiksel seçim yöntemi kullanılarak

ve kısıt durumunda uygun olmayan çözümlere uygulanan teknikler incelenerek çok amaçlı karar verme problemlerinin çözümü için üç yeni GA önerilmiştir.

3.1. Önerilen Kısıtsız Çok Amaçlı Genetik Algoritma

Kısıtsız çok amaçlı optimizasyon problemlerinin çözümleri için günümüze kadar geliştirilen GA tabanlı algoritmaların pozitif yönleriyle birlikte, eleştirilen tarafları incelendikten sonra yine GA tabanlı bir algoritma geliştirilmiştir.

Seçkin çözümlerin izlendiği algoritmaların Pareto optimal çözüm kümesini daha iyi yakaladığı bilinmektedir. Seçkin çözümlerin izlenmediği algoritmalarda (VEGA [25], MOGA [26], NSGA [29], NPGA [30]) kullanılan alt etme durumuna göre çözümlere sıra sayısı vermek ve bu sayılarla aralarında turnuva yapmak basit, hızlı ve etkili seçim yapılmasını sağladığı için seçkin çözümlerin izlendiği algoritmalarda da kullanıldığı bu uygulamanın alındığı gözlenmiştir [5]. Seçkin çözümlerin izlenmediği algoritmalarda kullanılan ekstra parametrelerin (örneğin “niche” sayısı) algoritmayı çok bağımlı hale getirdiği anlaşılmaktadır [15],[31]. Literatürdeki seçkin çözümlerin izlendiği ve izlenmediği yöntemlerde seçim mekanizması alt etme kriteri, uzaklık ölçütleri veya bunların yardımıyla hesaplanan uyum değerleriyle yapılmaktadır.

SPEA [34] yönteminde seçkin çözüm kümesinde oluşacak kalabalığın azaltılmasında kümeleme yöntemi kullanılmıştır. Bunun dışında literatürde hiç bir algoritmada istatistiksel yöntem kullanılmamıştır.

Önerilen algoritmada kümeleme yöntemiyle çözümlerin komşulukları belirlenmiştir; Elde edilen komşuluklardan alt etme kriterine göre çözümlere sıra sayıları verilmiştir. Sıra sayıları yardımıyla çözümler karşılaştırılmıştır. Her komşuluktan en az bir çözüm alınarak çeşitlilik mekanizması çalıştırılmıştır. Her kümeye ait sıra sayısı hesaplanmış ve bu değerler yardımıyla ana popülasyona alt etme özelliği fazla olan komşuluktan daha fazla çözüm seçilmiştir. Komşuluk incelenmesi diğer yöntemlerde olmayan bir özelliktir.

Önerilen algoritmanın adımları bu kesimde verilmiştir. Burada;

P_i = i. adımda elde edilen N tane çözümün oluşturduğu ana popülasyon

Q_i= i. adımda elde edilen yeni popülasyon kümelerini göstermektedir.

P'_i = i. adımda elde edilen ana popülasyondan (P_i) seçilen ve sadece bu popülasyon içinde alt edilemeyen çözümlerin olduğu küme

Q'_i = i. adımda elde edilen yeni popülasyondan (Q_i) seçilen ve sadece bu popülasyon içinde alt edilemeyen çözümlerin olduğu küme olarak tanımlanmıştır.

E_i= i. adımda $P'_i + Q'_i$ birleşik kümesinden seçilen, alt edilemeyen seçkin çözümlerin meydana getirdiği küme olarak tanımlanmıştır. E_i elit çözümlerin saklandığı ve her adımda yeni bulunan çözümlerle güncellenen çözüm kümesidir. Bu özelliği ile önerilen algoritmada önemli bir işlevi vardır. Kısıtsız çok amaçlı optimizasyon problemlerinin çözümü için önerilen algoritmanın adımları aşağıda ifade edilmiştir.

1. **Adım:** Rasgele N tane çözüm seçilir ve P_1 popülasyonu oluşturulur. Bu çözümlerden N kez rastgele çözüm çifti alınarak GA operatörleri uygulanır ve Q_1 yeni popülasyonu elde edilir. Herhangi t. adım için bu popülasyonlar P_t ve Q_t popülasyonlarıdır.
2. **Adım:** Birbirinden bağımsız olarak her iki popülasyonda bulunan çözümlerden alt edilmeyen çözümler bulunur ve P'_t, Q'_t kümeleri oluşturulur. Bu kümelerdeki çözümlere, alt etme durumuna göre sıra sayısı verilir. P'_t kümesindeki alt edilemeyen çözümler, P_t kümesindeki çözümlerden hiçbirini alt edemiyorsa "0" değerini alır. Alt ettikleri var ise, alt ettiği çözüm sayısını alır. Q'_t kümesindeki alt edilemeyen çözümlere de aynı şekilde sıra sayısı verilir.
3. **Adım:** $P'_t + Q'_t$ birleşik kümesi oluşturulur ve eğer varsa alt edilemeyen çözümler bulunur. Bu çözümler seçkin çözümler olarak E_t kümesine aktarılır. Seçkin kümeye konulan çözümlerin 2. adımdan elde edilen sıra sayıları, birleşik kümedeki alt etme durumuna göre güncellenir. En iyi sıra sayısı olan ilk 0.25 N tane çözüm bu kümede saklanır, diğer çözüm ya da çözümler silinir.
4. **Adım:** $P'_t + Q'_t$ birleşik kümesinde yer alan çözümler kullanılarak Aşamalı Kümeleme Analizi (Hierarchical Cluster Analysis) uygulanır ve çözümler kümelere ayrılır. Bu yöntemde bağlantı tekniği olarak küresel ortalama (Centroid Method) tekniği, uzaklık ölçütü olarak ise Öklit uzaklığı kullanılmaktadır. Bu adımda en son olarak elde edilen kümelerde bulunan

çözümlere ait sıralama sayıları toplanır ve kümeleri temsil eden toplam sıralama sayıları bulunur.

5. **Adım:** (t+1). adımdaki yeni kuşak için kullanılacak, N büyüklüğünde olması gereken, P_{t+1} ana popülasyonunda yer alacak çözümler seçilir. Bunun için 0.25N tane seçkin çözüm E_t popülasyonundan alınır. Geri kalan çözümler önceki adımda bulunan kümelerdeki çözümlerden seçilir. Çeşitliliği sağlamak amacıyla önce, her sınıfta en iyi sıra sayısına sahip olan çözüm seçilir. Böylece çözümlerin bulunduğu her bölgeden, o bölgeyi temsil eden en az bir çözüm, gelecek kuşağa aktarılmaktadır. Bu adımdan sonra iki durum ile karşılaşılabilir.
    a. P_{t+1}'de bulunan çözüm sayısı N'den az ise, en büyük sıra sayısına sahip kümeden en büyük sıralı çözüm alınır. Bu kümenin küme sayısı azalır. Daha fazla çözüm almak gerekiyorsa, yine kümelerin sıra sayısına göre önce küme belirlenir ve seçilen kümedeki çözümlerin sıra sayısına göre çözüm seçilir.
    b. P_{t+1}'de bulunan çözüm sayısı N'den fazla ise, bu popülasyonda bulunan çözümlerden sıra sayısı az olan ya da olanlar atılır. Böylece gereksiz yere başka çözümleri alt edemeyen ya da az sayıda çözümü alt edenler elenirler.
6. **Adım:** Durdurma kriteri sağlayıncaya kadar algoritma 2. adıma dönülerek devam eder.

Seçkin çözümlerin izlendiği E_t kümesi her adımda alt etme kriterine göre güncellenmesine rağmen; bu kümeye olması gereken sayıdan daha fazla çözüm girerse en yaşlı (en erken kümeye giren) çözümler kümeden atılır.

Bu algoritmada seçkin olan çözümlerin önemlileri izlenmekte ve gelecek kuşaklar için yararlanılmaktadır. Hem ana hem de yeni popülasyondaki çözümlerden yararlanılmaktadır. Bunlar arasındaki alt edilemeyen çözümler kümelere ayrılarak, çözümlerin geldiği her bölgeden en az bir temsilci alınmaktadır. Her zaman seçkin çözümlere gelecek kuşaklar için şans vermenin iyi olmayacağı düşünülerek, seçkin çözümler arasında da kısıtlama yapılmıştır. Birleşik popülasyonda alt edilen çözüm ya da çözümler olabileceğinden, alt edilen çözümlere de gelecek kuşaklarda kalıtımlarını sürdürme şansı verilmiştir. Çözüm seçiminde kümeleme

analizinden ve alt etme kriterinden faydalanılmıştır. Böylece ana popülasyonda bulunan çözüm sayısı ve seçkin kümede bulunan çözüm sayısı dışında herhangi bir parametre kullanılmamaktadır.

3. 2. Kısıtlı Çok Amaçlı Problemlerde Önerilen Genetik Algoritma

Kısıtlı çok amaçlı karar verme problemlerinin çözümünde kullanılacak yöntemlerde uygun olmayan çözümlerle karşılaşıldığında sanki bu çözüm hiç elde edilmemiş gibi değerlendirme dışı bırakılabilir veya bir ceza fonksiyonu yardımıyla değerlendirmeye alınabilir. Coello [45] uygun olmayan çözümlerin değerlendirilmesinde önce ihlal edilen kısıt sayısına, sonra kısıt miktarına bakılabileceğini göstermiştir. Oyama vd. [46] bu uygulamayı uçak üretimi modelinde kullanarak iyi sonuçlar elde etmiştir. Kısıtlı ve ceza fonksiyonu kullanılmadan önerilen algoritmada uygun olmayan çözüm ile uygun çözüm karşılaştırıldığında, uygun çözüm doğrudan seçilmektedir. Aslında burada Coello'nun önerdiği gibi ihlal edilen kısıt sayısı kriteri uygulanmaktadır; çünkü uygun çözümün ihlal ettiği kısıt yoktur. Eğer iki uygun olmayan çözüm karşılaştırılıyorsa, önce ihlal edilen kısıt sayısına bakılmış ve küçük olan seçilmiştir. İhlal edilen kısıt sayısı aynı olduğu durumda Coello'nun baktığı kısıt miktarı kriteri yerine önerilen yöntemde seçkin kümedeki çözümlere uzaklıklar kontrol edilmektedir. Böylece uygun olmayan çözümler içinde Pareto optimal bölgeye yakın olanların seçilmesi amaçlanmıştır.

Uygun olmayan çözümlerin karşılaştırılması dışında kısıtsız durum için önerilen algoritma ile aynı şekilde işleyen bu algoritma, kısıtsız durum için önerilen algoritmanın bütün özelliklerine sahiptir.

Bu algoritmada geçen N tane çözümün oluşturduğu P_i , Q_i i. adımda elde edilen, sırayla ana ve yeni popülasyonu göstermektedir. P'_i , Q'_i ise i. adımda elde edilen, ana ve yeni popülasyonların kendi içinden seçilen alt edilemeyen çözümlerin olduğu kümeleri tanımlamaktadır. i. adımda $P'_i + Q'_i$ birleşik kümesinden seçilen alt edilemeyen seçkin çözümlerin meydana getirdiği küme E_i ile gösterilmektedir. Önerilen algoritmanın adımları aşağıda ifade edilmiştir.

1. **Adım:** Rasgele N tane çözüm seçilir ve P_1 popülasyonu oluşturulur. Bu çözümlerden N kez rasgele çözüm çifti alınarak GA operatörleri uygulanır ve Q_1 yeni popülasyonu elde edilir. Bu şekilde t. adımda P_t ve Q_t popülasyonlarına ulaşılır.
2. **Adım:** Birbirinden bağımsız olarak P_t ve Q_t popülasyonlarından alt edilmeyen çözümler bulunarak P'_t , Q'_t kümeleri oluşturulur. Bu kümelerdeki çözümlere, alt etme durumuna göre sıra sayısı verilir. P'_t kümesindeki alt edilemeyen çözümler, P_t kümesindeki çözümlerden hiçbirini alt edemiyorsa "0" değerini alır. Alt ettikleri var ise, alt ettiği çözüm sayısını alır. Q'_t kümesindeki alt edilemeyen çözümlere de aynı şekilde sıra sayısı verilir.
3. **Adım:** $P'_t + Q'_t$ birleşik kümesi oluşturulur ve eğer varsa alt edilemeyen çözümler bulunur. Bu çözümler seçkin çözümler olarak E_t kümesine aktarılır. Bu işlem sırasında uygun olmayan çözüm ya da çözümlerle karşılaşılırsa kümeye alınmaz. Seçkin kümeye konulan çözümlerin 2. adımdan elde edilen sıra sayıları, birleşik kümedeki alt etme durumuna göre güncellenir. En iyi sıra sayısı olan ilk 0.25 N tane çözüm bu kümede saklanır, diğer çözüm ya da çözümler silinir.
4. **Adım:** $P'_t + Q'_t$ birleşik kümesinde yer alan çözümler kullanılarak Aşamalı Kümeleme Analizi uygulanır ve çözümler kümelere ayrılır. Bu yöntemde bağlantı tekniği olarak küresel ortalama tekniği, uzaklık ölçütü olarak ise Öklit uzaklığı kullanılmaktadır. Bu adımda en son elde edilen kümelerde bulunan çözümlere ait sıralama sayıları toplanır ve kümeleri temsil eden toplam sıralama sayıları bulunur.
5. **Adım:** (t+1). adımda ulaşılan, N büyüklüğündeki Pt_{+1} ana popülasyonunda yer alacak çözümler seçilir. Bunun için 0.25N tane seçkin çözüm E_t popülasyonundan alınır. Geri kalan çözümler önceki adımda bulunan kümelerdeki çözümlerden seçilir. Çeşitliliği sağlamak amacıyla önce, her sınıfta en iyi sıra sayısına sahip olan çözüm seçilir. Böylece çözümlerin bulunduğu her bölgeden, o bölgeyi temsil eden en az bir çözüm, gelecek kuşağa aktarılmaktadır. Her kümeden seçilecek çözümler uygun ise alınır; değilse ait olduğu kümedeki çözümlerden ikinci en iyi sıra sayısına sahip çözüme bakılır. Eğer bu çözüm uygun ise alınır; değilse iki uygun olmayan çözüm arasında tercih yapılması gerekir. Bu tercihin yapılması için, iki çözümün kısıtlardan kaç tanesini ihlal ettiğine bakılır ve daha az sayıda ihlal

eden çözüm seçilir. Aynı sayıda kısıt ihlal edilmiş ise seçkin kümedeki çözümlere uzaklığı az olan çözüm seçilir. Seçkin kümede N_E tane çözüm varsa i çözümünün bu kümeye uzaklığı;

$$d_{iE} = \sqrt{\sum_{j=1}^{N_E} \sum_{k=1}^{M} \left(\frac{f_k^{(i)} - f_k^{(j)}}{f_k^{max} - f_k^{min}} \right)^2} \qquad (3.1)$$

formülü ile hesaplanmaktadır. Burada $f_k^{(i)}$ tercih edilmesi söz konusu olan i çözümüne ait k.amaç fonksiyon değeridir. $f_k^{(j)}$ ise seçkin kümede bulunan j. çözüme ait k. amaç fonksiyon değeridir. j indisi seçkin küme büyüklüğü olan N_E'ye kadar gitmektedir. Böylece bu kümedeki tüm çözümlerin amaç fonksiyonları ile tercih edilecek çözümlere ait amaç fonksiyonları arasında farklar alınmaktadır. $f_k^{(max)}$ ve $f_k^{(min)}$ ise seçkin kümeye ait k. amaç fonksiyonunun maksimum ve minimum değerleridir. Burada kullanılan ve Eşitlik 3.1.'de görülen uzaklık formülü MOGA yönteminde çözümler arasında uzaklığın hesaplanmasında kullanılan formüldür.

