KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ÇOK AMAÇLI NSGA-II VE MOPSO OPTİMİZASYON ALGORİTMALARI İLE KABLOSUZ ALGILAYICI AĞLARINDA OPTİMUM KÜME BAŞI YERİ SEÇİMİ VE KÜMELEMESİ

YÜKSEK LİSANS TEZİ

Vahid FARYAD AGHJEH KAND

OCAK 2014
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ÇOK AMAÇLI NSGA-II VE MOPSO OPTİMİZASYON ALGORİTMALARI İLE KABLOSUZ ALGILAYICI AĞLARINDA OPTİMUM KÜME BAŞI YERİ SEÇİMİ VE KÜMELEMESİ

Vahid FARYAD AGHJEH KAND

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"YÜKSEK LİSANS (ELEKTRONİK MÜHENDİSLİĞİ)"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 07.01.2014
Tezin Savunma Tarihi : 02.01.2014

Tez Danışmanı : Yrd. Doç. Dr.Gökce HACIOĞLU

Trabzon 2014

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalında

Vahid FARYAD AGHJEH KAND tarafından hazırlanan

ÇOK AMAÇLI NSGA-II VE MOPSO OPTİMİZASYON ALGORİTMALARI İLE KABLOSUZ ALGILAYICI AĞLARINDA OPTİMUM KÜME BAŞI YERİ SEÇİMİ VE KÜMELEMESİ

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 17/12/2013 gün ve 1534 sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda
YÜKSEK LİSANS TEZİ
olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Doç.Dr.Ali GANGAL ……..…….

Üye : Yrd. Doç. Dr.Gökce HACIOĞLU ……..…….

Üye : Yrd. Doç. Dr.Tuğrul ÇAVDAR ……..…….

Prof. Dr. Sadettin KORKMAZ

Enstitü Müdür

ÖNSÖZ

Kablosuz algılayıcı ağların yerlestirildiği ortamlarda herhangi bir altyapı bulunmamaktadır ve bir çok durumda ağda bulunan düğümlerin bakım, onarım ve enerjilerini yenileme imkanları bulunmamaktadır. Bu yüzden sahip olunan optimum küme başı yeri seçimi ve kümelemesinin yanısıra enerjiyi verimli bir şekilde kullanmak kablosuz algılayıcı ağların işlemesinde önemli bir yer tutar. Böylesine güncel ve önemli bir konuyu seçmemde bana destek olan ve tüm çalışma boyunca desteklerini esirgemeyen saygıdeğer hocam Yrd. Doç. Dr. Gökce HACIOĞLU'na çok teşekkür eder, şükranlarımı sunarım. Bu çalışmada yardımı geçen herkese çok teşekkür ederim. Ayrıca hayatım boyunca her an yanımda olan sevgili anneme ve saygıdeğer babama teşekkür ederim.

Vahid FARYAD AGHJEH KAND
Trabzon, 2014

TEZ BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum “Çok Amaçlı NSGA-II ve MOPSO Optimizasyon Algoritmaları ile Kablosuz Algılayıcı Ağlarında Optimum Küme Başı Yeri Seçimi Ve Kümelemesi” başlıklı bu çalışmayı baştan sona kadar danışmanım Yrd. Doç. Dr.Gökce HACIOĞLU’nun sorumluluğunda tamamladığımı, verileri kendim topladığımı, analizleri ilgili laboratuarlarda yaptığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 7/1/2014

Vahid FARYAD AGHJEH KAND

İÇİNDEKİLER

Sayfa No

Yüksek Lisans Tezi

ÖZET

ÇOK AMAÇLI NSGA-II VE MOPSO OPTİMİZASYON ALGORİTMALARI İLE KABLOSUZ ALGILAYICI AĞLARINDA OPTİMUM KÜME BAŞI YERİ SEÇİMİ VE KÜMELEMESİ

Vahid FARYAD AGHJEH KAND

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı
Danışman: Yrd. Dr. Gökce HACIOĞLU
2014, 159 Sayfa, 3 Ek Sayfa

Bu tezde sunucu konumlandırma probleminin özel bir durumu kablosuz algılayıcı ağlarının tasarımı için kullanılmıştır. Bu problem np-zor kombinatoryel problemlerin kategorisinde yer almaktadır. Problemin çözümü ile hem kümeleme hem de optimum küme başı konumunun belirlenmesi gerçekleştirilmiştir. Tanımlanan problem ilk başta tek amaçlı dekompozisyon ağırlıklı toplam genetik algoritma yaklaşımı ile incelenmiştir. Daha sonra çok amaçlı yaklaşımlarıyla çözülmüştür. Bu konu zaten bir çok amaçlı problemdir. Elde edilen sonuçlardan görülmüştür ki dekompozisyon yöntemine dayalı çok amaçlı evrimsel algoritması, doğrusal yaklaşımından dolayı *pareto* cephesinin tüm noktalarını keşfedememektedir. Ayrıca pareto cephesindeki her cevabı bulmak için algoritma baştan koşturulmalıdır. Bu sayılan sorunlar; mağlup olmayan sıralama genetik algoritma ve çok amaçlı parçacık sürü optimizasyonu yöntemlerinin kullanımı ile giderilmiştir. Bu tarz algoritmalar yüksek doğrulukla *pareto* cephesinin tümünü tek sefer koşturulmasıyla bulabilmektedir. Ancak *NSGA-II* algoritmasındaki popülasyon sayısının artması işlem karmaşıklığı ve işlem yükünü oldukca çoğaltmaktadır. Bu durum kablosuz algılayıcı ağlarının kısıtlı işlemcilerinde kullanılmak için uygun değildir. Bu sebeplerle hız artışı, işlem yükü ve karmaşıklığın azaltılması doğrultusunda *MOPSO* algoritma kullanılmıştır. *MOPSO* algoritma sürekli bir algoritmadır. Diğer taraftan problem ikili türdendir. Dolayısıyla *MOPSO* algoritma *PSO* Çekirdeğini sigmoid fonksiyonu aracılığıyla bir ikili algoritmaya dönüştürülerek problem için istenilen çözüme ulaşılmıştır.

Anahtar Kelimeler: Çok amaçlı optimizasyon, kablosuz algılayıcı ağları, sürü zekası, genetik algoritma, hub yeri problemi, pareto cephesi, kümeleme, leach protokolü

Master Thesis

SUMMARY

SELECTING THE OPTIMUM LOCATION OF THE CLUSTER HEAD IN THE WIRELESS SENSOR NETWORKS AND CLUSTERING VIA NSGA-II AND MOPSO ALGORITHMS

Vahid FARYAD AGHJEH KAND

Karadeniz Technical University
The Graduate School of Science
Electronic Engineering Graduate Program
Supervisor: Assistant Prof. Dr. Gökce HACIOĞLU
2014, 159 Pages, 3 Pages Appendix

In this study one special case of the Hub Location Problem known as Wireless Sensor Network Design is used for problem definition and solving, which lies in the field of NP-hard combinatorial problems. This concept has been studied for optimal selection of cluster head location in wireless sensor networks and also clustering. The defined problem is a multi-objective and binary inherently. The optimal solutions for location of head nodes and the members of clusters are found by genetic algorithm via converting the problem to single objective by weighted-sum decomposition. However MOEA/D methods are not able to explore all parts of the Pareto Front due to their linear approach. Moreover the programs which are used MOEA/D algorithms can explore one pareto-front point at each run hence the programs must re-run several times. In order to solve these defects multi-objective optimization algorithms should be used to solve inherently multi-objective problems. Therefore we used NSGA-II and MOPSO algorithms. It can be seen that these algorithms can explore all the points of pareto-front accurately with a single run. However the speed of NSGA-II algorithm falls quickly by growing initial population which lead to the computational load. This is not acceptable for the limited processors of the wireless sensor networks. The MOPSO algorithm is used to handle long convergence duration of pareto-front and increased computational load. MOPSO algorithm is continuous in nature, but the problem is binary in nature. So our problem has reached to desired solution by converting MOPSO to binary algorithm via Sigmoid Function.

Key Words: Multi-objective optimization, wireless sensor networks, swarm intelligence, genetic algorithms, hub location problem, pareto front, clustering, leach protocol.