Uygun olmayan çözümler arasında seçkin kümesine uzaklıklara göre seçim yapılmasının amacı Pareto optimal uygun bölgesine en yakın uygun olmayan çözümü seçmektir. Bu şekilde uygun olmayan çözümler arasından faydalı olabilecekler alınarak bunlardan elde edilecek yararlı bilgiler değerlendirilebilecektir.

Bu adımın sonunda iki durum ile karşılaşılabilir.

a. P_{t+1}'de bulunan çözüm sayısı N'den az ise, en büyük sıra sayısına sahip kümeden en büyük sıralı çözüm alınır. Bu kümedeki çözüm sayısı azalır. Daha fazla çözüm almak gerekiyorsa, yine kümelerin sıra sayısına göre önce küme belirlenir ve seçilen kümedeki çözümlerin sıra sayısına göre çözüm seçilir. Kümelerden seçilecek çözümlerin uygun olmaması durumunda yukarıda anlatılan yöntem ile seçim yapılır.

b. P_{t+1}'de bulunan çözüm sayısı N'den fazla ise, bu popülasyonda bulunan çözümlerden sıra sayısı az olan ya da olanlar atılır. Böylece gereksiz yere başka çözümleri alt edemeyen ya da az sayıda çözümü alt edenler elenirler. Bu aşamada bir seçimin elenmesi uygun olması ya da olmamasına göre yapılmamaktadır. Sıra

sayısına göre yapılarak uygun olmayan çözümlere de ileri nesillerde şans verilmektedir.

6. **Adım:** Durdurma kriteri sağlayıncaya kadar 1. adıma dönülerek algoritma devam ettirilir.

Bu algoritmada çözümlerin ihlal ettikleri kısıt sayısına ve seçkin kümeye uzaklıklarına bakılması; Pareto optimal çözüm kümesine en yakın olan çözümlerin seçilmesini sağlamaktadır. Bu ise seçilen çözümde genetik operatörlerin kullanımı ile uygun ve iyi bir çözüme ulaşma olasılığını arttırmaktadır. Yöntemin bir avantajı da 2.adımda yapılan çözümler arasındaki turnuvalardır. Turnuva yönteminin diğer yöntemlere göre daha iyi sonuç verdiği uygulamalarla kanıtlanmıştır [5]. Turnuva yöntemi ile alt etme ölçütü de çözümlerin seçimlerinde kullanılmıştır. Yöntemde kümelere ayrılan çözümler önceden uygun olup olamamalarına göre ayrılmadığı için uygun olmayan çözümler arasında da çeşitlilik sağlanmaktadır. Bu özellik diğer yöntemlerde gözlenmemektedir.

3.3. Önerilen Uyarlı Ceza Fonksiyon Yöntemi

Çok amaçlı optimizasyon yöntemlerinde uyarlı ceza fonksiyonu kullanılan GA literatürde bulunamamıştır. Tek amaçlı problemler için çalışmalar mevcuttur. Bu çalışmalardan en önemlisi Smith vd. [39] tarafından önerilen uyarlı ceza fonksiyonudur. Önerilen uyarlı ceza fonksiyonu bu araştırmacıların bulduğu ceza fonksiyonunun çok amaçlı problemler için değiştirilmesiyle elde edilmiştir. Bu algoritmanın kısıtsız durum için önerilen algoritmadan farkı uygun olmayan çözümlerin ceza fonksiyonu ile cezalandırılması ve ana popülasyon için yarıştırılmasıdır. Dolayısıyla kısıtsız algoritmanın özelliklerine sahiptir. Kısıtlı durum için önerilen ilk algoritmadan farkı ise, uygun olmayan çözümlere cezadan sonra uygun olanlarla birlikte eşit seçilme şansı tanımasıdır. İlk algoritmada uygun çözümlere öncelik tanınmaktaydı.

Bu algoritmada geçen P_i , Q_i , P'_i ve E_i önceki kesimde tanımlanan kümeleri temsil etmektedir. Ceza fonksiyonu kullanılmayan algoritma ile uyarlı ceza fonksiyonunu kullanan algoritmanın adımları arasında tek farklılık ceza fonksiyonu ve uygun olmayan çözümler nedeniyle 5. adımdır. Buna göre algoritmada ana

popülasyondan yeni popülasyon elde edildikten sonra alt edilme durumlarına göre popülasyonlar değerlendirilir ve sıra sayısı verilir. Birleşik kümedeki çözümlerden seçkin küme elde edilir. Kümeleme yöntemi ile birleşik kümedeki çözümler sınıflandırılır ve 5. adıma gelinir. 5. adım aşağıda verilmiştir.

5. Adım: (t+1). adımda ulaşılan, N büyüklüğündeki P_{t+1} ana popülasyonunda yer alacak çözümler seçilir. Bunun için 0.25N tane seçkin çözüm, E_t popülasyonundan alınır. Seçkin kümeye çözümler alınırken uygun olmayanlar alınmadığı için yeni kuşak ana popülasyonuna uygun olmayan çözüm gelmemektedir. Yeni ana popülasyona kümelerden çözümler seçilmesi sırasında her sınıfta en iyi sıra sayısına sahip olan çözüm alınır. Eğer seçilmesi gereken çözüm uygun değilse, ceza terimi hesaplanarak amaçların türüne göre çözümden elde edilen her amaç fonksiyon değerine eklenir ya da çıkartılır (maksimum ise çıkartılır, minimum ise eklenir). Kısıtların küçük eşit ya da eşit durumunda olduğu düşünülerek bir j çözümünün k.amaç fonksiyonuna gelecek ceza büyüklüğünün hesaplanması için;

$$p_k{}^j(x) = \sum_{i=1}^{M} \frac{\text{Maks}(0, \Delta b_i(x))}{\overline{\Delta b_i}(E)} \left[\text{maks}\left(\varepsilon, F_k^{(E)} - F_k^{(j)}\right)\right] \qquad (3.2)$$

uyarlı ceza fonksiyonundan yararlanılır. $\Delta b_i(x) = b_i - b_i^j$ problemde i.kısıta karşılık gelen sağ yan değeri ile j çözümüne ait karar değişkenleri değerlerinin yerine yazılmasıyla aynı kısıttan elde edilen değer arasındaki farktır. Bir başka değişle kısıt ihlal edilme miktarıdır. Eğer bütün kısıtlar için ihlal yoksa bu değer 0 ya da sıfırdan küçük olacak ve bütün ceza terimi sıfır olacaktır. Eğer bir ya da birden fazla kısıtta ihlal varsa bu j çözümün her amaç fonksiyon değerine yansıyacaktır. $\overline{\Delta b}_i$ (E) terimi ise j çözümüne ait i.kısıt ile E seçkin kümedeki çözümlere ait i.kısıt sağ yan değerleri arasındaki farkların ortalamasıdır. Bu terimin sıfır olması demek hem j hem de diğer seçkin çözümlerin aynı çözüm olması demektir. Eğer böyle ise önceki adımlarda bu teşhis edilmek durumundadır. Buna göre bu terim hiçbir zaman sıfır olamaz. Formüldeki köşeli parantez içindeki terimde, $F_k^{(j)}$ ile j çözümün k. amaç fonksiyonu değeri ve $F_k^{(E)}$ ile E seçkin kümede bulunan çözümler arasında en iyi k.amaç fonksiyon değeri gösterilmektedir. Bu değer arasındaki fark alınarak en iyi amaç değeri ile çözüm arasındaki uzaklık ölçülmektedir. Burada ε, 0 ile 1 arasında küçük bir sayıdır. ε ve amaç fonksiyonlarının farkı arasında maksimum

değerin alınmasının sebebi, j çözümüne ait k amaç fonksiyon değerinin seçkin kümedeki çözümlere ait k amaç fonksiyon değerlerinden daha iyi olması durumunda ε'u seçmek ve sadece sağ yan değerlerinden kaynaklanan cezanın küçük bir parçasının gelmesini sağlamaktır. Böylece uygun olmayan çözüme ait bir amaç fonksiyon değeri seçkin çözümlerin ilgili amaç fonksiyon değerlerinden daha iyi ise bu amaca büyük bir ceza gelmesi önlenmiş olacaktır.

Uygun olmayan çözümün her amacı için cezalar hesaplandıktan ve bunların amaç değerlerine eklenmesinden sonra bulunduğu kümedeki kendinden sonra gelen çözüm ile karşılaştırılır. Eğer bu çözüm uygun ve cezalı çözümü alt ediyorsa seçilir. Alt edemiyorsa daha iyi sıra sayılı cezalı çözüm seçilir. Cezalı çözümden sonra gelen çözüm uygun değilse, bu çözüm içinde cezalar bulunarak amaç fonksiyon değerleri yeniden hesaplanır. Eğer bu çözüm birinci çözümü alt ediyorsa seçilir ama alt edilemiyorsa daha iyi sıra sayılı birinci çözüm alınır. Bir seçim yapıldıktan sonra kısıtsız yöntemde izlenenler uygulanır.

Önerilen yöntemdeki ceza fonksiyonu tek amaçlı optimizasyon problemleri için izlenen Smith, Tate ve Coit [39] yönteminde uygulanan uyarlı ceza fonksiyonuna benzemektedir. Fonksiyondaki oranda, payda bulunan terim aynı ama payda da bulunan terim farklıdır. Orana çarpım şeklinde gelen terimde farklıdır. Aslında bunların farklılığı problemin çok amaçlı olmasından ve önerilen yöntemde elde edilen seçkin kümedeki çözümlerden yararlanılmasından kaynaklanmaktadır. Örneğin Smith vd. [39] yönteminde orana gelen çarpım terimi o zamana kadar algoritmanın bulduğu en iyi amaç fonksiyonu değeri ile en iyi bulunan uygun çözümün amaç fonksiyonuna verdiği değer arasındaki farktır. Önerilen yöntemde ise bu terim uygun olmayan çözümün amaç fonksiyon değeri ile seçkin kümedeki çözümlerin bu amaca verdiği en iyi değer arasındaki farktır. Seçkin kümede uygun ve en iyi çözümler olduğuna göre hesaplanan iki terim birbirine yakındır.

Literatürde çok amaçlı optimizasyon problemleri için uyarlı ceza fonksiyonu önerisi yoktur. Vieria vd. [64] birbirine benzer iki yöntem önermiştir. Birinci yöntemde her uygun olmayan çözüme ait kısıt ihlal miktarının karesi ile sabit değer olarak belirlenen bir parametre çarpılarak toplanmıştır ve bir ceza terimi elde edilmiştir. Bu terim bütün amaç fonksiyonlarına eklenmiştir. İkinci yöntemde ise, birincisinde

elde edilen ceza terimi ile ihlal edilen kısıt sayısı iki yeni amaç olarak probleme eklenmiştir. İki yöntemde de yapılan ksıtlı çok amaçlı optimizasyon problemini kısıtsız duruma çevirmektir. Böylece NPGA yöntemi ile Pareto çözüm kümesi araştırılmıştır. Burada yapılan önceden belirlenen sabit bir ceza ile bütün uygun olmayan çözümlerin uygun hale getirilmesidir. Bu yöntemin amaç fonksiyon sayısını arttırması Pareto optimal çözümlerinin bulunmasını güçleştirmektedir [5]. Yöntemde NPGA uygulandığı için seçkin çözümler izlenmemektedir. Oysa seçkin çözümlerin izlenildiği algoritmaların daha iyi çözümler bulduğu bilinmektedir [31]. Douglas vd.'nin önerdiği yöntemde uyarlı ceza fonksiyonu kullanılmamıştır ve uygun olmayan çözümler önceden belirlenerek kısıt ihlal miktarları değerlendirilmektedir. Oysa önerilen algoritmada kümeleme işleminden sonra, uygun olmayan çözümün bir sonraki nesil için seçilme şansı ortaya çıkarsa ceza terimi hesaplamakta ve sadece o çözümü cezalandırmaktadır. Bu özellik hesap kolaylığı sağlamaktadır. Dolayısıyla önerilen yöntem uyarlı ceza fonksiyonu ile ilktir ve tek amaçlı optimizasyonda iyi sonuç veren Smith vd. [39] önerisinin çok amaçlı optimizasyona uyarlanması şeklinde olmaktadır.

4. UYGULAMALAR

Bu kesimde kısıtlı ve kısıtsız çok amaçlı karar verme problemleri için önerilen algoritmaların uygulamaları verilmiştir.

4.1. Kısıtsız Çok Amaçlı Karar Verme Problemi Uygulaması

Kısıtsız problemlerin çözümü için önerilen algoritma literatürde ismi KUR problemi olarak geçen ve Alman akademisyen Frank Kursawe [47] tarafından ortaya atılan optimizasyon probleminin çözümü için çalıştırılmıştır. Problemde üç tane karar değişkeni, iki tane amaç fonksiyonu vardır ve ikisinin de minimum yapılması istenmektedir. Bazı araştırmacılar karar değişkeni sayısını daha çok tutarak bu problemi çözmeye çalışmışlardır. Problemin matematiksel ifadesi

$$\text{Minimum } f_1 = \sum_{i=1}^{n-1} \left(-10 \exp\left(-0.2 \sqrt{x_i^2 + x_{i+1}^2} \right) \right)$$

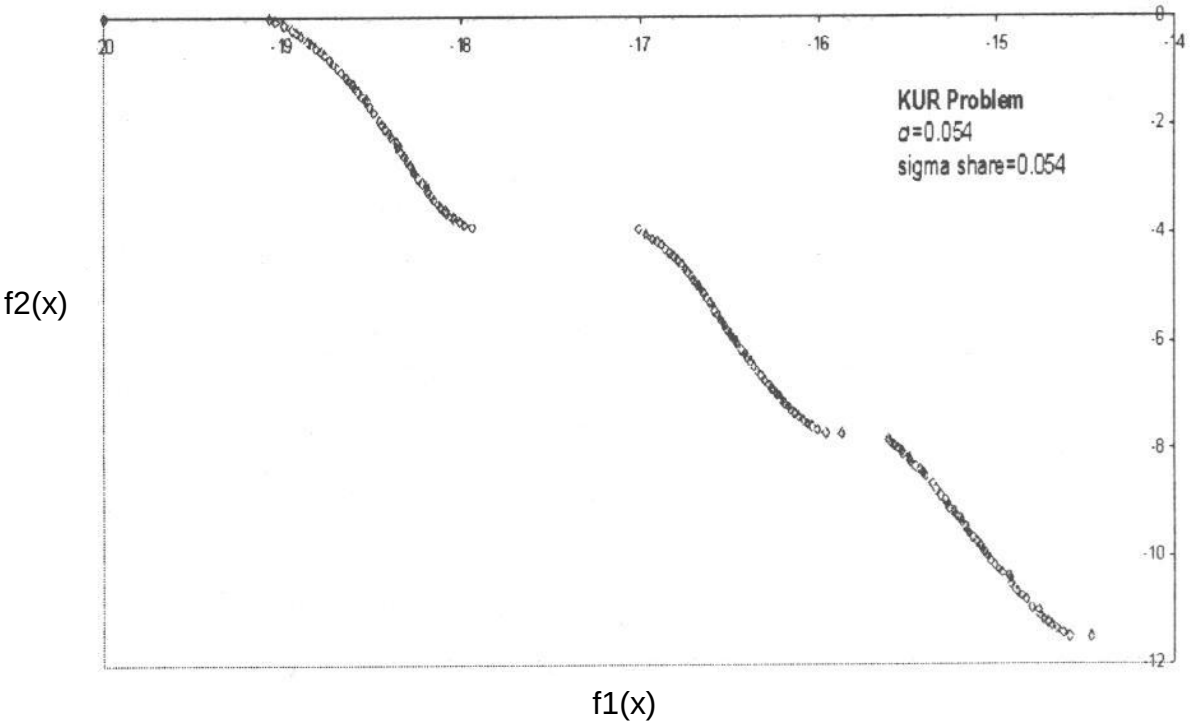
$$\text{Minimum } f_2 = \sum_{i=1}^{n} \left(|x_i|^{0.8} + 5 \sin(x_i)^3 \right)$$

$$n = 3 \ , \ -5 \le x_1, x_2, x_3 \le 5$$

olarak verilmiştir. Değişkenlere ait tanım aralıkları kısıtlı olduğu için bu problem kısıtlı çok amaçlı problem olarak kabul edilebilir ;ancak kısıtsız durum için çözülen problemde, değişkenler tanımlı olduğu aralıkta değer alabilecek şekilde bilgisayar programında kodlandığından, algoritma hiç bir zaman uygun olmayan çözümle karşılaşamaz. Bu durumda önerilen kısıtsız çok amaçlı GA ile çözüm araştırılabilir.