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

WSN **:** Wireless Sensor Networks

NP-Hard **:** Non-deterministic Polynomial-time

MOEA/D **:** Multiobjective Evolutionary Algorithm Based on Decomposition

NSGA-II **:** Non-dominated sorting genetic agorithm

MOPSO **:** Multi-objective particle swarm optimization

PSO **:** Particle swarm optimization

NEST **:** Network Embedded Systems Technology

CENS **:** Center for Embedded Networked Sensing

GPS **:** Global Positioning System

LEACH **:** Low energy adaptive clustering hierarchy

TDMA **:** Time division multiple access

CDMA **:** Code division multiple access

MAC **:** Medium-access control

MATLAB **:** Matrix laboratory

RWS **:** Roulette Wheel Selection

MOEA/D **:** Multiobjective Evolutionary Algorithm Based on Decomposition

CD **:** Crowding Distance

NPGA **:** Niched Pareto Genetic Algorithm

MVGA **:** Variable-length Genetic Algorithm

PESA-II **:** Pareto Envelope-based Selection Algorithm

SPEA-II **:** Strength Pareto Evolutionary Algorithm

$\mathbf{d_i}$ **:** Demands of nodes

$\|\boldsymbol{x}\|_{\boldsymbol{P}}$ **:** P-Norm of x

NS **:** Non-Dominated Sorting Algoritması

$\boldsymbol{D_i^{min}}$ **:** Minimum distances from cluster heads

C **:** Sep up Energy of nodes

f **:** Each solution of problem or chromosome of genetic algorithms

z **:** Objective function

D **:** All distances among nodes and cluster heads

W **:** Weighted coefficient

1. GENEL BİLGİLER

1.1. Giriş

Sunucu konumunun tahsisi (*Hub location allocation*) [1,2] konusunda sunucu yeri seçiminin problemi Bilgisayar, Elektrik-Elektronik, Telekomünikasyon [3] ve Endüstri mühendisliği alanlarında temel konu sayılmakta ve farklı isimlerle tanınıp çözülmeye çalışılmaktadır. Sunucu düğüm kuruluş yeri seçimindeki amaç, kablolu yada kablosuz bir ağda hizmet veren çeşitli sunucuları maliyet ve hizmet kalitesini gözeterek en uygun sayıda kullanıp en uygun konumlara yerleştirmektir. Örneğin; gezgin iletişim ağlarında bts'ler [4] belli özel yerlerde kurulmalıdır. Eğer bu gerçekleştirilmezse İstanbul gibi büyük şehirlerde cep telefonu ile konuşurken hattın kesilmesi gibi sorunlarla karşılaşılabilmektedir. Bu durumun sebebi kullanıcı sayısının fazlalığı ve bts kapasite yetersizliği olabilir. Buradaki esas sorun bts kuruluş yerleri nereleri olmalı ki ağ kapsama artışı sağlansın ve maliyeti gözeterek kaç adet bts yerleştirilsindir. Benzer probleme petrokimya tesislerindede rastlanır. Bu tesislerde çeşitli noktalardaki basınç ve sıcaklığın ölçülmesi gerekir ve bu ölçümler denetleme odasında incelenmelidir. Eğer bir yerde sıcaklık normal sınırı aşarsa patlamaya sebep olabilir ve bunun sonucunda can ve mal kaybı yaşanabilir. Burada da algılayıcı kuruluş yeri seçim problemi söz konusudur ve bu problem kablosuz Algılayıcı ağlarının tasarımı (*wireless sensor network design*) [7] olarak adlandırılır. Endüstri Mühendisliği dalında ise suncu düğüm yeri problemi *hub location allocation* ismi ile tanınır. *HUB*, sunuş yapan bir servis birimidir ve buradaki *HUB* [2] ile kablosuz Algılayıcı ağlarındaki Küme başı'nın (*Cluster head*) [6] işlevi birbirine oldukça yakındır. Olayı daha iyi kavramak için çeşitli servis büroları olan bir şirketi göz önüne alalım. Bu şirket [8,9] bir takım büroları kapatmak istiyor veya tam tersi yenilerini kurmak için bir takım potansiyelli yerleri bulmuş ve büro kurmak için yer seçimi yapmak istiyor olsun. Gelen müşteri müracaatı, müşteri adresleri yada posta merkezleri gibi raporlara göre bir takım yerler büro kurmak için potansiyelli olarak belirlenebilir buradaki problemin adı Sunucu Konumunun Tahsisi [5] (*Hub location allocation*)'dır.

1.2. Kablosuz Algılayıcı Ağları

WSN (*Wireless Algılayıcı Networks*), "Kablosuz Algılayıcı Ağları"[15] veya "Telsiz Duyarga Ağları" olarak dilimizde kullanılmaktadır. Belirli alanlardaki sıcaklık, basınç, nem vs. gibi değerleri ölçüp bu verileri bir merkeze iletmekle görevli otonom algılayıcıların oluşturduğu kablosuz ağ sistemleridirler. Gelişmiş algılayıcı ağları çift yönlü iletişim yeteneğine de sahiptirler. Böylece, algılayıcı cihazlarının merkezden kontrolü mümkün olmaktadır. Kablosuz algılayıcı ağları; çift taraflı iletişim özelliği, düşük güç tüketimi, yüksek iletişim menzili, esneklik, uygun maliyet, ölçeklenebilirlik, özellikleri sayesinde pek çok sektörde kablolu ölçüm ve kontrol sistemlerinin yerini almaya başlamıştır. Ayrıca aynı ağ üzerine, gerçek zamanlı konumlandırma sistemleri, kısa mesaj haberleşmesi, acil durum ikaz ve yönlendirme eklentileri yapılarak tek sistem ile birden fazla ihtiyaç karşılanabilmektedir.

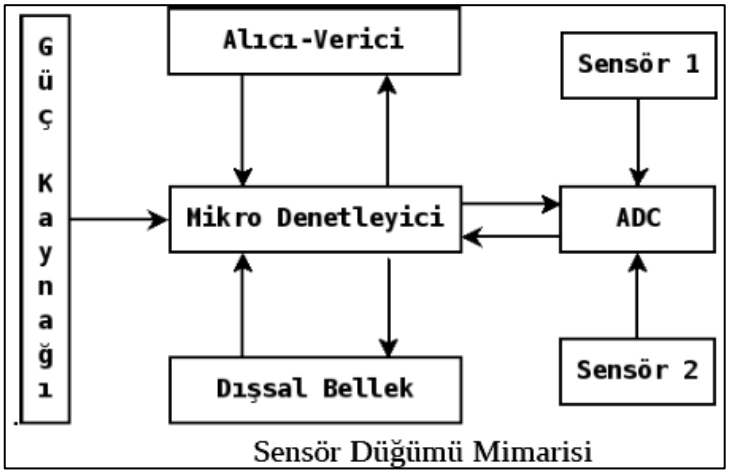
1.3. Algılayıcı Düğümü

Şekil 1'de bir algılayıcı ya da sensör düğümü gösterilmiştir. Algılayıcı düğümlerinin geliştirilmesinin başlangıcı 1998 yılındaki Smartdust projesine dayanır [11]. Bu projenin amaçlarından biri kübik milimetre içerisinde otonom algılama ve iletişim yaratmaktır. Bu proje erken bitmesine rağmen, bir kaç araştırma projesinin doğmasına neden olmuştur. Bu projeler *Berkeley NEST1* ve *CENS2* [11] projeleridir. Bu projelerde yer alan araştırmacılar algılayıcı düğümü için *mote* terimini kullanmaktadır.

Şekil 1. Algılayıcı düğümü

1.4. Algılayıcı Bileşenleri

Algılayıcı düğümünün ana bileşenleri; mikrodenetleyici, alıcı-verici, dışsal bellek, güç kaynağı ve bir veya daha fazla algılayıcıdır. Algılayıcı düğümü mimarisi Şekil 2'de şöyle gösterilmektedir.



Şekil 2. Algılayıcı düğümün bileşenleri

Mikrodenetleyici veriyi işler ve algılayıcı düğüm içerisindeki diğer bileşenlerin işlevselliğini denetler. Alıcı-verici algılayıcı düğümleri *ISM* bandını kullanırlar. Bu band sayesinde geniş dalga kuşağında ve global elverişlilikte özgür radyo yayını sağlanmış olur. Dışsal belleğe enerji bakış açısından yaklaşıldığında en uygun bellek çeşitleri mikro denetleyici çipi üzerindeki bellek ve *Flash* [11] belleklerdir. Çip dışı *RAM*'ler seyrek veya hiç kullanılmamaktadır. *Flash* bellekler maliyeti ve depolama kapasitesi nedeniyle kullanılmaktadır. Bellek gereksinimleri yüksek oranda uygulama bağımlıdır. Depolamanın türüne göre iki farklı bellek kategorisinden bahsedilebilir: a) Uygulamayla ilgili veya kişisel bilgileri saklamak için kullanılan kullanıcı belleği, b) Aygıtın programlanması için kullanılan program belleği, bu bellek ayrıca eğer varsa aygıtın tanımlayıcı verisini de içerebilir. Algılayıcı düğümündeki enerji tüketimi; algılama, iletişim ve veri işleme nedeniyle olmaktadır. Algılayıcı düğümünde veri iletişimi için daha fazla enerji gerekmektedir [11]. Algılama ve veri işleme için enerji tüketimi daha azdır. 1 Kb veriyi 100 metrelik bir uzaklığa iletmek için gereken enerji, yaklaşık olarak saniyede 100 milyon komut işleyen bir işlemcide 3 milyon komut işlemek için gereken enerjiye eşittir. Enerji pil veya kapasitörler içerisinde saklanmaktadır. Durumlardaki değişimlere ölçülebilir tepkiler

üretebilen donanım aygıtlarıdır. Algılayıcılar tarafından algılanan sürekli analog sinyaller "*Analog-to-Digital*" [11] çeviriciler yardımıyla sayısallaştırılarak denetleyicilere daha fazla işlem için gönderilir. Algılayıcı düğümleri küçük boyutlarda, düşük enerji tüketimli, yüksek hacimsel yoğunluklarda çalışabilen, otonom ve gözetimsiz çalışan, ortama uyum sağlayabilme özelliklere sahip olmalıdır. Kablosuz algılayıcı düğümleri sadece sınırlı güç kaynağına sahip (0.5 Ah ve 1.2 V gibi) mikro elektronik algılayıcı aygıtlarını kullanabilir. Algılayıcılar üç kategori şeklinde sınıflandırılmaktadır.