Bu probleme ait Pareto optimal çözüm kümesi önceden bahsedilen çok amaçlı optimizasyon algoritmalarını önerenler tarafından bulunmuştur. Kamiura vd. [48] kendilerinin geliştirdiği MOGADES (Multi-objective Genetic Algorithm with distributed Enviromental Scheme) algoritmasını kullanarak n=100 için ve Watanabe vd. problemi yine n=100 için ve NSGA-II (Sıralı Seçkin Alt Edilemeyen Genetik Algoritma) kullanarak çözmüşler. Costa [49] ise n=3 için ve NSGA-II kullanarak problemi çözmüştür. Bu çalışmada Costa tarafından kullanılan problemle aynı sayıda değişkene sahip olunduğu için onun sonuçları ile

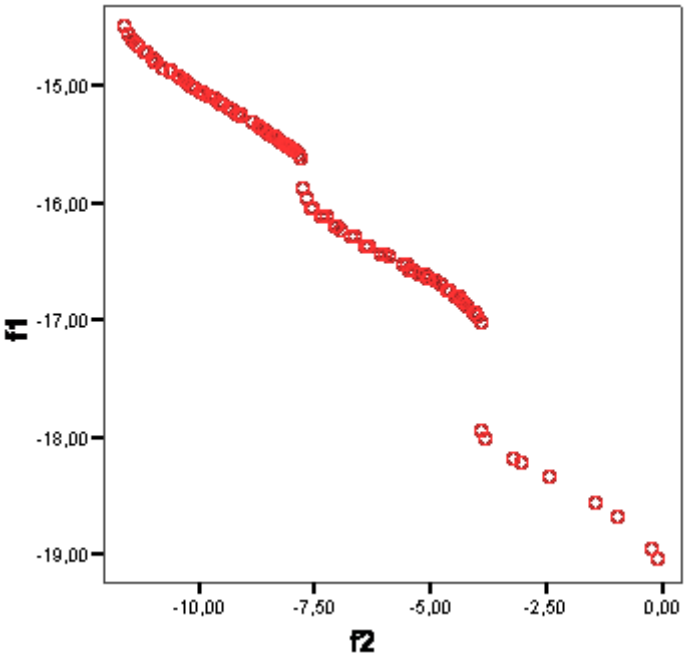
karşılaştırma yapılmıştır. Şekil 3'de Costa tarafından bulunan Pareto optimal çözümlerinin dağılımı görülmektedir.



Şekil 3. Costa tarafından elde edilen KUR problemine ait (n=3) Pareto optimal çözüm kümesi

Bu şekle göre Pareto optimal çözüm kümesi konkav bir şekle sahiptir. Çözümlerin birbirinden ayrı 3 kısımda toplanması problemin çözümünün bulunmasını daha da zorlaştırmaktadır.

Önerilen algoritmanın bulduğu Pareto optimal çözüm kümesi Şekil 4'de görülmektedir. Ana popülasyonda 100 tane çözüm vardır. Buna göre bu çözümlerden üretilen yeni popülasyonda da 100 tane çözüm vardır. Seçkin kümede ise 25 tane çözüm izlenmektedir. Genetik Algoritma operatörlerinden çaprazlama operatörü için beş nokta çaprazlama kullanılmıştır. Mutasyon operatörü için mutasyon olasılığı 0,05 alınmıştır.



Şekil 4. Önerilen kısıtsız çok amaçlı Genetik Algoritma'nın bulduğu Pareto optimal çözüm kümesi (100 ana çözüm)

Şekil 4'de görülen Pareto çözümler önerilen algoritma tarafından bulunmuş ve bu 100 çözüme ait fonksiyon değerleri Ek 1'de verilmiştir. Costa tarafından bulunan Pareto optimal çözümlerin (Şekil 3) gösterdiği dağılım birbirine benzemektedir. Buna göre önerilen algoritmanın Pareto optimal çözüm kümesini bulduğu açıktır. Elde edilen çözüm kümesindeki çözümlerin 1. ve 2. amaç fonksiyonlarına ait değerlerin temel istatistikleri Çizelge 1'de verilmiştir.

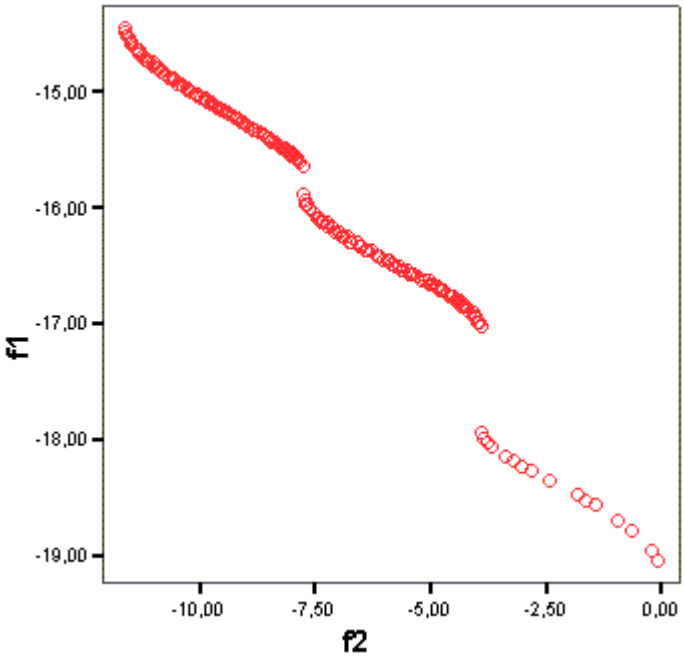
Çizelge 1. Önerilen kısıtsız çok amaçlı Genetik Algoritma'nın bulduğu Pareto optimal çözümlere ait istatistikler

	f_1	f_2
N	100	100
Ortalama	-15,99	-7,32
Medyan	-15,62	-7,80
Tepe Değeri	-19,05	-11,60
Standart Sapma	1,06	2,75
Minimum	-19,05	-11,60
Maksimum	-14,60	-0,05

Çizelge 1'e göre Pareto optimal çözümlere ait 1. amaç fonksiyon değerlerinin ortalaması -15.99, 2. amaç fonksiyon değerlerinin ortalaması ise -7.32'dir.

Standart sapmalara bakılırsa, 2. amaç fonksiyonuna ait olan sapmanın 1.amaç fonksiyonuna göre 2 kattan daha fazla olduğu görülmektedir.

Ana popülasyon büyüklüğünün 100 yerine 200 alınmasıyla önerilen algoritmanın elde ettiği Pareto optimal çözümler Şekil 5'te ve bu çözümlere ait temel istatistikler Çizelge 2'de verilmiştir.

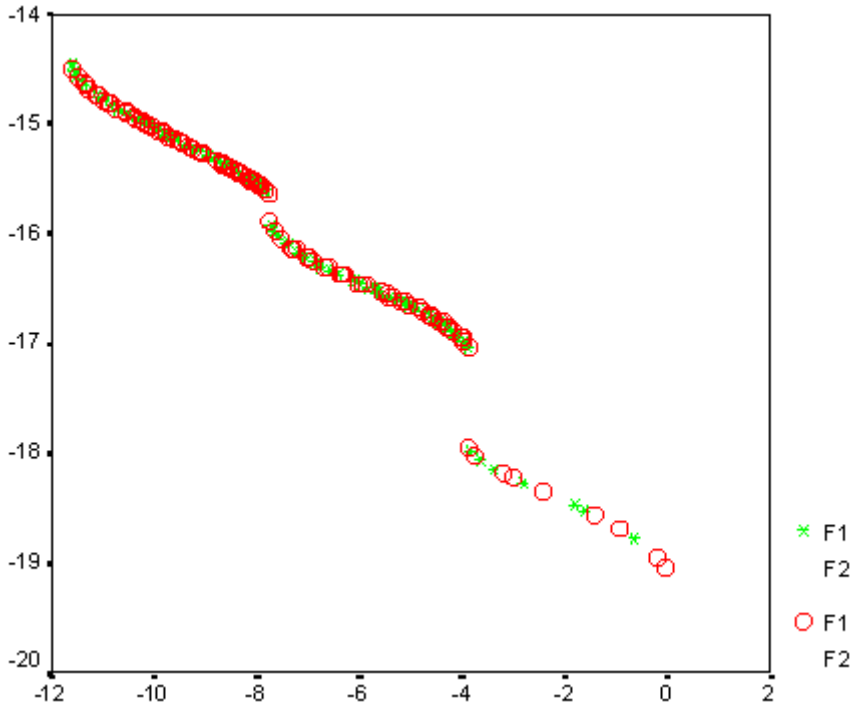


Şekil 5. Önerilen kısıtsız çok amaçlı Genetik Algoritma'nın bulduğu Pareto optimal çözüm kümesi (200 ana çözüm)

Çizelge 2. Ana popülasyonda 200 tane çözüm olduğunda Önerilen kısıtsız çok amaçlı Genetik Algoritma'nın bulduğu Pareto optimal çözümlere ait istatistikler

	f_1	f_2
N	200	200
Ortalama	-15,97	-7,37
Medyan	-15,60	-7,84
Tepe Değeri	-19,05	-11,63
Standart Sapma	1,02	2,69
Minimum	-19,05	-11,63
Maksimum	-14,45	-0,05

Elde edilen sonuçlara göre ana popülasyondaki çözüm sayısının arttırılması Pareto optimal çözümlere ait her iki amaç fonksiyonun hem ortalamalarının hem de standart sapmalarının daha iyi olmasını sağlamıştır. Buna göre algoritma ana popülasyon sayısına duyarlıdır. Her iki durum için (ana popülasyon 100 ve 200) elde edilen Pareto optimal çözüm kümelerinin beraber çizimi aşağıdaki Şekil 6'ta verilmiştir.



Şekil 6. Önerilen kısıtsız çok amaçlı Genetik Algoritma'nın bulduğu ana popülasyonda 100 ve 200 çözümün alınması ile elde edilen Pareto optimal çözüm kümelerinin beraber çizimi

Şekil 6'ta görüldüğü gibi her iki Pareto optimal çözüm kümesi birbiriyle örtüşmektedir. 200 tane ana çözüm alındığında algoritma bunlardan 200 tane yeni çözüm üretmekte, ana ve yeni çözümlerden 50 tane seçkin çözümü kayıt altına alarak bir sonraki nesil için ana popülasyon oluşturmaktadır. Bu durumda algoritma özellikle Şekil 6'ta görünen en alttaki bölgede daha fazla çözüm bulmuştur. Böylece amaç fonksiyon değerlerine ait ortalama ve standart sapmalarda küçülme elde edilmiştir. Ana popülasyondaki çözüm sayısının arttırılmasıyla algoritmanın önceden bulamadığı çözümleri de elde ettiği görülmektedir;ancak algoritmanın 100 tane ana çözümle optimal bölge araştırıldığında da elde edilen optimal kümenin yeterli olduğu anlaşılmaktadır.

Buna göre çok daha büyük problemlerde ana popülasyonun büyültülmesi fazla maliyet getirebileceğinden bu sayı küçük tutulabilir.

Sastry [65] tarafından hazırlanan teknik rapor yardımıyla Matlab programında aynı problem NSGA-II algoritmasıyla çözülmüştür. Çözümlere ait istatistikler Çizelge 3'de verilmiştir.

Çizelge 3. NSGA-II algoritmasının KUR problemi için bulduğu Pareto optimal çözümlere ait istatistikler

	f_1	f_2
N	235	235
Ortalama	-16,09	-6,57
Medyan	-16,21	-6,79
Tepe Değeri	-19,00	-11,45
Standart Sapma	1,46	3,53
Minimum	-20,00	-10,63
Maksimum	-1,45	11,54

Buna göre NSGA-II tarafından bulunan çözümlerin standart sapmaları daha büyüktür. NSGA-II Pareto optimal kümeye ulaşmıştır. Burada önerilen algoritmanın amaç fonksiyon değerleri bakımından birbirine daha yakın çözümler bulduğu söylenebilir.

4.2. Kısıtlı Çok Amaçlı Karar Verme Problemleri Uygulamaları

Bu kesimde iki farklı biçimiyle algoritmanın problemlere uygulanması yapılmış ve sonuçlar verilmiştir.

4.2.1. Uygun Olmayan Çözümlerin Değerlendirilmediği Algoritma Uygulaması

Kısıtlı çok amaçlı problemlerin çözümü için önerilen algoritma çok sayıda araştırmacının çözmeye çalıştığı ve literatürde ismi "Çok Amaçlı Sırt Çantası Problemi" olarak geçen probleme uygulanmıştır [50]. Ele alınan problemde kapasiteleri belli olan 2 sırt çantası ve her çantaya yerleştirilebilecek 500'er tane yük vardır. Yüklere 1'den 500'e kadar numara verilmiş ve bunlar çift olarak kabul edilmiştir. Örneğin; eğer birinci çantaya kendisine ait 100 numaralı yük yerleşmiş ise ikinci çantaya da kendisine ait 100 numaralı yük yerleşmek zorundadır. Her

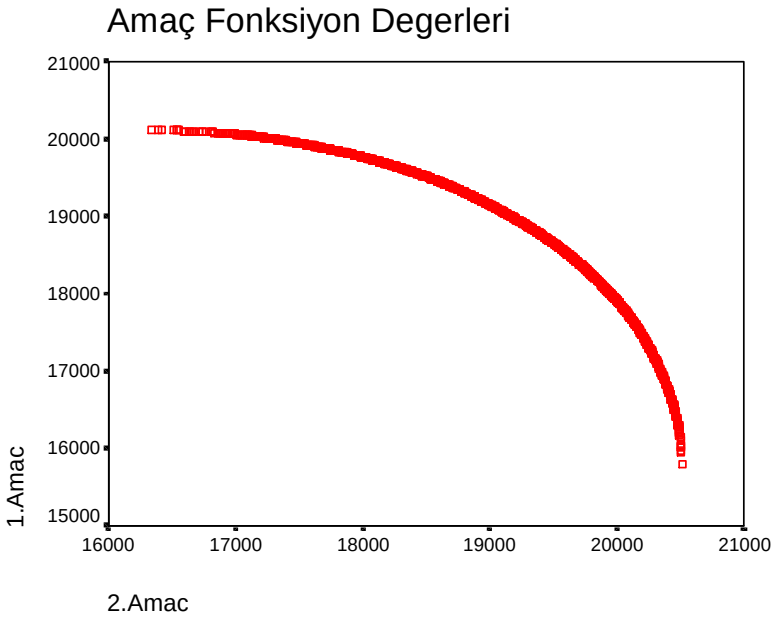
yükün yerleşmesiyle elde edilecek kazançlar bilinmektedir [51]. Buna göre problemde iki tane amaç fonksiyonu vardır. Bu amaçlar her iki çantaya da yerleştirilecek yüklerden elde edilecek kazançların maksimum yapılmasıdır. Bu yüklemede çanta kapasiteleri aşılamaz. Problemin matematiksel ifadesi maksimum $f1 = \sum_{i=1}^{500} p_{1i} x_i$ ve maksimum $f2 = \sum_{i=1}^{500} p_{2i} x_i$ şeklinde, kısıtlar ise

$$\sum_{i=1}^{500} w_{1i} x_i \le C_1 \ , \ \sum_{i=1}^{500} w_{2i} x_i \le C_2$$

şeklinde verilebilir. Burada x_i , 0 ya da 1 değerlerinden birini alabilir. 0 olması i.yükün yerleşmediği, 1 olması i.yükün yerleştiği anlamına gelmektedir. p_{1i} ve p_{2i} sırasıyla birinci çantaya ve ikinci çantaya i. yük alındığında elde edilecek kazanç miktarlarıdır. w_{1i} ve w_{2i} ise sırasıyla birinci ve ikinci çantaya i. yükün alınmasıyla çantada işgal edilecek hacimdir. C_1 ve C_2 1. ve 2. çantaların kapasitelerini göstermektedir.

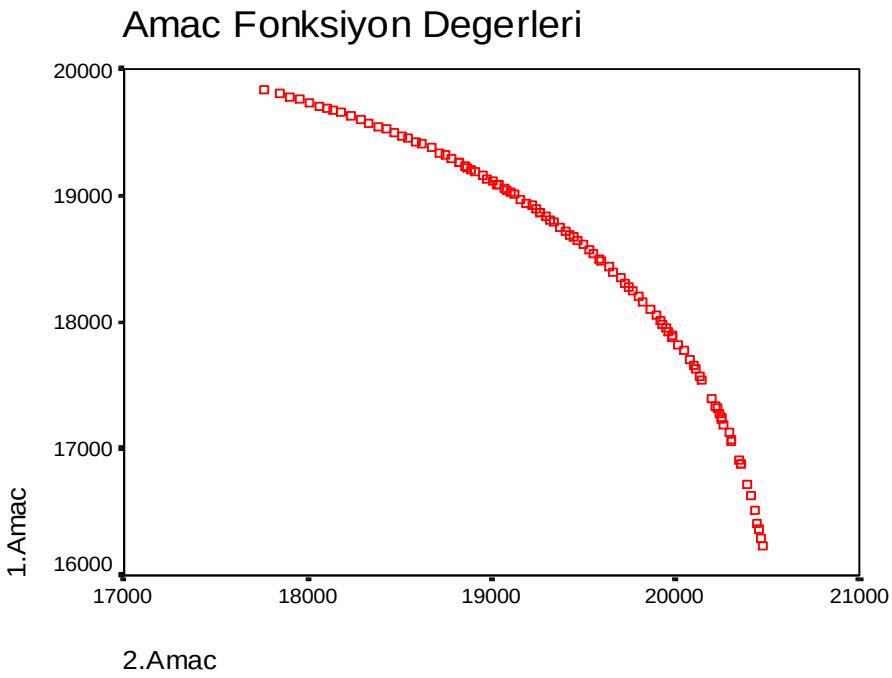
Çantalara konulacak yüklerin kapasiteyi aşmasıyla elde edilen çözümler uygun olmayan çözümlerdir. Bu çözümler algoritmada ceza fonksiyonu ile cezalandırılarak yeni çözümlerin aranmasında uygun olan çözümlere göre daha az rol almaları sağlanmaktadır.

Bu probleme ait Pareto optimal çözüm kümesi E. Zitzler ve L. Thiele tarafından “Güçlü Pareto Evrimsel Algoritma” yardımıyla bulunmuştur [52]. Problem için çantaların kapasiteleri, yükler ve bu yüklerin hacimleri yazarlara ait internet sitesinde [51] verilmiştir. Şekil 7’de Zitzler ve Thiele tarafından bulunan toplam Pareto optimal çözüm kümesindeki 1427 çözümün dağılımı görülmektedir. Bu şekle göre çözümlerin 1. amaç fonksiyonuna verdiği değerler küçüldükçe 2. amaç fonksiyonuna verdiği değerler büyümektedir. Amaç fonksiyonları arasında ödünleşim vardır.



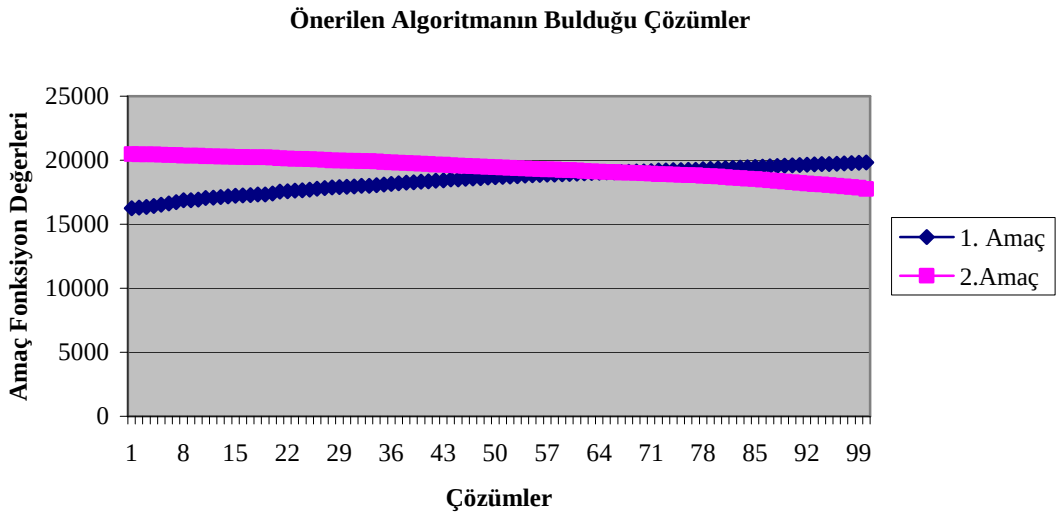
Şekil 7.Çok amaçlı sırt çantası problemi için Zitzler ve Thiele tarafından bulunan Pareto optimal çözüm kümesi (1427 çözüm)

Önerilen algoritmanın bulduğu Pareto optimal çözüm kümesi Şekil 8'de ve bu çözümlere karşılık gelen amaç fonksiyon değerlerinin değişimi Şekil 9'ta verilmiştir. Ana popülasyonda 100 tane çözüm vardır. Buna göre bu çözümlerden üretilen yeni popülasyonda da 100 tane çözüm vardır. Seçkin kümede ise 25 tane çözüm takip edilmektedir. Genetik Algoritma operatörlerinden çaprazlama operatörü için beş nokta çaprazlama kullanılmıştır. Diğer operatör olan mutasyon için ise mutasyon olasılığı olarak 0,05 kullanılmıştır.



Şekil 8. Çok amaçlı sırt çantası problemi için önerilen kısıtlı çok amaçlı Genetik Algoritma tarafından bulunan Pareto çözümler (100 ana çözüm).

Şekil 7'de görülen Pareto optimal çözümler ile önerilen algoritma tarafından bulunan optimal çözümlerin (Şekil 8) gösterdiği dağılım birbirine benzemektedir. Buna göre önerilen algoritmanın Pareto optimal çözüm kümesini bulmaktadır. Ayrıca bulunan çözümlerin çoğu Zitzler ve Thiele tarafından bulunan çözümlerle aynıdır. Şekil 9 ile iki amaç arasındaki ödünleşimin nasıl değiştiği anlaşılmaktadır. Bu değişim Ek 2 kesiminde verilen amaç fonksiyon değerlerinin kontrol edilmesiyle sayısal olarak görülmekte ve ödünleşimler buradan hesaplanabilmektedir. Böylece elde edilen Pareto optimal çözümler sayesinde karar vericiye birbirini alt edemeyen çözümler sunulabilir ve ödünleşimlerin dikkate alınmasıyla karar verme aşamasında yardımcı olunabilir.



Şekil 9. Önerilen kısıtlı çok amaçlı algoritmanın bulduğu Pareto optimal çözümlerine ait amaç fonksiyon değerlerinin değişimi

Elde edilen Pareto optimal çözüm kümesindeki çözümlerin 1. ve 2. amaç fonksiyonlarına ait değerlerin temel istatistikler Çizelge 4'te ve Zitzler ve Thiele tarafından bulunan Pareto optimal çözümlere ait temel istatistikler Çizelge 5'te verilmiştir.

Çizelge 4. Çok amaçlı sırt çantası problemi için ana popülasyonda 100 tane çözüm olduğunda önerilen kısıtlı algoritmanın bulduğu Pareto optimal çözümlere ait istatistikler

	f_1	f_2
N	100	100
Ortalama	18454	19385,10
Medyan	18685	19436,50
Tepe Değeri	16240	17762
Standart Sapma	982,72	755,38
Minimum	16240	17762
Maksimum	19832	20470

Çizelge 5. Çok amaçlı sırt çantası problemi için Zitzler ve Thiele tarafından bulunan Pareto optimal çözümlere ait istatistikler

	f_1	f_2
N	1427	1427
Ortalama	18688.56	19109.77
Medyan	18880	19293
Tepe Değeri	15796	20448
Standart Sapma	1046,68	1010.36
Minimum	15796	16331
Maksimum	20114	20511

Çizelge 4 ve 5'e göre Pareto optimal çözümlere ait amaç fonksiyon değerlerinin ortalamaları birbirine yakındır. Aynı şekilde diğer istatistikler arasında da fazla farklılık görünmemektedir. Önerilen algoritma 100 nesille Pareto optimal bölgesinden çözümlere ulaşmaktadır.

4.2.2. Uygun Olmayan Çözümlerin Ceza Fonksiyonu ile Değerlendirildiği Algoritma Uygulaması

Önerilen uyarlı ceza fonksiyon yönteminin iş-makine çizelgeleme problemlerinden birine uygulaması yapılmıştır. Bu nedenle sonraki kesimde iş-makine çizelgeleme problemleri hakkında bilgi verilecek ve uygulama sonuçları tartışılacaktır.

İş- makine çizelgeleme problemlerinde belli amaç ya da amaçlar doğrultusunda, n tane işin m tane makinede nerede ve hangi sırada işlem görmesi gerektiği bulunmak istenmektedir. İşlere ait operasyonlar vardır ve bunlar sırasına göre işlem görmesi gerekmektedir. Üretilecek olan parçaların montesi, sipariş işlemleri gibi işlerin rahatlıkla yapılabilmesi için her işe ait bitirilmesi gereken bir zaman vardır ve bu zaman en geç bitirilme zamanı olarak adlandırılır. Çizelgeleme sonunda eğer bir iş kendine ait en geç bitirilme zamanından sonra bitiriliyorsa, o iş geç kalmış bir iş olarak çözümde yer alır. Böylece iş- makine çizelgeleme problemlerinde geç kalmış iş sayısı ya da ortalama geç kalma zamanı minimum yapılmaya çalışılabilir. Bunun yanında makinelere ait ortalama boş kalma sürelerinin minimum yapılması, makinelerin çalıştığı ortalama zamanın minimum yapılması, erken bitirilen iş sayısının minimum yapılması gibi amaçlar iş- makine çizelgeleme problemlerinde sıklıkla kullanılan amaçlardır. Literatürde bu amaçlardan biri kullanılabileceği gibi bunlardan ikisi ya da daha fazlası da kullanılarak iş- makine çizelgeleme problemlerine çok amaçlı optimizasyon problemleri olarak yaklaşıldığı görülmüştür. "NP-hard" problemler sınıfına giren çok amaçlı iş- makine çizelgeleme problemlerini çözmek için sezgisel yöntemler kullanılmaktadır. He,Yag ve Tiger [53] , Low ve Wu [54], Van Laarhove, Aarts ve Lenstra [55] çok amaçlı benzetim tavlaması yöntemini; Barnes ve Chambers [56], Amico ve Trubia [57], Widmer [58] ise çok amaçlı tabu arama yöntemini; Bigel ve Davern [59], Della vd. [60] ve Sakawa vd. [61] çok amaçlı genetik algoritma

yöntemini kullanmıştır. Bu yöntemlerin birbirine karşı üstünlükleri kuramsal olarak ispatlanmamıştır.

Önerilen çok amaçlı genetik algoritma, Low, Yuklink ve Wu [62] tarafından çok amaçlı tabu algoritması ile çözülen 4 iş ve bu işlere ait 13 operasyonun toplam 3 makineye çizelgelenmesi problemine uygulanmıştır. Problemde üç amaç vardır. Birincisi, her işe ait son operasyonunun bittiği, yani işlerin tamamlandığı ortalama sürenin minimum yapılması, ikincisi, geç kalan işlere ait ortalama gecikme sürelerinin minimum yapılması ve son amaç ise makinelerin ortalama boş kalma sürelerinin minimum yapılmasıdır. Amaçlar arasında öncelik yoktur.

Önerilen algoritmanın kümeleme yöntemi ile çeşitlendirme (diversification) özelliğinin güçlü olması, tabu algoritması ile karşılaştırma yapılmasının anlamlı olacağını düşündürmüştür. Bu özellik tabu algoritmasının kendisine özgü bir algoritma olmasını sağlayan başlıca özelliktir. Low, Yuklink ve Wu [62] tarafından çok amaçlı tabu algoritması ile çözülen probleme ait veriler Çizelge 6'ta verilmiştir.

Çizelge 6. 3 makine 4 iş problemine ait veriler

İş no	Operasyon no	Hazırlık ve İşlem Zamanı						Bitirilmesi Gereken Zamanı
		Makine 1		Makine 2		Makine 3		
		Hazırlık	İşlem	Hazırlık	İşlem	Hazırlık	İşlem	
1	1	4	4	-	-	4	5	37
	2	2	5	1	7	1	5	
	3	1	7	1	4	-	-	
2	1	-	-	4	3	-	-	31
	2	1	7	2	3	0	2	
	3	-	-	-	-	3	7	
	4	4	7	-	-	-	-	
3	1	-	-	0	7	2	8	28
	2	1	8	-	-	3	5	
	3	2	9	-	-	-	-	
	4	-	-	3	4	-	-	
4	1	1	7	-	-	-	-	25
	2	-	-	2	8	3	4	

Çizelge 6'tan da anlaşılacağı gibi 1. işe ait 3, 2.işe ait 4, 3.işe ait 4 ve 4. işe ait 2 tamamlanması gereken operasyon vardır. Bu operasyonların bazıları bir, bazıları iki, bazıları üç makinede de yapılabilmektedir. Örneğin 2. işin 1.operasyonu sadece ikinci makinede yapılabilmektedir. Dolayısıyla bu operasyon başka bir

makinede yapılamaz. Bu operasyonun çizelgeleme işi bu durumda kolay gibi görünmekte ancak amaçlar için operasyonun ne zaman yapılacağı önem teşkil etmektedir.

Probleme ait 3 amaç fonksiyonu sırasıyla f1 ; işlerin tamamlandığı ortalama süre, f_2; geç kalan işlere ait ortalama gecikme süresi ve f_3 ; makinelerin ortalama boş kalma süresi olarak gösterilirse, Low, Yuklink ve Wu tarafından bulunan alt edilemeyen çözümler arasında her amaç için elde edilen en kötü değerler Çizelge 7'de verilmiştir.