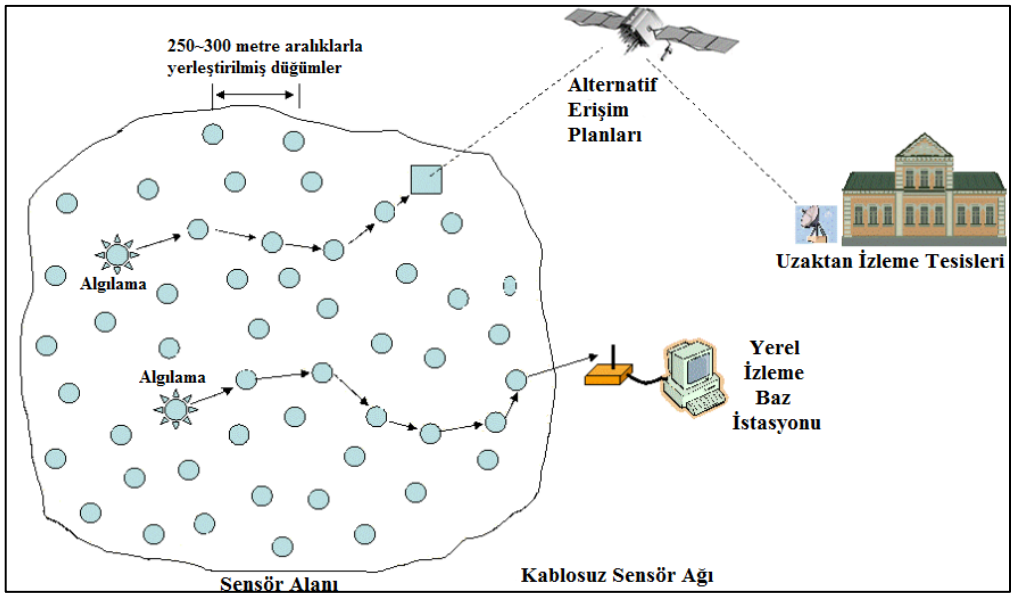
Pasif, her yöne açık (yönsüz) algılayıcılar: Pasif algılayıcılar ortamı aktif araştırma ile değiştirmeden verileri toplayan algılayıcılardır. Kendi enerjilerine sahiptir, enerji analog sinyali yükseltmek için gereklidir. Bu ölçümlerde "yön" [11] şeklinde bir kavram yoktur. Pasif, dar ışınlı algılayıcılar: Bu algılayıcılar pasiftir ancak iyi tanımlanmış ölçüm yönü kavramına sahiptir. Tipik bir örnek olarak kamera verilebilir.

Aktif algılayıcılar, Bu gruptaki algılayıcılar ortamı aktif olarak araştırırlar, örnek olarak sonar veya radar algılayıcıları veya küçük patlamalarla şok dalgaları üreterek çalışan bazı sismik algılayıcı tipleri verilebilir.

Kablosuz Algılayıcı ağlarındaki kapsayıcı teorik çalışmalar Pasif, yönsüz algılayıcıları kastetmektedir. Her algılayıcı düğümü belirli bir kapsama alanına sahiptir. Bu kapsama alanındaki gözlemlerini güvenilir ve doğru bir şekilde raporlayabilir. Kapsama alanını arttırmaya ve algılayıcıların dizilimini iyileştirmeye yönelik çalışmalar yapılmaktadır.

1.5. Kablosuz Algılayıcının Ağ Mimarisi

Kablosuz algılayıcı ağındaki temel elemanları algılama, veri işleme ve haberleşme özelliğine sahip algılayıcı düğümlerdir. Bilindiği gibi algılayıcı düğümleri, herhangi bir kablo olmaksızın, izleyecekleri ortama rastgele saçılmış halde bulunurlar. Aşağıdaki Şekil 3'de bir algılayıcı ağı mimarisi gösterilmektedir [12]. İzlemenin yapıldığı ortamda toplanan veri genelde 3 seviyede işlenilir.



Şekil 3. Kablosuz algılayıcının ağ mimarisi

1. İzlenilecek ortamdaki olaylar, algılayıcı düğümler tarafından algılanır.
2. Her bir algılayıcı düğümü elde ettiği veriyi ayrı ayrı işlemektedir. İkinci seviyede her düğüm algılayıp, işledikleri veriyi komşularına yollamaktadır.
3. Algılayıcı ağ haberleşmesindeki en üst katman, işlenmiş verinin baz (*base*) olarak adlandırılan merkeze yollanılmasıdır.

Baza gönderilen veri eğer başka kıstaslar eşliğinde tekrar analiz edilecekse yada başka amaçlar için kullanılacaksa bazdan bu işlemlerin yapılacağı sistemlere yada merkezlere iletimi sağlanır.

1.5.1. Kablosuz Algılayıcının Ağ Sistemleri

Kablosuz Algılayıcının Ağ Sistemleri çeşitli alanlarda kullanılmaktadır ve bir takım uygulamalar şöyle sıralı başlıklarla örnek verilmiştir.

1.5.2. Çevresel Gözlem ve Koruma

Gerçek zamanlı sıcaklık ve nem ölçümleri ile Şekil 4'deki gibi çevresel gözlem sistemi [13] olası yangınların tespiti veya kıymetli varlıkların ve sanat eserlerinin korunması uygulamalarında kullanılabilir. Çevresel Gözlem Sistemi iç ve dış mekanlarda

kullanılabilecek sıcaklık ve nem algılayıcıları içeren algılayıcı noktaları ve bu bilgileri işleyip gerekli alarm durumlarını oluşturabilen ana birimden oluşur.

Şekil 4. Çevresel gözlem ve koruma

1.5.3. Akıllı Tarım

Kablosuz algılayıcı ağlarının bir başka önemli uygulama alanı Şekil 5'te de gösterildiği gibi akıllı tarımdır [13]. Akıllı tarımda amaç yerel iklim ve toprak koşullarını gözlemleyerek ürünün en optimum ve verimli bir şekilde üretilmesidir.

Şekil 5. Akıllı tarım

Bu bağlamda, kablosuz algılayıcı ağ sistemleri, ürünün yetiştiği çevrenin çevresel parametrelerinin kesin bir şekilde gözlemlenmesinde kullanılır. Sıcaklık, nem ve ışık algılayıcıları kullanılarak ürünlerin don olasılığına karşı korunması ve ürünlerin sulama miktarının toprağın durumuna göre kontrol edilmesi mümkündür.

1.5.4. Endüstriyel Otomasyon ve Bakım

Kablosuz algılayıcı ağ sistemleri, endüstriyel proseslerdeki akışkanların gözlemlenmesi [13] veya üretim hattındaki makinelerin kontrolünü uzaktan düşük maliyet ile gerçekleştirmekte önemli katkılar sunar. Bu sistemler Şekil 6'daki gibi kablosuz olmaları sayesinde hareketli parçaların yoğun olarak kullanıldığı endüstriyel proseslere kolayca adapte edilebilir.

Şekil 6. Endüstriyel otomasyon ve bakım

1.5.5. Tedarik Zinciri ve Varlık Yönetimi

Kablosuz algılayıcı ağ sistemleri, varlıkların hem nakliyesi hem de depolanması sırasında bulundukları durumların gözlemlenmesinde [13] kullanılabilir. Şekil 7'de gösterildiği gibi bu sistem ile soğuk zincir nakliyesi gerçekleştirilen bir ürünün soğuk zincirinin bozulup bozulmadığını veya kıymetli bir tablonun nakliye sırasında çok fazla neme maruz kalıp kalmadığını tespit edilebilir.

Şekil 7. Tedarik zinciri ve zarlık yönetimi

Nakliyeye verilen bir ürünün taşınma sırasında düşürülüp düşürülmediğini, darbe alıp almadığını, fazla sıcaklıklara maruz kalıp kalmadığını veya ürünün paketinin izinsiz bir şekilde açılıp açılmadığı öğrenilebilir.

1.5.6. Güvenlik ve Kontrol

Kablosuz algılayıcı ağ sistemleri binalardaki, evlerdeki, fabrikalardaki kapı ve pencerelerin durumlarını gözlemleyerek ve hareketli cisimlerin tespitini sağlayarak bu ortamların güvenliğine [13] katkı sağlayabilir. Şekil 8'de bir güvenlik ve kontrol uygulaması gösterilmektedir.