Çizelge 7. Low, Yuklink ve Wu tarafından bulunan alt edilemeyen çözümler arasında her amaç için elde edilen en kötü değerler

	f_1	f_2	f_3
En kötü Amaç Değeri	48.75	18.50	17.67

Low, Yuklink ve Wu tarafından problemin tabu araması yöntemi ile bulduğu sonuçlar Çizelge 7'de verilmiştir.

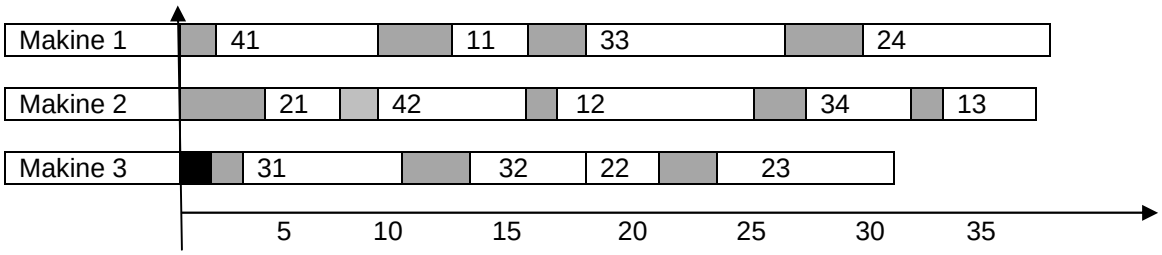
Çizelge 8. Çok amaçlı tabu arama algoritmasının 3 makine 4 iş problemi için bulduğu Pareto optimal çözümler

Çözümler	f_1	f_2	f_3
1	31	2.75	3
2	28	3	5
3	32.5	2.5	5.33
4	34.25	6	2.67

Yazarların tabu araması ile buldukları ve Çizelge 8'de görülen çözümler, birbirini alt edemeyen çözümlerdir. Makalede 1. çözüme ait çizelge de vermiştir. Bu çizelge Şekil 10'da görülmektedir. Şekilde kutu içinde iş ve operasyonu gösterilmektedir. Örneğin 41, 4.işin 1.operasyonunu göstermektedir. Kutularda görülen gri renkli alan, o iş ve operasyona ait hazırlık süresini, beyaz alan ise makinede göreceği işlem süresini göstermektedir.

Elde edilen şekilden amaç fonksiyonları hesaplanabilmektedir. Çizelge 9'daki 1.çözüme ait 1. amaç olan işlerin tamamlandığı ortalama süre 1, 2, 3 ve 4. işlerin son operasyonlarının bitim zamanı olan sırasıyla 38, 37, 32 ve 17'nin ortalaması olan 31 değeridir. 2.amaç olan geç kalan işlere ait ortalama gecikme süresi için işlerin bitim süresi ile bitirilmesi gereken süre karşılaştırılmalıdır. 1.iş en geç 37

zaman birimi içinde bitmelidir, bu işte gecikme yoktur. 2.iş 38 zaman biriminde bitmiş oysa en geç 31'de bitmesi istenilmektedir. 7 birimlik gecikme vardır. 3.iş 32'de bitmiş oysa 28'de bitmesi istenmiştir. 4 birimlik gecikme vardır. 4. iş ise 17'de bitmiş oysa 25'e kadar tamamlanabilirdi. Bu işte gecikme yoktur. Toplam gecikme 11 birimdir. 4 iş olduğu için 11'in 4'e bölünmesiyse 2.amacın değeri 2.75 bulunur. Son amaç makinelerin ortalama boş kalma süresiydi. 3.makinenin toplam 8 birimlik boş kalma süresi vardır. 2. makinede ise 1 birimlik gecikme vardır. Toplam 3 makine olduğu için ortalama boş kalma süresi 3 olmuştur.



Şekil 10. 3 makine 4 iş çizelgeleme problemi için Tabu arama yöntemi ile bulunan ve amaç fonksiyon değerleri $f_1 = 31$, $f_2 = 2.75$ ve $f_3 = 3$ olan çözüme ait çizelgeleme.

Makalede elde edilen çözümler global kriter yaklaşımı (global criterion approach) ile değerlendirilmiştir. Bu kriter

$$d(x) = \sum_{i \in X} w_i \left[\frac{f_i(x) - f_i^{+}}{f_i^{-} - f_i^{+}} \right]$$

formülü ile verilir [63]. Bütün alt edilemeyen çözümler için bu kriter ile bulunan değerlerin hangisi küçükse o çözümün daha iyi olduğu sonucuna varılmaktadır.

Formüldeki w_i katsayısı amaç fonksiyonlarının arasındaki önceliğe göre değer alan ağırlıklardır. Önceliğin olmaması durumunda katsayılar aynı olur. f_i^{+} alt edilemeyen çözümler arasında i. amaç fonksiyonunun almış olduğu en küçük değerdir. f_i^{-} ise alt edilemeyen i. amaç fonksiyonuna ait en kötü değerdir. Bu formülasyona göre global kriter, bir çözümün en iyi ve en kötü amaç değerlerine göre hesaplanan bir uzaklık (distance) fonksiyon değeridir. İlgilenilen problem için

$$d(x) = \frac{1}{3}\left[\frac{f_1(x) - 28}{48.75 - 28} \right] + \frac{1}{3}\left[\frac{f_2(x) - 2.5}{18.5 - 2.5} \right] + \frac{1}{3}\left[\frac{f_3(x) - 2.67}{17.67 - 2.67} \right]$$

şeklinde yazılabilir. Formülde görülen paydaki sayılar 28, 2.5 ve 2.67 alt edilemeyen çözümler arasında ilgili amaç fonksiyonun aldığı en küçük değerdir. Paydada bulunan sayılardan 48.75, 18.5 ve 17.67 ilgili amaç fonksiyonunun en

kötü değerleridir. Amaçlar arasında öncelik olmadığı için w_i değerleri 1/3 olarak alınmıştır.

Alt edilemeyen çözümlere ait global kriter yaklaşımının sonuçları Çizelge 8'de verilmiştir.

Çizelge 9. 3 makine 4 iş problemi için bulunan
Tabu araması çözümlerine ait Global Kriter değerleri

Çözümler	Global Kriteri
1	0.0602
2	0,0616
3	0,1302
4	0,1716

Bu sonuçlara göre 1.çözüm çözümler arasında en iyi olan denilebilir.

Kısıt durumunda önerilen genetik algoritmanın 4 iş 3 makine çizelgeleme problemine uygulanmasıyla elde edilen çözümler Çizelge 10'da görülmektedir.

Çizelge 10. 3 makine 4 iş problemi için önerilen
çok amaçlı genetik algoritmanın bulduğu çözümler

Çözümler	f_1	f_2	f_3
1	32.50	3	4.66
2	28.75	3,25	3,67
3	32.50	2.5	5.33
4	31.00	2.75	3

Tabu araması ile bulunan çözümlerle, önerilen genetik algoritmanın bulduğu çözümler karşılaştırıldığında, amaç fonksiyon değerleri bakımından alt edilemeyen çözümler arasındaki 3 ve 4 numaralı çözümlerin Low, Yuklink ve Wu tarafından bulunan çözümlerden ikisiyle aynı olduğu görülmektedir. Bu çözümlerden 3 numaralı çözüme ait çizelgeleme tabu araması yönteminin bulduğu çözüme ait çizelgelemesiyle de aynıdır, ancak 4 numaralı çözümün çizelgelemesi tabu aramasının bulduğu çözümünki ile aynı değildir.

Farklı iki çözüme ait global kriter değerleri sırasıyla 1. ve 2. çözümler için 0.127 ve 0.050 olarak bulunmuştur. Bu kriter hesaplanırken tabu aramasında bulunan çözümlere ait en iyi ve en kötü amaç değerleri kullanılmıştır. Böylece tabu çözümleri ile karşılaştırma daha sağlıklı yapılabilmektedir. 1 numaralı çözümün tabu araması yönteminin bulduğu bir çözüm tarafından alt edildiği görülmektedir, ancak 2 numaralı çözüm (f_1 = 28.75, f_2 = 3,25, f_3 = 3,67) alt edilememektedir.

Global kriteri yaklaşımı ile bu çözümün hem tabu araması yönteminin hem de önerilen genetik algoritmanın bulduğu çözümler arasında en iyi olduğu söylenebilir. Bu çözüme göre birinci makinede sırasıyla 4.işin 1. operasyonu, 1.işin 2. operasyonu.3.işin 3. işin 3.operasyonu yapılmalı ve 2 birimlik boş zaman makine boş bırakılıp 2.işin 4.operasyonu yapılmalı ve toplam 40 zaman birimi sonunda birinci makinenin işi bitmelidir. İkinci makinede sırayla 3. ve 2. işlere ait 1.operasyonlar yapılmalı ve 1 zaman birimi makine boş bırakılmalı ve ardından 1.işin 3.operasyonu, 2.işin 2.operasyonu ve son olarak 3.işin 4.operasyonu yapılmalı ve 32 zaman birimi sonunda makine durdurulmalıdır. Son makine olan 3.makinede sırasıyla 1.işin 1.operasyonu, 3.ve 4. işlerin 2.operasyonları ve 2.işin 3.operasyonu yapılmalı ve 33 zaman birimi sonunda makine durdurulmalıdır.

Önerilen algoritma tabu araması yönteminin bulduğu çözümlere ulaşmıştır. Tabu çözümleri tarafından alt edilemeyen bir çözüm de elde edilmiştir. Bu çözüm tabu çözümlerini alt edemediği için, çözümün kalitesinin incelenmesi amacıyla global kriteri yaklaşımından faydalanılmıştır. Bu çözüme ait global kriteri 0,050 olarak bulunmuştur. Tabu çözümlerine ait global kriter değerlerinin olduğu Çizelge 8'de verilen değerlerle karşılaştırılan bu değerin hepsinden küçük olduğu görülmüştür. Böylece çözümün tabu çözümlerinden daha iyi bir çözüm olduğu sonucuna varılmıştır.

5. SONUÇ ve TARTIŞMA

Bu çalışmada çok amaçlı karar verme problemlerinde kısıtsız ve kısıtlı problemler için Genetik Algoritma temelli yöntemler önerilmiştir. Kısıtlı problemler için önerilen algoritmalar, kısıtsız durum için önerilen algoritmaya benzemekte sadece uygun olmayan çözümlerin değerlendirilmesi aşamasında farklılık göstermektedir. Önerilen algoritmalarda GA'nın temel parametresi olan çaprazlama için 1, 2, 3, 4 ve 5 noktadan çaprazlama uygulanmış, 4 ve 5 noktadan çaprazlama ile daha iyi sonuçlar elde edilmiştir. Bu nedenle 5 noktadan çaprazlama uygulanmıştır. Literatürde de 5 noktadan çaprazlamanın iyi sonuçlar verdiği belirtilmektedir [3,5]. GA'nın parametresi olan mutasyon olasılığı ise 0,05 olarak alınmıştır. Literatürde bu olasılığın küçük olması gerektiği vurgulanmıştır [3,5].

Kısıtsız durum için önerilen algoritmanın oluşturulmasından önce literatürde çok sayıda uygulaması olan algoritmalar incelenmiştir. Bu algoritmaların iyi ve eleştirilen yanları değerlendirilerek kısıtsız durum için önerilen algoritma geliştirilmiştir. Kısıtsız durum algoritmasında ana popülasyon ve yeni popülasyon bireyleri amaç fonksiyon değerleri bakımından önce kendi kümeleri içinde karşılaştırılarak sıra sayısı almakta ve sonra diğer kümeye ait çözümlerle karşılaştırılarak sıra sayıları güncellenmektedir. Böylece seçkinlik güçlendirilmektedir. Seçkinliğin izlenmesi için ayrıca bir parametre kullanılmamaktadır. Bu özellikler önemlidir, çünkü seçkin çözümlerin izlenmediği algoritmalarda (VEGA [25], MOGA [26], NSGA [29],NPGA [30]), Pareto optimal çözüm kümesine yaklaşımın yavaş olduğu, amaçlara ait uç değerlere yaklaştıkları ve "niche" tekniği ile yapılan çeşitlendirme sonucunda parametre büyüklüğüne çok bağımlı oldukları bilinmektedir [31]. Önerilen algoritma popülasyon temelli bir algoritmadır ve çeşitliliği sağlamak amacıyla hem ana hem de yeni popülasyondan alınan çözümler kümelenerek bir sonraki ana popülasyon için değerlendirilmektedir. Kümeleme için herhangi bir parametre kullanılmamaktadır. SPEA [34] yönteminde kümeleme, seçkin kümedeki çözüm sayısının aşırı fazlalaşmasını önlemek amacıyla kullanılmaktadır. Bu amaçla sadece bu kümede bulunan çözümlere uygulanır. Seçilen çözümler seçkin çözümler olarak alınır; ancak atılan çözümlerde seçkin olduğu için önemli çözümlerin kaybedilmesi durumunda algoritma daha iyi çözümlere asla ulaşamayabilir. Önerilen

algoritmada kümeleme seçim stratejisinin bir parçasıdır. Ayrıca seçkin çözümlerin konulduğu kümenin aşırı dolması durumu yoktur, çünkü her adımda seçkin kümede alt etme kriterine göre alt edilen çözümler elenmektedir. Yine de olması gerekenden fazla çözüm seçkin kümede yer alırsa, en yaşlı (en önce kümeye giren) çözümler seçkin kümeden silinir. Bu sayede eğer bir çözüm birinci nesilde seçkin ise bu statüsü bozulana dek seçkin kümede kalır ya da yeni seçkin çözümlere yer kalmadığında, kendisinden de uzun süre faydalanılmasından dolayı seçkin kümeden çıkar.

Optimizasyon algoritmalarında seçim tekniği önemlidir. NSGA [29] ve NSGA – II ([32],[33]) yöntemlerinde her çözüm için diğer çözümleri alt etme durumuna göre sıra sayısı verilmektedir. Önerilen algoritmada da bu işlem yapılmaktadır ve buna ek olarak kümelemeden sonra oluşan kümelerin de sıra sayısı hesaplanmaktadır. Bu sayı basitçe kümede bulunan çözümlere ait sıra sayılarının toplamı olarak elde edilmekte ve sıra sayısı büyük olan kümelerden daha çok çözüm seçilmektedir. Böylece çözümler hem tek başına hem de bulunduğu komşulukla birlikte değerlendirilmektedir.

Kısıtlı çok amaçlı problemlerde önerilen Genetik Algoritma kısıtsız durumla benzer işlemektedir. Algoritma seçilecek olan çözümün uygun olmayan çözüm olması durumunda ek olarak değerlendirme yapmaktadır. Bu değerlendirme kümeleme sonrasında olabilir. Kümelerden çözüm seçerken ilk defa uygun olmayan çözümle karşılaşılırsa bu çözüm tutulur. Sonraki seçimlerde bir tane daha uygun olmayan çözümle karşılaşılırsa önceki çözüm ile yarıştırılarak bir tanesi seçilir ve bir sonraki nesil için uygun olmayan çözüm seçilmiş olur. Uygun olmayan çözümlerin seçiminde önce ihlal edilen kısıt sayısına bakılır ve az sayıda ihlal eden seçilir. Eğer aynı sayıda kısıt ihlal ediyorlarsa, bu çözümlerin seçkin kümeye uzaklıkları elde edilir. Küçük uzaklığa sahip olan çözüm seçilir. Böylece sonraki çözümlerin oluşumunda kullanılacak uygun olmayan çözümün seçkin çözümlere yakın olması sağlanmaktadır.

Kısıtlı durum için önerilen ikinci algoritma uyarlı ceza fonksiyon temellidir. Çok amaçlı optimizasyon yöntemlerinde ceza fonksiyonu uygulaması literatürde yoktur. Önerilen algoritmalarda tek amaçlı problemler için statik ve uyarlı ceza fonksiyon

uygulamaları vardır. Bunlardan Smith vd. [39] tarafından önerilen uyarlı ceza fonksiyonu iyi sonuçlar vermiştir. Önerilen algoritmadaki uyarlı ceza fonksiyonu Smith vd.'nin tek amaçlı opimizasyon için bulduğu fonksiyonun çok amaçlı optimizasyon için değiştirilmesiyle elde edilmiştir.

Bundan sonra yapılacak çalışmalar, önerilen algoritmaların literatürdeki başka problemlere de uygulanması ve çeşitli değişikliklerle bu algoritmaların geliştirilmesi olabilir. Bu çalışma ile istatistiksel tekniklerin optimizasyon algoritmalarında çok önemli yeri olan seçim stratejisinde kullanılabileceği gösterilmiştir. Gelecek çalışmalarda diğer istatistiksel yöntemler yine seçim stratejisinin önemli bir parçası olarak kullanılabilir.

Uygulamaların sonuçlarına göre önerilen algoritmalar iyi şekilde işlemekte, Pareto optimal çözüm bölgesine yakınsamakta, nesil sayısı arttıkça farklı alt edilemeyen çözümler bulmakta ve nesil sayısının az olması durumunda bile Pareto optimal çözümlere ulaşabilmektedir.

KAYNAKLAR

[1] Narula,S.C., Wellington, J.F., 2002. Multiple Objective Optimization in Statistics, International Transactions in Operational Research (9), pp 415-425.

[2] Winker, P., Gilli,M.,2004, Applications of Optimization Heuristics to Estimation and Modelling Problems, Computational Statistics & Data Analysis 47(2),pp. 211-223.

[3] Reeves, C.R., Modern Heuristic Techniques for Combinatorial Problems, John Wiley & Sons, Inc., New York, NY, 1993.

[4] Jaszkiewicz, A., 1998, How to Solve It : Modern Heuristics, Springer, New York.

[5] Deb K., 2001, Multiobjective Optimization Using Evolutionary Algorithms, Wiley & Sons, England.

[6] Coello, C.A., 1999, A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques, Knowledge and Information Systems. An International Journal, 1(3),pp.269-308.

[7] Zitzler, E., Deb, K and Thiele, L., 2000, Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 8(2),pp.173-195.

[8] Mietttinen K., 1999, Nonlinear Multiobjective Optimization, Kluwer, Boston.

[9] Haimes, Y.Y., Lasdon L. S., Wismer, D. A., 1971, On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Transactions on Systems, Man, and Cybernetics 1 (3), 296-297.

[10] Charnes, A., Cooper, W. and Ferguson, R.,1955, Optimal Estimation of Executive Compansation by Linear Programming, Management Science , 1(2),138-151.

[11] Ignizio, J.P., 1978, A Review of Goal Programming : A tool for Multiobjective Analysis,Journal of Operatins Research Society,29 (11), 1109-1119.

[12] Lee, S.M., 1972, Goal Programming for Decision Analysis, Auerbach Publishers, Pa, US.

[13] Schniederjans M.J., 1995, Goal programming: Methodology and Applications, Kluwer Academic Publishers, Norwell, USA.

[14] Deb K., 1995, Optimization for Engineering Design: Algorithms and Examples, Prentice Hall,New Dlphi.

[15] Winston, W.L., 1994, Operation Research: Applications and Algorithms, Third EditionDuxbury Press,Ca,USA.

[16] Holland, J.H., 1975, Adaptation in Natural and Artificial Systems, MIT press, Ann Arbor, MI, USA.

[17] Goldberg,D.E.,1989, Genetic Algorithms for Search, Optimizations and Machine Learning, Addison Wesley, Reading, MA.

[18] Michalewicz, Z., 1992, Genetic Algorithms + Data Structures = Evolutionary Programs, Springer- Verlag, Berlin.

[19] Gen, M, Chen, R., 1997, Genetic Algorithms and Engineering Design, NewYork, Wiley.

[20] Mitchell, M., 1996, Introduction to Genetic Algorithms, MIT press, Ann Arbor, MI, USA

[21] Vose, M.D., 1999, Simple Genetic Algorithms: Foundation and Theory, MIT press, Ann Arbor, MI, USA

[22] Deb,K., 1999, Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems, Evolutionary Computation, 7(3), pp.205-230.

[23] Baker, J.E.,1985,Adaptive Selection Methods for Genetic Algorithms, In Proceedings of an International Conference of Genetic Algorithms and Their Applications,pp.101-111.

[24] Joines J., Houck C., 1994,On the use of non-staionary penalty functions to solve nonlinear constrained optimization problems with GAs, the Proceedings of the First IEEE Conference on Evolutionary Computation, pp. 98-108.

[25] Schaffer, J.D, 1985, Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, In Proceedings of the First International Conference on Genetic Algorithms, pp. 93-100.

[26] Fonseca, C.M., Fleming, P.J., 1993, Genetic Algorithms for Multiobjective Optimization: Formulation, discussion and generalization, In Proceedings of the Fifth International Conference on Genetics Algorithms, pp. 416-423.

[27] Goldberg, D.E., Richardson, J., 1987, Genetic Algorithms with Sharing fot Multimodal Function Optimization, In Proceedings of the First International Conference on Genetic Algorithms and Their Applications, pp. 41-49.

[28] Deb, K., Goldberg, D.E., 1989, An Investigation Niche and Species Formation in Genetic Function Optimization, In Proceedings of the Third International Conference on Genetic Algorithms, pp. 42-50.

[29] Srinivas,N., Deb, K.,1994, Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, Evolutionary Computation, 2(3),pp.221-248.

[30] Horn,J., Nafpliotis,N., Goldberg, D.E., 1994, A Niched Pareto Genetic Algorithm for Multiobjective Optimization, In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, vol 1, pages 82-87, Piscataway, New Jersey.

[31] Konak, A., Coit, D.W, Smith, E.A., 2006, Multi-objective Optimization using Genetic Algorithms: A tutorial, Reliability Engineering & System Safety, vol. 91, pp. 992-1007.

[32] Deb, K., Goel, T., 2001, Controlled Elitlist Non-dominated Sorting Genetic Algorithms for Better Convergence, In Proceedings of the First Conference on Evolutionary Multi-Criterion Optimization (EMO-2001), pp. 61-81.

[33] Deb, K., Goel, T., 2001, A Hybrid Multi-objective Evolutionary Approach to Engineering Shape Design, In Proceedings of the First Conference on Evolutionary Multi-Criterion Optimization (EMO-2001), pp. 385-399.

[34] Zitzler, E.,Thiele,L.,1999, Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, IEEE Transactions on Evolutionary Computation, 3(4), pp.257-271.

[35] Zitzler,E.,Laumanns, M., Bleuler, S., 2004, A Tutorial on Evolutionary Multiobjective Optimization, Metaheuristics for Multiobjective Optimisation, Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535, pp. 3-37.

[36] Knowles, J.D., Corne, D.W., 2000, Approximating the Non-dominated Front using the Pareto Archived Evolutionary Strategy, Evolutionary Computation Journal 8(2), pp. 149-172.

[37] Coello,C.A., 2000, Treating Objectives as Constraints for Single Objective Optimization,Engineering Optimization, 32 (3),pp. 275-278.

[38] Joines J., Houck C., 1994,On the use of non-staionary penalty functions to solve nonlinear constrained optimization problems with GAs, the Proceedings of the First IEEE Conference on Evolutionary Computation, pp. 98-108.

[39] Smith E.A., Coit D.W., Tate D.M., 1996, Adaptive Penalty Methods for Genetic Optimization of Constrained Combinatorial Problems, the Journal on Computing, vol 8 (2).

[40] Gen M., Cheng R., 1996, Interval programming using genetic algorithms, Proceedings of the Sixth International Symposium on Robotics and Manufacturing, Montpellier, France.

[41] Coello C.A., 2000, Constraint – Handling Using an Evolutionary Multiobjective Technique, Civil Engineering and Environmental Systems, vol. 17,pp. 319-346.

[42] Homaifar A., Qi C., Lai S., 1994, Constrained optimization via genetic algorithms, Simulation, vol.62 (4), pp. 242-254.

[43] Jimenez, F.,Verdegay, J.L.,Gomez, S.,A.,F., 1999, Evolutionary Techniques for Constrained Multiobjective Optimization Problems, In Proceeding of Workshop on Multi-Criterion Optimization using Evolutionary Methods held at Genetic and Evolutionary Computation Conference, pp.115-116.

[44] Deb K., 2000, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering, vol.186 (2-4), pp.311-338.

[45] Coello C.A., 2000, Constraint – Handling Using an Evolutionary Multiobjective Technique, Civil Engineering and Environmental Systems, vol. 17,pp. 319-346.

[46] Oyama, A., Shimoyama, K., Fujii, K., 2005, New Constraint-handling Method for Multi-objective Multi-constraint Evolutionary Optimization and Its Application to Space Plane Design, Evolutionary and Deterministic Methods and Design, EUROGEN, Technical Report, 13 p.

[47] Kursawe, F., 1991, A variant of evolution strategies for vector optimization, Parallel Problem Solving from Nature 1st Workshop (PPSN I), vol. 496 of Lecture Notes in Computer Science, Springer-Verlag, pp. 193-197, Berlin, Germany.

[48] Kamiura, J., Hiroyasu,T., Miki, S., Watanabe, M., 2002, MOGADES: Multi-Objective Genetic Algorithm with Distributed Environment Scheme, Second International Workshop on Intelligent Systems Design.

[49] Costa,L., Oliveira,P., 2005, An elitist genetic algorithm for multiobjective optimization, MIC'2001 - 4th Metaheuristics International Conference.

[50] Jaszkiewicz, A., 2004, On the Computational Efficiency of Multiple Objective Metaheuristics. The Knapsack Problem Case Study, European Journal of Operational Research, Vol. 158, No. 2, pp. 418—433.

[51] http://www.tik.ee.ethz.ch/sop/download/supplementary/testProblemSuite/

[52] Zitzler, E., Thiele, L., 1999, Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation, 3(4), pp.257-271.

[53] He Z, Yang Ti Tiger A., 1996, An Exchange heuristic immedded with simulated annealing for due-dates job shop scheduling. Europen Journal of Operations Research 91, pp. 99-117.

[54] Low C, Wu T.H., 2001, Mathematical modelling and heuristic approaches to operation scheduling problems in an FMS enviroment. International Journal of Production Research 39(4) , pp. 689-708.

[55] Van Laarhoveni P., Lenstra, A.E., 1992, Job shop scheduling by simulated annealing. Operations Research 40(1) , pp.113-25.

[56] Barnes J, Chambers J., 1995, Solving the job shop scheduling problem with tabu search. IIE Transactions 27(2) , pp. 257-63.

[57] Amico D.M, Trubian M., 1993, Applying tabu search to job shop scheduling problem, .Annals of Operations Research 41, pp. 231-52.

[58] Widmer M., 1991, Job shop scheduling with tool constraints: a tabu search approach. Journal of Operational Research Society 41(1) , pp. 75-82.

[59] Bigel J.E, Davern J.J., 1990, Genetic algorithms and job shop scheduling. Computers and Industrial Engineering 19, pp. 81-91.

[60] Della C.F, Tadei R., Volta G., 1995, A genetics –based for job shop problem. Computers and Operations Research 22, pp. 15-24.

[61] Sakawa, M., Kato K., Mori T., 1996, Flexible scheduling in a machining center through genetic algorithms. Computers and Industrial Engineering 30(4), pp.931-40.

[62] Low C., Yuklink Y., Wu T.H., 2006, Modelling and heuristics of FMS scheduling with multiple objectives, Computers and Operations Research 33, pp. 674-694.

[63] Hwang C.L, Yoon K., 1981, Multiple objective decision making: methods and its applications,Heidelberg: Springer, Berlin, Germany.

[64] Vieira,D. A.G., Adriano,R.L.S.,Vasconcelos, J.A., Krahenbülhl,L., 2004, Treating Constraints as Objectives in Multiobjective Optimization Problems Using Niched Pareto Genetic Algorithm, IEEE Transactions on Magnetics,vol 40,pp 1188-1191.

[65] Sastry,K.,2007, Single and Multiobjective Genetic Algorithm Toolbox for Matlab in C++, Thecnical Report, Illinois Genetic Algorithms Laboratory University of Illinois at Urbana-Champaign.

EKLER

Ek 1: Önerilen kısıtsız çok amaçlı genetik algoritmanın KUR problemi için (n=3) bulduğu Pareto optimal kümedeki 100 çözümlün amaç fonksiyonlarına verdiği değerler.

Çözüm	f1	f2	Çözüm	f1	f2
1	-16,585	-5,344	26	-14,678	-11,298
2	-16,756	-4,626	27	-14,728	-11,164
3	-16,375	-6,401	28	-15,604	-7,833
4	-16,541	-5,588	29	-16,051	-7,520
5	-16,754	-4,635	30	-16,132	-7,230
6	-16,457	-6,028	31	-16,810	-4,369
7	-16,624	-5,210	32	-16,944	-4,055
8	-18,187	-3,207	33	-17,034	-3,896
9	-16,581	-5,430	34	-14,652	-11,333
10	-18,228	-3,019	35	-14,730	-11,124
11	-16,664	-5,027	36	-14,782	-10,990
12	-16,664	-5,029	37	-14,806	-10,938
13	-18,564	-1,429	38	-14,856	-10,763
14	-16,854	-4,254	39	-14,906	-10,574
15	-18,958	-0,208	40	-14,955	-10,376
16	-18,353	-2,411	41	-15,032	-10,068
17	-18,694	-0,930	42	-15,081	-9,862
18	-19,048	-0,049	43	-15,128	-9,657
19	-14,498	-11,598	44	-15,175	-9,454
20	-15,891	-7,751	45	-15,222	-9,256
21	-15,971	-7,653	46	-15,356	-8,704
22	-16,949	-3,999	47	-15,399	-8,537
23	-17,945	-3,875	48	-15,442	-8,381
24	-14,574	-11,500	49	-15,490	-8,203
25	-14,626	-11,403	50	-15,533	-8,032

Ek 1: Önerilen kısıtsız çok amaçlı genetik algoritmanın KUR problemi için (n=3) bulduğu Pareto optimal kümedeki 100 çözümün amaç fonksiyonlarına verdiği değerler (devamı).

Çözüm no	1. Amaç	2.Amaç	Çözüm no	1. Amaç	2.Amaç
51	-15,569	-7,928	76	-15,463	-8,299
52	-15,639	-7,764	77	-15,506	-8,142
53	-16,050	-7,555	78	-15,516	-8,130
54	-16,131	-7,317	79	-15,523	-8,098
55	-16,212	-6,985	80	-15,563	-7,973
56	-16,295	-6,599	81	-16,131	-7,346
57	-16,675	-4,864	82	-16,212	-7,056
58	-16,762	-4,561	83	-16,294	-6,692
59	-16,850	-4,300	84	-16,376	-6,290
60	-16,984	-3,986	85	-16,460	-5,879
61	-18,025	-3,778	86	-16,544	-5,478
62	-14,808	-10,886	87	-16,630	-5,102
63	-14,887	-10,595	88	-16,716	-4,760
64	-14,938	-10,420	89	-16,803	-4,456
65	-14,989	-10,231	90	-16,890	-4,195
66	-15,011	-10,167	91	-14,965	-10,286
67	-15,061	-9,963	92	-15,067	-9,900
68	-15,110	-9,758	93	-15,167	-9,490
69	-15,158	-9,552	94	-15,265	-9,082
70	-15,205	-9,349	95	-15,359	-8,692
71	-15,252	-9,151	96	-15,451	-8,332
72	-15,267	-9,064	97	-15,539	-8,009
73	-15,329	-8,826	98	-16,252	-6,922
74	-15,374	-8,641	99	-16,375	-6,360
75	-15,419	-8,465	100	-16,459	-5,951

Ek 2: Önerilen kısıtlı çok amaçlı genetik algoritmanın çok amaçlı sırt çantası problemi için 100 ana popülasyon genişliği ile elde ettiği Pareto optimal çözümlerin amaç fonksiyonlarına verdiği değerler.