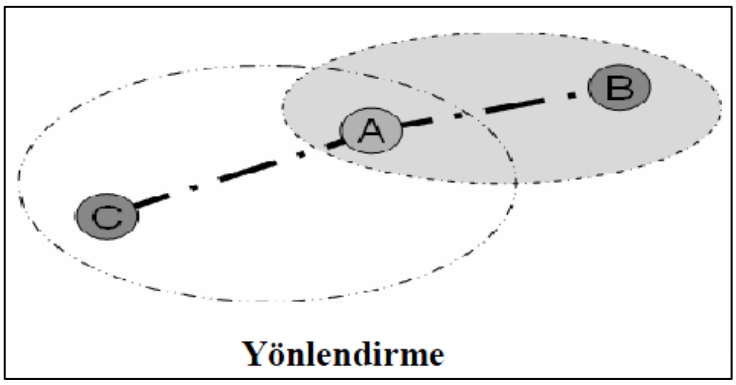
Şekil 8. Güvenlik ve kontrol

1.6. Kablosuz Algılayıcı Ağların Konu Kapsamı

Kablosuz algılayıcı ağların (*KAA*) tanıtılması, *KAA* düğüm yapısını, *KAA* uygulama alanlarını, katmanlı ağ yapısını, geliştirilen ortam erişim ve yönlendirme protokollerini, veri toplama yöntemlerini, güvenlik gereksinimlerini öğrenme, kablosuz algılayıcı ağların benzetim gerçekleştirme, gerçek düğümler üzerinden uygulama gerçekleştirme. Kablosuz algılayıcı ağlarda ticari ve bilimsel uygulamalar, temel algılayıcı ağ teknolojileri, kablosuz algılayıcı ağ protokolleri, fiziksel katman, ortam kontrol katmanı, yol bulma katmanı, taşıyıcı katman, algılayıcı ağlar için ara çözümler, kablosuz algılayıcı ağların yönetimi, algılayıcı ağlar için işletim sistemleri, performans ve trafik yönetimi.

1.6.1. Algılayıcı Ağlarında Yönlendirme

Algılayıcı ağlarında bulunan düğümler kablosuz teknolojiler kullandıkları için kapsama alanları oldukça sınırlıdır. Bir düğüm ancak komşularının kapsama alanı dahilinde bulunan diğer düğümler ile iletişim kurabilmektedir. Şekil 9'da da gösterildiği gibi C ve B düğümlerinin iletişim kurabilmesi için A düğümünün yönlendirme yapması gerekmektedir. Bu yüzden ağ içinde iletimin sağlanması için yönlendirmeye gerek duyulmaktadır.



Şekil 9. Algılayıcı ağlarında yönlendirme

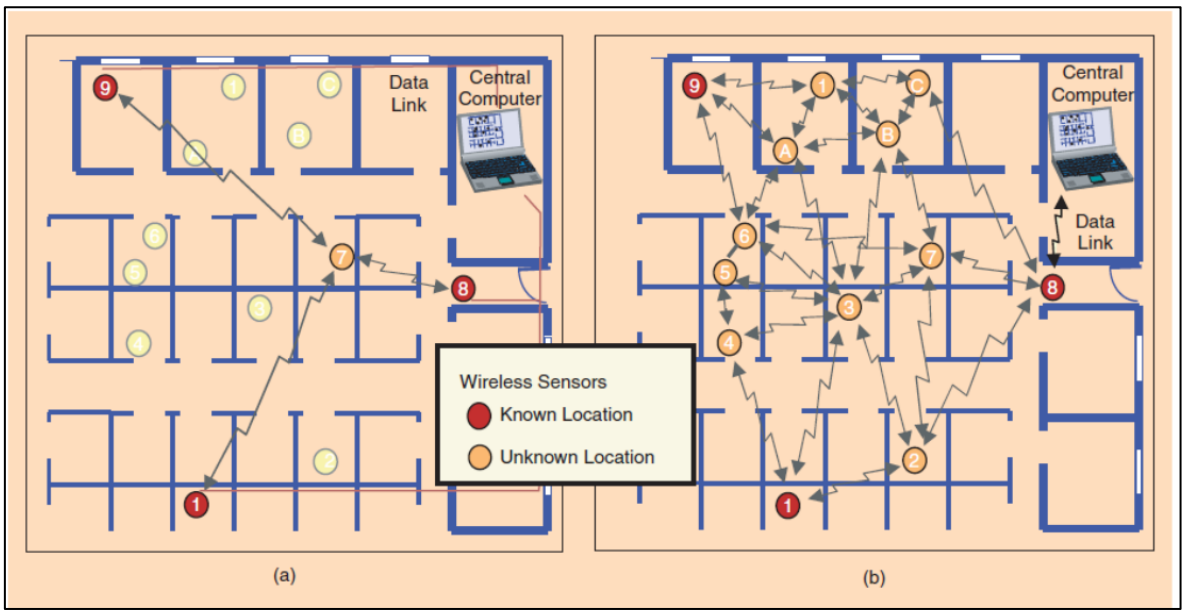
Algılayıcı ağlarında normal koşullar altında klasik ağların tersine genellikle ağ içinde herhangi bir iletişim yoktur. Yani başka bir deyişle algılayıcı düğümleri kendi aralarında haberleşmezler. Algılayıcı düğümlerinin esas görevi fiziksel ortamdan algıladıkları verileri BAZ istasyonuna ulaştırmaktır. Bu yüzden algılayıcı ağlarındaki yönlendirme biçimi doğrultulmuş N den 1 e yönlendirme olarak adlandırılmaktadır. Burada N adet algılayıcı düğümünden bir tane BAZ düğümüne yönlendirme yapıldığı anlatılmak istenmektedir. Bu farklı haberleşme biçimi günümüz klasik yönlendirme protokollerinin [14] bu ağlara uygulanmasını zorlaştırmaktadır. Klasik protokoller ağ içinde tüm düğümlerin haberleştiği varsayılarak inşa edilmişlerdir. Algılayıcı ağlarında düğümler olay bazlı veya periyodik olarak ölçtükleri verileri BAZ istasyonuna aktarmaktadırlar. Olay bazlı sistemlerde ölçülen değer belli eşik değerlerinde ise aktarılmakta aksi halde sadece ölçüm yapılmaktadır. Örneğin bir düğüm sadece sıcaklık değeri 20 C 'nin üzerinde olduğunda haber verecek şekilde programlanabilir. Periyodik sistemlerde ise düğümler belirli zaman aralıklarında ölçümler yaparak ölçüm değerlerini periyodik olarak BAZ istasyona aktarırlar.

1.6.2. Yönlendirme ve Heinzelman'ın LEACH Protokolü

Kümeleme (*Clustering*) [18], algılayıcı ağlarının yönetimde önemli bir metot olup, kümeleme sayesinde mesaj transferlerindeki enerji kullanımı düşürülebilinir. Bir diğer önemli metot da kapsayan ağaç oluşturmaktır. Hiyerarşik küme-tabanlı veya (*cluster-based*) yönlendirme ilk olarak kablolu ağlarda önerilmiştir. Bu yönlendirme, sistem kapasitesini ayarlayan ve verimli haberleşmeyi sağlayan avantajlara sahiptir. Bununla birlikte hiyerarşik yönlendirme kavramı, kablosuz algılayıcı ağlarında verimli enerji kullanımlı yönlendirmeye yardımcı olmaktadır. Bir hiyerarşik mimaride, az enerjili noktalar hedefle ilgili algılamalar yaparken yüksek enerjili noktalar bilgiyi işleme ve iletme görevini üstlenirler. Bu ise, kümelerin oluşturulması ve kümenin baş elemanlarına (*Cluster head*) özel görevlerin tahsisiyle kapasitenin ayarlanabilirliliğini, ağ ömrünün artırılmasını ve enerji verimliliğini sağlamaktadır. *Heinzelman et al* [19] algılayıcı ağları için düşük enerjili adaptif kümeleme hiyerarşisi (*low energy adaptive clustering hierarchy-LEACH*) [18] isminde hiyerarşik kümeleme algoritmasını önermiştir. *LEACH* protokolü ağ ömrünü artırmasına rağmen bu protokolde bazı kabuller tartışma konusudur. *LEACH,* dağıtılmış küme oluşumunu içeren küme tabanlı bir protokoldür. Daha az enerji tüketimi için kümelerin baş elemanlık rolü rasgele bir şekilde dönüşümlü olarak verilir ve enerji harcaması ağdaki algılayıcılara paylaştırılır. *LEACH*, kapasite ayarlanabilirliliği ve dinamik ağlardaki dayanıklılığı (*Robustness*) sağlamak üzere yerel bir koordinasyon kullanır ve baz istasyonuna gönderilecek bilgi miktarını azaltmak üzere veri-birleştirmesini (*data fu*sion) gerçekleştirir. Ayrıca çalışmada küme içi ve kümeler arası çakışmaları azaltmak amacıyla *TDMA*/*CDMA MAC* kullanılmaktadır. *Tillett* ve arkadaşları [19] tarafından önerilen çalışmada *PSO* (*Particle Swarm Optimization*) olarak bilinen optimizasyon tekniği ile algılayıcı noktaların kümelendirilmesi probleminin çözülmesi hedeflenmiştir. *PSO* yaklaşımı, verilen bir problem için en iyi çözümün bulunmasında birbirleriyle etkileşim ve işbirliği içinde test sonuçlarını üreten 'böceklerin' kullanıldığı gelişime dayalı programlama tekniğidir. Tipik bir optimizasyon probleminde fonksiyon veya uygunluk (*fitness*) değerleri kriter olarak kullanılmaktadır.

1.6.3. Lokalizasyon Problemi

Kablosuz Algılayıcı Ağlarındaki Algılayıcı düğümler tarafından algılanan veri yada olayın nerede oluştuğu bilgisi önemlidir (Çevresel görüntüleme, trafik yönetim sistemi orman yangın alarm sistemi, su kalite ölçüm uygulamaları vb). Bu amaçla Şekil 10'da gösterildiği gibi Algılayıcı düğümlerinin koordinatlarının [15] belirlenmesi gereklidir. Sonuç olarak konum bilgileri elde edildiği takdirde daha verimli yönlendirme algoritmaları oluşturulur. *WSN*'lerde iki çeşit konumlandırma yaklaşımı söz konusudur.



Şekil 10. Lokalizasyon problemin şeması

Bunlardan birincisi *pre-localization* adı verilen önceden konumlandırma yaklaşımdır. Bu yaklaşımda düğümlerin kurulum ve konum ölçümü yapılır. Daha sonra konum bilgisi algılayıcı düğümün belleğine kaydedilir. Bu yaklaşımın dezavantajı konum değişikliği oluşan bir Algılayıcı düğüm konum bilgisi olarak eski konumunun bilgisini sürekli verecektir. Dolayısıyla gezgin mobil *KAA* için kullanışlı değildir. İkinci yaklaşım ise *self-localization* [16] adı verilen kendi kendini konumlandırma yaklaşımıdır. Bu yöntemde gerçek zamanlı ölçümlerle konumlar hesaplanır. Böylece hareketli düğümler gerçek zamanlı konumlarını güncelleyebildiği için hareketlilik (*mobility*) problemi çözülmüş olur. Kendi kendini konumlandırma yaklaşımında *GPS* gibi geleneksel konum belirleme metotlarının kullanımı pek uygun değildir. Bunun nedenlerini şöyle sıralamak mümkündür:

- Pahalı bir yöntemdir (her düğüme bir *GPS* donanımı gerektirir).
- Askeri uygulamalarda karıştırma (*jamming*) özelliği kısıtlıdır.
- Açık alan (*Outdoor*) uygulamalarla sınırlıdır.

GPS yerine bir *KAA*'da m adet kendi konum bilgisini ya *GPS*'ten ya da bir bilgi merkezinden elde eden düğüm noktasının (*anchor nodes*) olduğu göz önünde bulundurulur. Diğer taraftan yine aynı *KAA*'da bulunan n adet düğümün yerleri bilinmemektedir ve bu düğümler (*non-anchor nodes*) [16] kendi koordinatlarını belirlemeleri gerekir. Eğer yüksek güç iletimi olabilseydi referans düğümler vasıtasıyla konumlarını belirleyebilirlerdi. Fakat *KAA*'larda en önemli amaçlardan biri düşük güç tüketimidir. Dolayısıyla *Non-anchor* düğümlerin yerlerini belirlemesinde diğer düğümlerle işbirlikçi bir yaklaşım kurulur ve diğer düğümlerden yardım alınarak konum tahmini yapılır. Bu iki durum Şekil 10'da gösterilmiştir.

1.6.3.1. Düğümlerde Konum Hesaplama ve Mesafe Tahmini (*RANGE-BASED*)

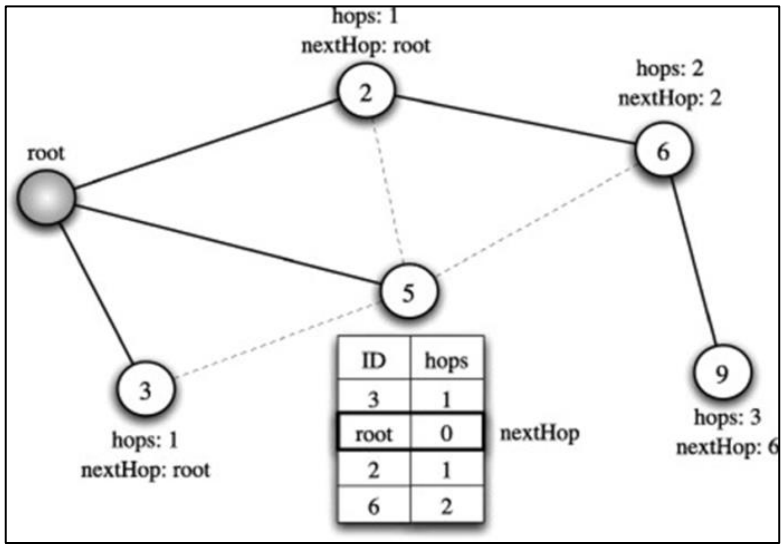
Self-localization [16] yaklaşımı göz önünde bulundurulduğunda neredeyse tüm ağ konumlandırmalarında iki temel aşama Konum Hesaplamaları ve Mesafe Tahmini mevcuttur. Bu aşamada ölçüm-tabanlı (*Measurement-based*) 4 adet teknik kullanılır :

- Varış Açışı (*Angle of Arrival*)
- Varış Zamanı (*Time of Arrival*)
- Varış Zamanı Farkı (*Time Difference of Arrival*)
- Alınan Sinyal Gücü Göstergesi (*The Received Signal Strength Incicator*)

1.6.3.2. Düğümlerde Konum Hesaplama ve Mesafe Tahmini (*RANGE-FREE*)

Konumlandırma konusunda var olan *Range-Based* yaklaşımı açıklandığı üzere konum tespiti yaparken ilave donanımlara gereksinim duyabilmektedir. Bu durum maliyeti arttıran bir etkendir. Diğer taraftan son zamanlarda oldukça popüler olan *Range-Free* [17] yaklaşımı ise ilave bir donanıma gerek duymadan sadece sıradan RF modülleriyle ve çeşitli optimizasyon algoritmalarıyla konumlandırma işlemini gerçekleştirebilmektedir. Böylelikle maliyet hatırı sayılır şekilde azalmaktadır. Zaten araştırmalarda ve pratik çalışmalarda çok popüler hale gelmesinin nedeni de budur. *Range-Free* yaklaşımında

konum tahmini yapılırken atlama bilgisi temel alınır. Uygulanan yöntemleri [16] *Centroid, APIT, DV-Hop, Amorphous* gibi sıralamak mümkündür. Bu yöntemler yaklaşık bir konum bilgisi verir ama bahsedildiği gibi donanımsal basitlik, maliyet açısından uygunluk, az enerji tüketimi avantajları vardır.

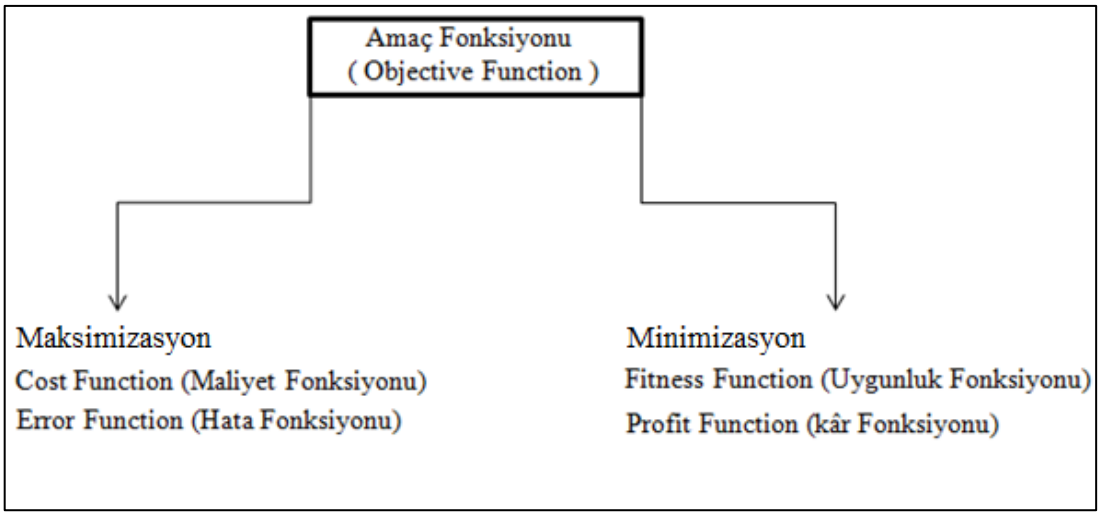


Şekil 11. Hop tabanlı konum saptama

Burda ağırlık merkezi (*Centroid*), üçgen içi yaklaşık nokta (*Approximate Point in Triangulation*), DV-Hop (*Distance Vector Hop*)**,** *Amorphous* [16], Kullanılan tekniklerdir. Şekil 11'de Hop tabanlı konum saptama örneği gösterilmiştir. Lokalizasyon konusunu kısaca toparlanırsa *RANGE-BASED* [17] yaklaşımı açıklandığı üzere konum tespiti yaparken ilave donanımlara gereksinim duyabilmektedir ve maliyeti yüksektir. *Range-Free* ise daha çok geliştirilen algotitmalara dayalıdır. Dolayısıyla donanımsal açısından basitlik, maliyet açısından uygunluk, az enerji tüketimi avantajları vardır.