Çözüm no	1. Amaç	2.Amaç	Çözüm no	1. Amaç	2.Amaç
1	16240	20470	26	17773	20042
2	16295	20462	27	17827	20015
3	16361	20454	28	17882	19986
4	16415	20446	29	17893	19978
5	16516	20429	30	17931	19958
6	16625	20409	31	17954	19945
7	16726	20388	32	17991	19927
8	16876	20354	33	18012	19914
9	16882	20353	34	18054	19892
10	16919	20344	35	18103	19859
11	17059	20305	36	18160	19824
12	17073	20302	37	18203	19798
13	17130	20287	38	18255	19765
14	17187	20265	39	18286	19744
15	17230	20253	40	18313	19725
16	17257	20245	41	18348	19700
17	17282	20236	42	18403	19662
18	17321	20224	43	18435	19639
19	17337	20220	44	18488	19600
20	17391	20201	45	18505	19586
21	17539	20146	46	18549	19551
22	17567	20134	47	18571	19534
23	17630	20108	48	18612	19502
24	17662	20095	49	18646	19471
25	17702	20075	50	18675	19445

Ek 2: Önerilen kısıtlı çok amaçlı genetik algoritmanın çok amaçlı sırt çantası problemi için 100 ana popülasyon genişliği ile elde ettiği Pareto optimal çözümlerin amaç fonksiyonlarına verdiği değerler (devamı).

Çözüm no	**1. Amaç**	**2.Amaç**	**Çözüm no**	**1. Amaç**	**2.Amaç**
51	18695	19428	76	19259	18821
52	18727	19402	77	19261	18818
53	18756	19372	78	19297	18776
54	18789	19343	79	19317	18744
55	18816	19316	80	19342	18713
56	18840	19292	81	19379	18668
57	18868	19267	82	19407	18618
58	18871	19264	83	19424	18591
59	18897	19236	84	19457	18542
60	18920	19215	85	19474	18513
61	18940	19191	86	19500	18472
62	18975	19155	87	19528	18425
63	19009	19122	88	19548	18386
64	19026	19099	89	19578	18333
65	19040	19084	90	19605	18286
66	19061	19065	91	19630	18233
67	19085	19033	92	19657	18177
68	19091	19025	93	19678	18134
69	19114	19002	94	19693	18103
70	19132	18978	95	19711	18062
71	19158	18948	96	19734	18011
72	19191	18907	97	19760	17957
73	19203	18891	98	19782	17897
74	19219	18871	99	19804	17845
75	19237	18850	100	19832	17762

EK 3: Kısıtsız algoritma c++ kodları

```
# include <iostream>
# include <string>
# include <stdlib.h>
# include <math.h>
# include <stdio.h>
# include <conio.h>
# include <iomanip.h>
#include <fstream>

# define N              100
#define FALSE  0
#define TRUE   1

using namespace std;
# include "kisitsiz_spec_file.h"
# include "randomnumber.h"

float uniform(){
        float a;
        a=(float) rand()/(float)RAND_MAX;
        return a;
}

int decideP1(){

        int i1=ceil(uniform()*100);
return i1;
}

int decideP2(){

        int i2=ceil(uniform()*100);
return i2;
}

int main (){

        kisitsiz P[N];
        kisitsiz Q[N];
        kisitsiz non_P[N];
        kisitsiz non_Q[N];

        for(int i=1;i<=N;i++){
                P[i].GetParents();
                /*P[i].objective(P[i]);*/
        }


        for (int w=1;w<=50;w++){
kisitsiz parent1,parentbin1;
kisitsiz parent2,parentbin2;
kisitsiz offspbin1,offspbin2;
kisitsiz offspring1,offspring2;

int k=decideP1();
int j=decideP2();

parent1.select(P[j]);
parent2.select (P[k]);
parentbin1.dec_to_bin(parent1);
parentbin2.dec_to_bin(parent2);

offspbin1.crossover1(parentbin1,parentbin2);
offspbin2.crossover1(parentbin2,parentbin1);

offspbin1.mutation(offspbin1);
offspbin2.mutation(offspbin2);

offspring1.bin_to_dec(offspbin1);
offspring2.bin_to_dec(offspbin2);

Q[w].Getoffsprings(offspring1);
```

```
Q[100-w+1].Getoffsprings(offspring2);
        }

P[1].Display();
P[2].Display();
P[3].Display();
P[4].Display();

for(int z=1;z<=N;z++){

        non_P[z].non_dom_P(P[z],P[z]);
}

for (int zz=1;zz<=N;zz++){
        for(int zzz=1;zzz<=N;zzz++){
                non_P[zz].non_dom_P(non_P[zz],P[zzz]);}}



         non_P[1].dominated(P[1],P[4]);
         non_P[1].dominated(P[1],P[5]);
                */
        int Ne=ceil(0.25*N);
        kisitsiz E[N];
        kisitsiz non_E[N];

        for (int k=1;k<=N;k++){
                for (int kk=1;kk<=N;kk++){

                        E[k].dominated(non_P[k],non_Q[kk]);}

        }

getch();
return 0;

}

"kisitsiz_spec_file.cc"
# include <iostream>
# include <string>
# include <stdlib.h>
# include <stdio.h>
# include <math.h>
#include <fstream>
# include <conio.h>

using namespace std;

# include "kisitsiz_spec_file.h"
# include "randomnumber.h"

kisitsiz::kisitsiz(){
        float x1=0.0;
        float x2=0.0;
        float x3=0.0;
        int a[]={0,0,0,0,0,0,0,0,0,0};
        int b[]={0,0,0,0,0,0,0,0,0,0};
        int c[]={0,0,0,0,0,0,0,0,0,0};
        float f1=0.0;
        float f2=0.0;
        int rank=0;

}

kisitsiz::kisitsiz(float pos1,float pos2,float pos3,
                                        int a1[],int b1[],int c1[],
                                        float ff1,float ff2,int rnk){
x1=pos1;
x2=pos2;
x3=pos3;
a[1]=a1[1],a[2]=a1[2],a[3]=a1[3],a[4]=a1[4],a[5]=a1[5],a[6]=a1[6],
a[7]=a1[7],a[8]=a1[8],a[9]=a1[9],a[10]=a1[10],a[11]=a1[11];

b[1]=b1[1],b[2]=b1[2],b[3]=b1[3],b[4]=b1[4],b[5]=b1[5],b[6]=b1[6],
```

```
b[7]=b1[7],b[8]=b1[8],b[9]=b1[9],b[10]=b1[10],b[11]=b1[11];

c[1]=c1[1],c[2]=c1[2],c[3]=c1[3],c[4]=c1[4],c[5]=c1[5],c[6]=c1[6],
c[7]=c1[7],c[8]=c1[8],c[9]=c1[9],c[10]=c1[10],c[11]=c1[11];

f1=ff1;
f2=ff2;
rank=rnk;

}

void kisitsiz::Display(){
        cout <<x1<<"    "<<x2<<"    "<<x3<<"    "<<f1<<"    "<<
                f2<<"   "<<rank<<endl;

        }

float kisitsiz::number(){
float x=-5+ random()*10;
return x;
}

void kisitsiz::GetParents(){
 x1=number();
 x2=number();
 x3=number();
 rank=0;
 f1=(-10)*(exp(-0.2*(pow(x1,2)+pow(x2,2)))+ exp(-0.2*(pow(x2,2)+pow(x3,2))));
 f2=pow(fabs(x1),0.8)+(5*sin(pow(x1,3)))+
                pow(fabs(x2),0.8)+(5*sin(pow(x2,3)))+
                pow(fabs(x3),0.8)+(5*sin(pow(x3,3)));


/* cout<<x1<<"/"<<x2<<"/"<<x3<<"obj===  "<<f1<<"  and   " <<f2<<endl;*/
}

void kisitsiz::Getoffsprings(kisitsiz P){
 x1=P.x1;
 x2=P.x2;
 x3=P.x3;
 rank=0;
 f1=-10*(exp(-0.2*(pow(x1,2)+pow(x2,2)))+
                exp(-0.2*(pow(x2,2)+pow(x3,2))));
 f2=pow(fabs(x1),0.8)+(5*sin(pow(x1,3)))+
                pow(fabs(x2),0.8)+(5*sin(pow(x2,3)))+
                pow(fabs(x3),0.8)+(5*sin(pow(x3,3)));


 cout<<x1<<"/"<<x2<<"/"<<x3<<"    obj===  "
          <<f1<<"  and   " <<f2<<endl;
}


float kisitsiz::random(){
int stime;
int RTAG=0;
long ltime;
float randnum;
if (RTAG==0){
        ltime=time(NULL);
        stime= (unsigned) ltime/2;
        RTAG=1;
}
randnum=rand()%100;
randnum /=100;
return (randnum);
}


void kisitsiz::select(kisitsiz P){

x1=P.x1;
x2=P.x2;
x3=P.x3;
f1=P.f1;
f2=P.f2;
```

```
rank=P.rank;
cout <<"yoo"<<x1<<"*"<<x2<<"*"<<x3<<"*"<<f1<<"*"<<f2<<"*"<<rank<<endl;
}


void kisitsiz::dec_to_bin (kisitsiz P)
{

	int ka=2;
	int kka=5;
	if (P.x1<0){
		a[1]=0;
			P.x1=-P.x1;
		/*	cout<<a<<endl;*/
	}
	else {
	a[1]=1;
	}

int integer_a=floor(P.x1);
float diffa = P.x1-integer_a;
for (int ja=2;ja>=0;ja--){
	if(integer_a>=pow(2,ja)){
		integer_a=integer_a - pow(2,ja);
	a[ka]=1;
	ka=ka+1;
	}
	else {
		a[ka]=0;
		ka=ka+1;
	}
}

	for (int jja=1;jja<=7;jja++){
		if (diffa < pow(2,-jja)){
			a[kka]=0;
			kka=kka+1;
		}
		else {
			a[kka]=1;
			kka=kka+1;
			diffa = diffa - pow(2,-jja);
		}

	}

/*cout <<a[1]<<a[2]<<a[3]<<a[4]<<a[5]<<a[6]<<a[7]
<<a[8]<<a[9]<<a[10]<<a[11]<<endl;*/

/*for b*/

	int kb=2;
	int kkb=5;
	if (P.x2<0){
		b[1]=0;
			P.x2=-P.x2;
	}
		else {
	b[1]=1;
	}

int integer_b=floor(P.x2);
float diffb = P.x2-integer_b;
for (int jb=2;jb>=0;jb--){
	if(integer_b>=pow(2,jb)){
		integer_b=integer_b - pow(2,jb);
	b[kb]=1;
	kb=kb+1;
	}
	else {
		b[kb]=0;
		kb=kb+1;
	}

}
```

```
		for (int jjb=1;jjb<=7;jjb++){
			if (diffb < pow(2,-jjb)){
				b[kkb]=0;
				kkb=kkb+1;
			}
			else {
				b[kkb]=1;
				kkb=kkb+1;
				diffb = diffb - pow(2,-jjb);
			}

		}

/*cout <<b[1]<<b[2]<<b[3]<<b[4]<<b[5]<<b[6]<<b[7]
<<b[8]<<b[9]<<b[10]<<b[11]<<endl;*/


 /*for c */
		int kc=2;
		int kkc=5;
		if (P.x3<0){
			c[1]=0;
				P.x3=-P.x3;
		}
			else {
		c[1]=1;
		}

int integer_c=floor(P.x3);
float diffc = P.x3-integer_c;
for (int jc=2;jc>=0;jc--){
		if(integer_c>=pow(2,jc)){
			integer_c=integer_c - pow(2,jc);
		c[kc]=1;
		kc=kc+1;
		}
		else {
			c[kc]=0;
			kc=kc+1;
		}
}

		for (int jjc=1;jjc<=7;jjc++){
			if (diffc < pow(2,-jjc)){
				c[kkc]=0;
				kkc=kkc+1;
			}
			else {
				c[kkc]=1;
				kkc=kkc+1;
				diffc = diffc - pow(2,-jjc);
			}

		}
/*cout <<c[1]<<c[2]<<c[3]<<c[4]<<c[5]<<c[6]<<c[7]
<<c[8]<<c[9]<<c[10]<<c[11]<<endl;*/


}


void kisitsiz::bin_to_dec (kisitsiz P){

		 x1=(4*P.a[2])+(2*P.a[3])+(1*P.a[4])+(0.5*P.a[5])+
			(0.25*P.a[6])+(0.125*P.a[7])+(0.0625*P.a[8])+
			(0.03125*P.a[9])+(0.015625*P.a[10])+(0.00078125*P.a[11]);
		if (P.a[1]=0) {x1=-1*x1;}


		x2=(4*P.b[2])+(2*P.b[3])+(1*P.b[4])+(0.5*P.b[5])+
			(0.25*P.b[6])+(0.125*P.b[7])+(0.0625*P.b[8])+
			(0.03125*P.b[9])+(0.015625*P.b[10])+(0.00078125*P.b[11]);
		if (P.b[1]=0) {x2=-1*x2;}


		 x3=(4*P.c[2])+(2*P.c[3])+(1*P.c[4])+(0.5*P.c[5])+
```

```
                (0.25*P.c[6])+(0.125*P.c[7])+(0.0625*P.c[8])+
                (0.03125*P.c[9])+(0.015625*P.c[10])+(0.00078125*P.c[11]);
        if (P.c[1]=0) {x3=-1*x3;}


        /*cout<<"****"<<x1<<"/"<<x2<<"/"<<x3<<endl;*/
}
void kisitsiz::crossover1(kisitsiz P1,kisitsiz P2){
        int ca=ceil(1+ random()*10);
        int cb=ceil(1+ random()*10);
        int cc=ceil(1+ random()*10);

        for (int r=1;r<=ca;r++){
                a[r]=P1.a[r];}

        for (int rr=ca;rr<=11;rr++){
                a[rr]=P2.a[rr];}

        /*cout <<"**"<<a[1]<<a[2]<<a[3]<<a[4]<<a[5]<<a[6]<<a[7]
        <<a[8]<<a[9]<<a[10]<<a[11]<<endl;*/

        for (int s=1;s<=cb;s++){
                b[s]=P1.b[s];}

        for (int ss=cb;ss<=11;ss++){
                b[ss]=P2.b[ss];}

/* cout <<"**"<<b[1]<<b[2]<<b[3]<<b[4]<<b[5]<<b[6]<<b[7]
        <<b[8]<<b[9]<<b[10]<<b[11]<<endl;*/


        for (int t=1;t<=cc;t++){
                c[t]=P1.c[t];}

        for (int tt=cc;tt<=11;tt++){
                c[tt]=P2.c[tt];}

/*cout <<"**"<<c[1]<<c[2]<<c[3]<<c[4]<<c[5]<<c[6]<<c[7]
        <<c[8]<<c[9]<<c[10]<<c[11]<<endl;*/

        float i,ii,iii;
        i=random();
        ii=random();
        iii=random();

        if (i<0.5){a[1]=0;
                                                                }
        else {a[1]=1;
                                                        }

        if (ii<0.5){b[1]=0;
                                                                }
        else {b[1]=1;
                                                        }

        if (iii<0.5){c[1]=0;
                                                                }
        else {c[1]=1;
                                                        }

}

void kisitsiz::mutation(kisitsiz P){
        float mpa=random();
        float mpb=random();
        float mpc=random();

        if(mpa<=0.2) {
                int change_a=ceil(1+ random()*10);
                if (P.a[change_a]=0) {P.a[change_a]=1;}
                else {P.a[change_a]=0;}

        }

        if(mpb<=0.2) {
                int change_b=ceil(1+ random()*10);
```

```
                    if (P.b[change_b]=0) {P.b[change_b]=1;}
                    else {P.b[change_b]=0;}
            }
            if(mpc<=0.2) {
                    int change_c=ceil(1+ random()*10);
                    if (P.c[change_c]=0) {P.c[change_c]=1;}
                    else {P.a[change_c]=0;}
            }
/*cout <<"***"<<P.a[1]<<P.a[2]<<P.a[3]<<P.a[4]<<P.a[5]<<P.a[6]<<P.a[7]
            <<P.a[8]<<P.a[9]<<P.a[10]<<P.a[11]<<endl;

cout <<"***"<<P.b[1]<<P.b[2]<<P.b[3]<<P.b[4]<<P.b[5]<<P.b[6]<<P.b[7]
            <<P.b[8]<<P.b[9]<<P.b[10]<<P.b[11]<<endl;

cout <<"***"<<P.c[1]<<P.c[2]<<P.c[3]<<P.c[4]<<P.c[5]<<P.c[6]<<P.c[7]
            <<P.c[8]<<P.c[9]<<P.c[10]<<P.c[11]<<endl;*/

}

void kisitsiz::non_dom_P (kisitsiz P1,kisitsiz P2){
            if((P1.f1<P2.f1) && (P1.f2<P2.f2)){
                    f1=P1.f1;
                    f2=P1.f2;
                    rank=(P1.rank)+1;}


            if ((P1.f1<P2.f1) && (P1.f2>P2.f2) ||(P1.f1>P2.f1) && (P1.f2<P2.f2)||
             (P1.f1==P2.f1) && (P1.f2<P2.f2)||(P1.f1<P2.f1) && (P1.f2==P2.f2)||(P1.f1==P2.f1) && (P1.f2==P2.f2)){
                    f1=P1.f1;
                    f2=P1.f2;
                    rank=P1.rank;}

            if((P1.f1>P2.f1) && (P1.f2>P2.f2)){
                    f1=1000;
                    f2=1000;
            rank=P1.rank;
            }
                    cout<<f1<<"  &&&  "<<f2<<"  &&&  "<<rank<<endl;


            }


double  ClusterDistance(int  cluster1, int  cluster2)
{
  double  sum = 0;
  int  numOfPairs = 0;
  int  c1 = clusters[cluster1];
  while (c1 >= 0) {
    int  c2 = clusters[cluster2];

      c2 = clusterList[c2];
    }
    c1 = clusterList[c1];
  }
  return (sum / double(numOfPairs));
} /* ClusterDistance */


void  JoinClusters(int  cluster1, int  cluster2, int&  numOfClusters)
{
  int  c1 = clusters[cluster1];
  while (clusterList[c1] >= 0)
    c1 = clusterList[c1];
  clusterList[c1] = clusters[cluster2];
  numOfClusters--;
  clusters[cluster2] = clusters[numOfClusters];
} /* JoinClusters */


int  ClusterCentroid(int  cluster)
{
  double  minSum = -1;
  int  minIndex;
  int  c1 = clusters[cluster];
  while (c1 >= 0) {
```

```
    int  c2 = clusters[cluster];
    double  sum = 0;

      c2 = clusterList[c2];
    }
    if (sum < minSum || minSum < 0) {
      minSum = sum;
      minIndex = c1;
    }
    c1 = clusterList[c1];
  }
  return minIndex;
```