1.7. Optimizasyon

Eğer amaç fonksiyonu (*Objective Function*) [25] bir minimizasyon probleminde tanımlanırsa maliyet (*Cost*) fonksiyonu olarak adlandırılır ve eğer maksimizasyon probleminde tanımlanırsa uygunluk (*Fitness*) fonksiyonu olarak isimlendirilmektedir. Optimizasyon problemlerinin genel şeması Şekil 12'de gösterilmiştir.



Şekil 12. Optimizasyon problemlerinin genel şeması

Eğer yapay sinir ağları (*neural networks*) gibi [26] bir modelleme (*modelling*) probleminin çözülmesi söz konusu olursa amaç fonksiyonun adı problemin türüne bağlı hata (*error function*) [26] ya da kar (*Profit*) fonksiyonu olarak tanılmaktadır. Genetik algotitmasında çaprazlama (*crossover*), mutasyon (*mutation*) operatörleri, yeni popülasyon seçimi, ebeveyn seçimi, genel olarak *Seçim* kelimesi önemlidir ve üzerinde durulmuştur. Sayılan husular problemin türüne bağlı farklı farklı ayarları vardır. Eğer problemin değişkenleri sıfır ve bir ya da ikili (*binary*) olursa yukarıdaki sayılan hususlar bu tür problemlere uyumlu ayarlanacaktır. Eğer problemin değişkenleri tam sayılı ise yukarıdaki sayılan hususlar buna göre ayarlanacaktır. Mesela seçim işlemi 1 ve 10 sayıların aralığında yapılması var sayılırsa seçimde her sayı bir kereden fazla tekrarlanamazdır ki burda permutasyon (*permutation*) [27] yapılması da söz konusu olmaktadır. Eğer problemin değişkenleri sürekli olursa yinede sayılan hususlar buna göre ayarlanmaktadır.

1.8. Genetik Algoritma

Genetik algoritma denilen kavram gerçek doğada yaşanılan olaylardan taklit yapılması ile gerçekleşmektedir. Sanki milyonlarca sene önceden programlanmış bir sürec kendisini bir takım cansız karmaşık moleküllerden en karmaşık varlığa yani insanoğluna ulaştırmıştır. Bu programın bir tek hücreden başlayıp insanoğluna varan süreci gerçekten şaheserdir ve bir mühendis ya da matematikci insanların açısından incelenmesine yer vardır. Bu süreçten taklit yapılma fikiri yaklaşık 40~50 yıl önce akıllarda oluşmuştur [20]

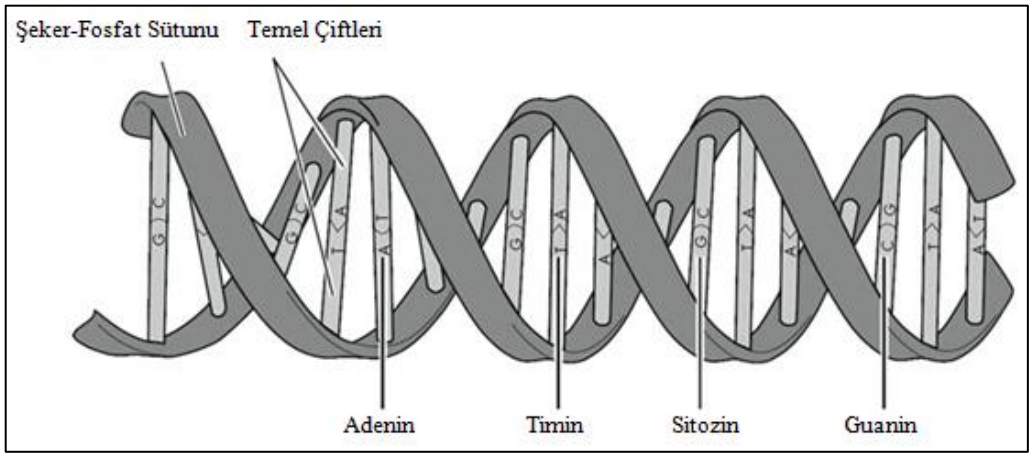
ve sonuç olarak genetik ve diğer evrimsel algoritmalar ile ortaya çıkmıştır. Genetik algoritma kuşkusuz en ünlü metasezgisel ya da evrimsel algoritma olarak tanınmaktadır. İlk önce bu algoritmayı almanlar ileri sürmüşlerdir ve ondan birkaç sene sonra amerika'da (*evolutionary programming*) [20] evrimsel programlama *L.J. Fogel* ve meslektaşları tarafından sunulmuştur [20]. Fakat bunlara rağmen herkes *J.H. Holland* [20] ile genetik algoritmasını tanımaktadır. İçinde yaşadığımız dünya hep değişmektedir (*stochastic*) ve sabit duran hiç bir durum yoktur. Birtakım etkenler istatistik açısından bakılırsa, sürekli değiştiği görülmektedir. Dünyada bu değişimlere uyum sağlanması için var olan her şey değişmektedir. Dünyadaki her şey bu stokastik ortamda iyi performansa sahip olmak isterse, bir o kadar da fazla uyumsal olması gerekmektedir. Bu uyum için birkaç teori gündemdedir. Yani *Darwin*'in çevreye uyum teorisi [20] ve *Lamarck*'ın evrim teorileri söz konusu olmaktadırlar. En çok *Darwin*'in teorisi tanınmaktadır. Yani burda doğa kendine en çok uyum sağlayan şeyleri seçmektedir, Ama *Lamarck* başka bir görüştedir yani her nesil birtakım deneyimleri (yetenekleri ya da özellikleri) biriktirmiş ve elde ettiklerini sonraki nesile aktarmıştır. Eğer *Darwin*'in teorisini bilgisayarda uygulanırsa sonuç genetik algoritma olacaktır. Ama *Lamarck*'ın evrim teorileri simulasyon yapılırsa sonuç evrimsel programlama (*evolutionary programming*) olacaktır ve buda başka tür evrimsel algoritmalarından sayılmaktadır. Bu süreç genel olarak evrim (*evolution*) adı ile tanılmaktadır. Evrim kelimesi tam olarak *evolution* terimini temsil etmeyebilir. Çünkü evrimi duyunca akla hep bir şeyin *evolution* geçirmesi için, elde etmek eylemi gelir. Halbuki bazen yılan gibi bir canlı elini ve ayağını kaybederek *evolution* geçirmiştir. Başka bir deyiş ile birşeyleri kaybederek de *evolution* gerçekleşebilmiştir. Bazen ne elde ederek nede kaybederek örneğin: suyun buhara dönüşmesi gibi evrim gerçekleşir. Bu tezde evrim terimi deyim olarak *evolution* anlamında kabul edilip ve kullanılmıştır. Evrimde birkaç olay paralel ya da seri bir şekilde yönlü kombinasyon yapılarak neticesi evrim olmuştur. Bunların hepsinin matematik modelleri vardır. Genetik algoritmasında *Darwin*'in teorisinden gelme doğal seçim (*natural selection*) [33] konusu kullanılmaktadır. Her şeyde doğanın kriterlerine en çok uyum sağlayabileni seçilir ve diğerleri her nekadar da iyi olsalar bile uyum derecesinin azlığından dolayı elenirler. Bir zamanlar doğanın en güçlü hakimleri dinozorlar olduğu söylenmiştir. Ancak önlerine çıkan yeni durumlara uyum sağlayamadıklarından dolayı nesilleri elenerek tükenmiştir. Diğer konulardan biri de canlılarda kendini çoğaltma (*self replication*) [20,21] duygusu ve motivasyonudur. Yani hayat karmaşık moleküllerden başlayıp şimdiye kadar insanlarda da kendini çoğaltma

duygusu hep olmuştur. Bir diğer konu mutasyon (*mutation*) operatörüdür ve rastgele ya da rastgele olmayan bir etkenlerin etkisi altında bir takım planlanmamış ya da programlanmamış değişimlere neden olmaktadır. Bazen doğada bu programlanmamış değişimlerin sonucu çok kötü olmuştur. Genetik algoritmasında bazen mutasyon çok iyi sonuçlara neden olmuştur. Başka bir konu da birliktelik ya da bir arada yaşamak (*coexistence, simbiosis*) [20,21] konusudur. Örneğin; kedi ve köpek insanlarla bir arada yaşadığından dolayı diğer hemcinslerinden daha zekalılardır. Bu konular hepsi genetik algoritmasında matematik modellemesi yapılmıştır. Sanki tabiatta bildiğimiz tüm yaratıklar canlı ya da cansız, yazılan genetik algoritmasının programı milyonlar sene önceden enter'e basıp koşturulmuştur. Şekil 13'e bakarsak yaklaşık bir takım uygun cevaplara yakınsama oluştuğunu görebiliriz. Örneğin: kutup ayısı ile çöl ayısı kıyaslaması, orman tilkisi ile kutup tilkisi kıyaslaması gibi her biri kendisine özel özelliklerini taşıyorlar ve bu nekadar onların ortama uyum sağlayabildiklerini göstermektedir.

Şekil 13. Çevreye uyumunun bir örneği

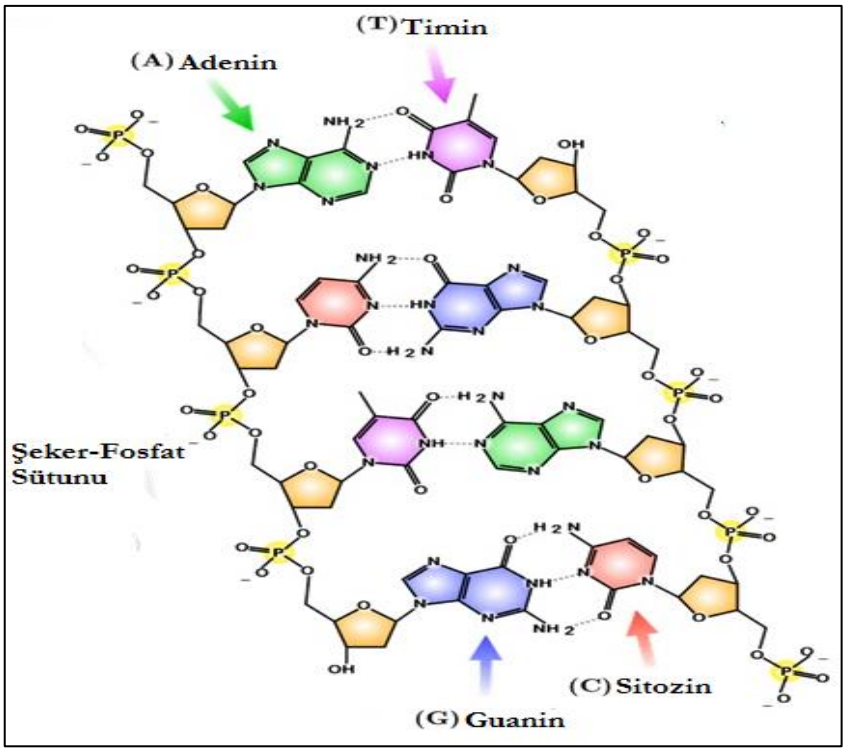
Mesela kutup ayısının açlıktan ölmemesi için koku algılama duyusu çok gelişmiştir ve yemek kokusunu kilometrelerce uzaklıktan algılayabilir. Çünkü kutupta besin kaynağı sınırlıdır ve diri kalmak için yemeği uzakta olsa bile bulup yemesi gerekir. Öte yandan besin kaynağı bol olan bölgelerde böyle bir güçlü koku algılama duyusuna gerek yoktur. Koku duyusu zayıf olan ayılar az yemek bulup ve yetersiz beslenme'den dolayı bu diri kalma yarışmasında yenilerek ölmüşlerdir. Kutup tilkisinin de ses algılama duyusu çok güçlüdür ve en küçük hareketleri metrelerce karın derinliğinde olsa bile işitebilmektedir. Bütün bunlar bizi bilimsel dili ile genetik bilimini tanımlamaya yönlendirmektedir.

Milyonlarca sene doğanın çabasının mahsülü insanoğludur. Dolayısıyla insanoğlu bu süreçte deneyimlerini ve bilgilerini kimyasal moleküllerden oluşan *DNA* (*deoxyribonucleic acid*) [21] adlı bir bellek'te saklamıştır. Tabiat *DNA*'yı oluşturarak bu bilgileri onun kimyasal bloklarında saklayarak sonraki yeni nesile aktarmıştır. Ayrıca yeni nesiller gereken değişimlerini aynen *DNA* üzerine uygulayarak bu süreci sürdürmektedirler. *DNA* karmaşık yapısıyla tüm canlı yaratıklarda mevcuttur. İnsan ile bitki arasında yüksek bir yüzdede ortak benzerlik vardır ve bazen yüzde 30'lara kadar çıkabilmektedir. Yeryüzünde herhangi iki insan arasında yüzde 99 ortak bir şekilde gen benzerliği vardır. Bu yüzde 1 farklılık oranı tüm bu çeşitliliklere neden olmuştur. *DNA* [21] karmaşık bir sarmal merdivan yapısı şeklindedir.



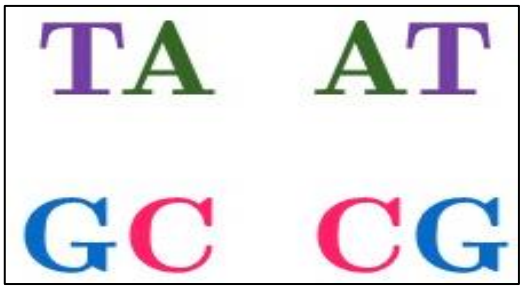
Şekil 14. DNA'nın yapısı

Şekil 14'te görüldüğü gibi bir şeker-fosfat omurgası (*Backbone*) birtakım temeller ile birbirine bağlanmıştır. Bu temeller 4 çeşitten ibarettir [21]. Bunlar Adenin (*Adenine*), Timin (*Thymine*), Sitozin (*cytosine*), Guanin (*Guanine*) gibi bir takım organik bazlardan oluşmaktadırlar.



Şekil 15. Şeker-fosfat omurgası

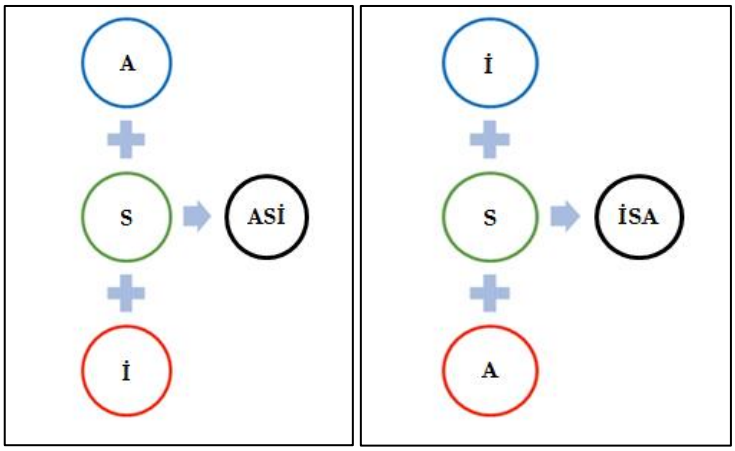
Şekil 15'te görüldüğü gibi bazı yerlerde Adenin-Timin hidrojenli bağlantısı kurulabilirken tam tersi Timin-Adenin bağlantısı da kurulabilmektedir. Aynen bu şekilde Sitozin-Guanin bağlantısı kurulmuştur. Dolayısıyla Şekil 16'daki gibi bunlardan dört çeşit kombinasyon oluşmaktadır.



Şekil 16. Hidrojenli bağlantılı kombinasyonlar

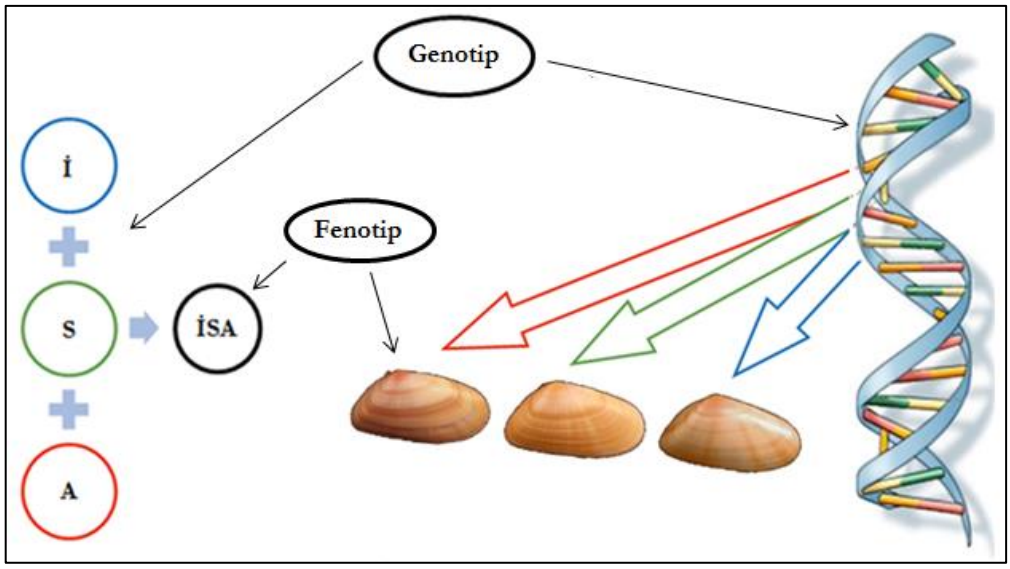
Bilgisayarda tüm bilgiler sıfır ve bir ile kodlanmıştır ve bukadar çeşitlilik 0 ve 1 ile oluşmuştur. Benzer bir şekilde tabiat böyle bir yaklaşımla çalışmıştır, Fakat taban 2 yerine 4 tabanında yani dört *TA, AT, GC, CG* elemanlarıyla çalışmıştır. Bunların çok sayıda dizilişi ve kombinasyonu daha da çok çeşitliliğe neden olmuştur. Genin [21] bir takım