"kisitsiz_spec_file.h" Dosyası

```
# ifndef kisitsiz_h
# define kisitsiz_h
class kisitsiz{

private:
            float x1,x2,x3;
            int a[11],b[11],c[11];
            float f1,f2;
            int rank;
public:
            kisitsiz ();
kisitsiz (float,float,float,int [],int[],int[],float,float,int);

void Display();
float number();
float random();
void GetParents();

void dec_to_bin (kisitsiz);

void select(kisitsiz P);
void crossover1(kisitsiz P1,kisitsiz P2);
void mutation(kisitsiz P);
void bin_to_dec (kisitsiz P);

void Getoffsprings(kisitsiz P);

void non_dom_P (kisitsiz P1,kisitsiz P2);
double  ClusterDistance(int  cluster1, int  cluster2)
void  JoinClusters(int  cluster1, int  cluster2, int& )
int  ClusterCentroid(int  cluster)

};

# endif
```

Ek 4: Kısıtlı algoritma kodları

```
# include <string>
# include <stdlib.h>
# include <math.h>
# include <stdio.h>
# include <conio.h>
# include <iomanip>
#include <fstream.h>
# include <time.h>
FILE *probfile;
FILE *outfile;

void read_data();
using namespace std;
# include "kisitli_spec_file.h"
# include "randomnumber.h"
# define N              400
# define M              25

float uniform(){
              float a;
              a=(float) rand()/(float)RAND_MAX;
              return a;
}


int main (){
void read_data()
{
int i,j;

   probfile = fopen("data.dat", "r");
   for (i=1; i<=500; i++)
   {
     for (j=1; j<i; j++)
     {
       fscanf(probfile, " %f", &profit[i][j]);
       profit [j][i] = profit [i][j];
     }

     for (j=i; j<=500; j++)
     {
       fscanf(probfile, " %f", &capacity[i][j]);
       capacity [j][i] = capacity [i][j];
     }
   }
   fclose (probfile);
}    /* end of read_data  */

kisitli PQC[N];



kisitli parent1,parent2;
/*kisitli offspring1,offspring2;*/

              for (int ss=1;ss<=100;ss++){
              PQC[ss].initial();
              PQC[ss].profit_capacity ();}
/*offspring1.initial();
offspring2.initial();*/

              for (int w=101;w<=200;w++){
randomnumber rnd;
int pp1=rnd.random(100)+1;
int pp2=rnd.random(100)+1;
parent1.select(PQC[pp1]);
parent2.select(PQC[pp2]);

/*parent1.DisplaySolutionArray();
parent2.DisplaySolutionArray();*/
```

```
/*randomnumber rnd1,rnd2,rnd3,rnd4,rnd5;*/

int p1=ceil(uniform()*499);
int p2=ceil(uniform()*499);
int p3=ceil(uniform()*499);
int p4=ceil(uniform()*499);
int p5=ceil(uniform()*499);

PQC[w].crossover(parent1,parent2,p1,p2,p3,p4,p5);

PQC[w].profit_capacity ();}


        for (int cc=1;cc<=100;cc++){
                for(int ccc=1;ccc<=100;ccc++){
                        PQC[cc].compare (PQC[cc],PQC[ccc]);}}


        for (int rr=101;rr<=200;rr++){
                for(int rrr=101;rrr<=200;rrr++){
                        PQC[rr].compare (PQC[rr],PQC[rrr]);}}


for (int a=1;a<=100;a++){
                for(int aa=101;aa<=200;aa++){
                        PQC[a].compareelit (PQC[a],PQC[aa]);}}


for (int y=101;y<=200;y++){
                for(int rrr=1;rrr<=100;rrr++){
                        PQC[y].compareelit (PQC[y],PQC[rrr]);}}


kisitli E[M];

for (int t1=1;t1<=199;t1++){
        for (int t2=t1+1;t2<=200;t2++){
        PQC[t1].true_elit(PQC[t1],PQC[t2]);
}
        E[t1].DisplaySolutionArray();
}

getch();

return 0;


# include <string>
# include <stdlib.h>
# include <stdio.h>
# include <math.h>
#include <fstream.h>
# include <conio.h>


using namespace std;

# include "kisitli_spec_file.h"
# include "randomnumber.h"

kisitli::kisitli(){

        for (int r=0;r<=499;r++){
         solution[r]=0;}
kn1_profit=0.0;
kn2_profit=0.0;
kn1_capacity=0.0;
kn2_capacity=0.0;
rank=0;
d_rank=0;
e_rank=0;
ed_rank=0;
elit=0;
}

kisitli::kisitli(int kn11_profit,int kn22_profit,
```

```
int kn11_capacity,int kn22_capacity, int sol [],int rnk, int dd_rank, int ee_rank, int edd_rank,int elitt){
kn1_profit=kn11_profit;
kn2_profit=kn22_profit;
kn1_capacity=kn11_capacity;
kn2_capacity=kn22_capacity;
rank=rnk;
d_rank=dd_rank;
e_rank=ee_rank;
ed_rank=edd_rank;
elit=elitt;
for (int f=0;f<=499;f++){
solution [f]=sol [f];
}
        }


float kisitli::random(){
int stime;
int RTAG=0;
long ltime;
float randnum;
if (RTAG==0){
        ltime=time(NULL);
        stime= (unsigned) ltime/2;
        RTAG=1;
}
randnum=rand()%100;
randnum /=100;
return (randnum);
}


void kisitli::initial (){

for (int k=0;k<=499;k++){
        float pos=random();
        if (pos <=0.5){
                solution [k]=0;}
        else {solution [k]=1;}
                                }

}


void kisitli::DisplaySolutionArray(){
        for (int j=0;j<=505;j++){
                cout <<solution [j]<<endl;
        }
cout<<"kn1profit=  "<<kn1_profit<<"   kn2profit=  "<<kn2_profit<<
"   kn1cap= "<<kn1_capacity<<"    kn2cap=  "<<kn2_capacity<<
"   rank= "<<rank<<"   drank= "<<d_rank<<"   erank= "<<e_rank<<"   edrank= "<<ed_rank<<"   elit= "<<elit<< endl;
}



void kisitli::select(kisitli P){
for (int rr=0;rr<=499;rr++){
                solution[rr]=P.solution[rr];}

kn1_profit=P.kn1_profit;
kn2_profit=P.kn2_profit;
kn1_capacity=P.kn1_capacity;
kn2_capacity=P.kn2_capacity;
rank=P.rank;
d_rank=P.d_rank;
e_rank=P.e_rank;
ed_rank=P.ed_rank;
elit=P.elit;
cout <<"selected"<<endl;
        }


void kisitli::mutation (){
        for (int mu=0;mu<=499;mu++){
        randomnumber fnum;
        float pmu=fnum.frandom();
```

```
		if(pmu<=0.10) {
			if (solution[mu]=0) {solution[mu]=1;}
		 else if (solution[mu]=1){solution[mu]=0; }
		}}
		cout<<"mutation"<<endl;
			}


void kisitli::crossover(kisitli P1,kisitli P2,int p1,int p2,int p3,int p4,int p5){

		if (p1>p2) {int temp =p1;
		p1=p2;
		p2=temp;}
		if (p1>p3) {int temp =p1;
		p1=p3;
		p3=temp;}
		if (p1>p4) {int temp =p1;
		p1=p4;
		p4=temp;}
		if (p1>p4) {int temp =p1;
		p1=p4;
		p4=temp;}
		if (p1>p5) {int temp =p1;
		p1=p5;
		p5=temp;}

		if (p2>p3) {int temp =p2;
		p2=p3;
		p3=temp;}
		if (p2>p4) {int temp =p2;
		p2=p4;
		p4=temp;}
		if (p2>p5) {int temp =p2;
		p2=p5;
		p5=temp;}
		if (p3>p4) {int temp =p3;
		p3=p4;
		p4=temp;}
		if (p3>p5) {int temp =p3;
		p3=p5;
		p5=temp;}
		if (p4>p5) {int temp =p4;
		p4=p5;
		p5=temp;}
/*cout <<p1<<"*"<<p2<<"*"<<p3<<"*"<<p4<<"*"<<p5<<endl;*/


for (int j1=0;j1<=p1;j1++){
			solution[j1]=P1.solution[j1];}

for (int j2=p1+1;j2<=p2;j2++){
			solution[j2]=P2.solution[j2];}

for (int j3=p2+1;j3<=p3;j3++){
			solution[j3]=P1.solution[j3];}

for (int j4=p3+1;j4<=p4;j4++){
			solution[j4]=P2.solution[j4];}

for (int j5=p4+1;j5<=p5;j5++){
			solution[j5]=P1.solution[j5];}

for (int j6=p5+1;j6<=499;j6++){
			solution[j6]=P2.solution[j6];}

/*for (int y=1;y<=504;y++){
		cout<<solution[y]<<endl;}*/

}

void kisitli::compare (kisitli P1,kisitli P2){

		if (P1.kn1_profit>P2.kn1_profit && P1.kn2_profit>P2.kn2_profit){
		P1.rank++;}
```

```
        if (P1.kn1_profit<P2.kn1_profit && P1.kn2_profit<P2.kn2_profit){
        P1.d_rank++;}

        cout<<rank<<d_rank<<endl;}


void kisitli::compareelit(kisitli P1,kisitli P2){

        if (P1.kn1_profit>P2.kn1_profit && P1.kn2_profit>P2.kn2_profit){
                P1.e_rank++;}
        if (P1.kn1_profit<P2.kn1_profit && P1.kn2_profit<P2.kn2_profit){
        P1.ed_rank++;}
        /*kn1_profit=P1.kn1_profit;
        kn2_profit=P1.kn2_profit;
        for (int r=0;r<=499;r++){
         solution[r]=P1.solution[r];}*/
        cout<<e_rank<<endl;
}


void kisitli::true_elit(kisitli P1,kisitli P2){
        if ((P1.d_rank=0) && (P1.ed_rank=0) && (P1.rank + P1.e_rank > P2.rank+P2.e_rank)){
                        P1.elit++;}

}


# ifndef kisitli_h
# define kisitli_h
class kisitli{

private:

int solution[500];
int kn1_profit;
int kn2_profit;
int kn1_capacity;
int kn2_capacity;
int rank;
int d_rank;
int e_rank;
int ed_rank;

public:
        kisitli ();
        kisitli(int ,int,int,int, int [],int,int,int, int);

float random();

void initial ();
void DisplaySolutionArray ();
void profit_capacity();
void select(kisitli P);
void crossover(kisitli P1,kisitli P2,int ,int ,int ,int ,int);
void mutation();
void compare (kisitli P1,kisitli P2);
void compareelit (kisitli P1,kisitli P2);
bool true_elit(kisitli P1);

~kisitli ();
};

# endif
```

ÖZGEÇMİŞ

Adı Soyadı : Fersin Keskin

Doğum Yeri : Kayseri

Doğum Yılı : 04.04.1972

Medeni Hali : Evli

Eğitim ve Akademik Durumu

Lise **:**1986-1989, Yenimahalle Halide Edip Lisesi, Ankara.

Lisans **:**1989-1993**,** Hacettepe Üniversitesi, Fen Fakültesi, İstatistik Bölümü, Ankara.

Yüksek Lisans **:**1996-1998, Michigan State University, Statistics and Probability Department, Operations Research Program, Usa.

Yüksek Lisans **:**1998-2001, University of Pittsburgh, Industrial Engineering,Usa.

Yabancı Dil **:** İngilizce

İş Tecrübesi:

2001-2002 Araştırma Görevlisi, Osmangazi Üniversitesi, Fen Fakültesi, İstatistik Bölümü, Eskişehir.

2002-2009 Araştırma Görevlisi, Hacettepe Üniversitesi, Fen Bilimleri Enstitüsü, İstatistik Bölümü, Ankara.