dizilerinin özel yerlerinde mesela 1 milyar tane var sayılsın, 1000'inci blok'tan 1010'nuncu bloğa kadar yani 10 tane temel göz rengi ile ilgili olabilir. Tabiki bunlar var sayılmıştır ve bu sayılar oldukca fazladır ve 10 tane ile sınırlı değildir. Fakat bir canlının genlerinde bazı yerler kesin bir takım dışsal özelliklerini etkilemektedir. Temeller deyim olarak genotip (*genotype*) [22] ve bunlardan oluşan çıktı, fenotip (*phenotype*) olarak adlandırılmaktadır. Mesela fenotip'e göz rengi örnek verilebilir. Genler *DNA*'nın özel kısımlarını oluşturur ve canlıların özelliklerini de temsil eder. Genetik bilimine bir alfabe gözü ile zama zaman bakılmıştır,. Yani tabiatın alfabesi sayılmıştır. Örneğin: kitaplar, makaleler, dergiler vs. bütün yazılı olan şeyler belki en fazla 100 tane alfabe harfleri ile yazılmıştır. Yani ortada büyük bir çeşitlilik (*diversity*) olduğu da görülmektedir. Sınırlı elemanlar ile nasıl böyle çeşitlilik sağlanır sorusu bir konudur ve bu dört *TA, AT, GC, CG* elemanları nedir sorusu bir başka konudur, Ayrıca bunların dizilişi nasıl ve kaç sayıda kombinasyon yapılması da apayrı bir meseledir. Örneğin : İ,S,A harfleri tek başına bir anlam vermemektedir. Ancak Şekil 17'deki gibi dizilişler ile bağlandıklarında İSA kelimesi oluşur ve Şekil 17'de gösterildiği gibi dizilişler değiştiğinde bambaşka anlamda ASİ kelimesi oluştuğu da görülmektedir.



Şekil 17. Alfabe ile bir genoip örneği

Şekil 17'deki İ, S, A harflerine genotip ve onlardan oluşan İSA ve ASİ kelimelerine fenotip denilir.



Şekil 18. Deniz kabağu ve alfabe ile fenotip ve genotip örnekleri

Deniz kabuğu gibi bir canlı var sayılsın, kabuğunun rengi, deseninin (*pattern*) bilgileri onun gen dizisinde saklıdır. Bu sınırlı genotip değişimleri nasıl bir sonuçlar vereceği Şekil 18 aracılığıyla kafada daha iyi canlandırılsın diye gösterilmiştir. *Coding*, bir canlının kendi gen dizisi ile birebir ilişkisi olursa bir takım kodlarla kodlama yapılır, Yani kodlama yapılarak kodların arasından optimum kod [20] bulunabilir. Adı geçen kodlar kendilerini fenotip şeklinde gösteriyorlar ve her fenotip tabiatın kriterlerine uyumlu olursa kalacaktır ve uyumu az olanlar elenecektir. Eğer bu prosedür diğerlere miras kalırsa sonunda özelliklerde bir nevi yakınsamalar görülecektir. Aynen doğada olan olaylar gibi evrimsel algoritmalar da buna benzemektedir.

1.9. Genetik Algoritmanın İşleyişi

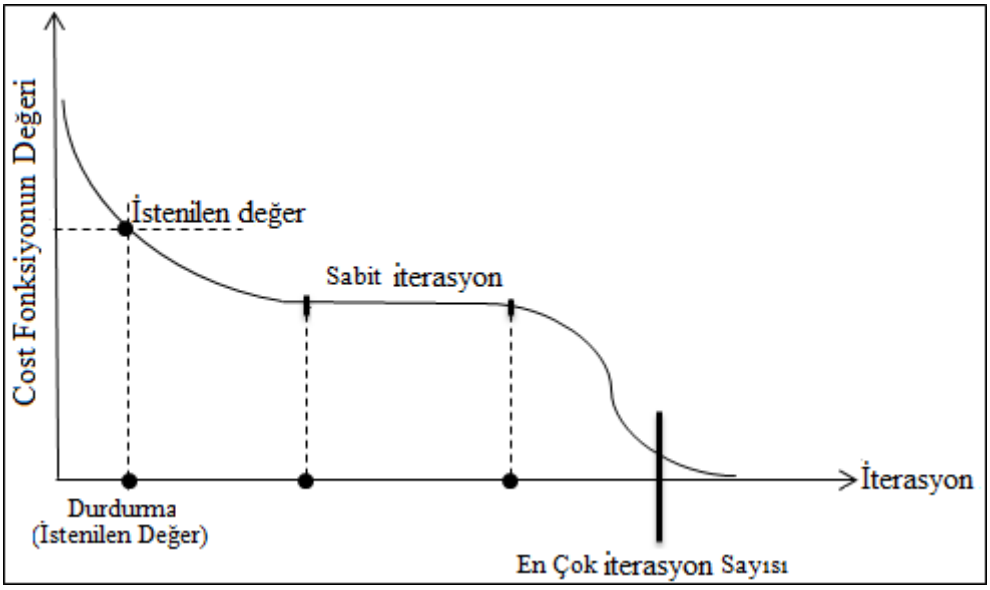
Aşağıda genetik algoritmanın işlem adımları sıralanmıştır [20,28]:

- İlk popülasyonun oluşturulması ve değerlendirilmesi
- Ebeveyn *seçimi*, çaprazlama (*Crossover*) uygulanması ve çocuklar popülasyonun oluşturulması
- Ebeveyn *seçimi*, mutasyon yapılması ve mutasyon popülasyonun (*offspring*) oluşturulması
- Ana popülasyonu, çocuklar popülasyonu, mutasyon Popülasyonların birleştirilmesi (*Merge*) ve *yeni popülasyonun* oluşturulması

- Eğer sonlandırma koşulları sağlanmazsa, ikinci aşamadan tekrar başlanır
- Son

1.10. Algoritmanın Sonlandırma Koşulları

İterasyon ve zaman kriterlerine göre iki tür sonlandırma durumu vardır.



Şekil 19. Sonlandırma koşullarının grafiği

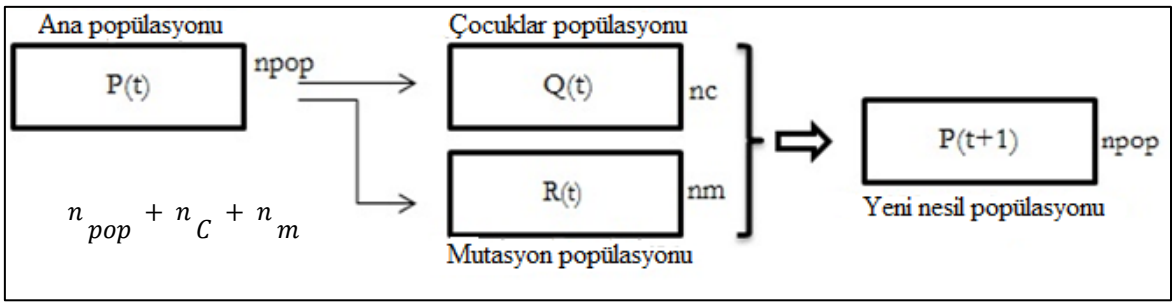
Şekil 19'da gösterildiği gibi eğer sabit iterasyon sayısı belli bir miktardan fazla olursa (*stall iteration*) [23] ozaman durdurma sırası yani algoritmanın sonlandırılmasının vakti gelmiştir. Zaman açısından da algoritma koşturulur. Mesela eğer 1 saatten fazla sürdüğü halde bir değişiklik tespit edilmese ya da değişimlerin genliği çok az yaklaşık 2% olursa algoritmanın sonlandırılmasının vakti gelmiştir ve programın fazla koşmasında yarar olmayacaktır. *NFE* (*number of function evaluation*) [24] algoritmanın performansını ölçmekte (*test*) en güvenilir yöntemdir ve (*cpu time*) [24] sınama yönteminden daha itibarlıdır. Önceki koşullar yani zaman ve iterasyon bilgisayardan bilgisayara değişebilir yani hızlı işlemciye sahip olan bilgisayar algoritma kötü olsa bile onu hızlı koşturarak iyi gösterebilir, Ama *NFE* bu durumdan bağımsız bir kriterdir ve denklem 1 ile ifade edilmiştir ve ilgili denklemi kullanarak *NFE* hesaplanabilmektedir.

Amaç fonksiyonun çağırılma sayısı (NFE) = (1)

Ana popülasyonun sayısı + {Çocukların sayısı + Mutasyonluların sayısı} * İterasyon sayısı

1.10.1. Genetik Algoritması İçin Çeşitli Senaryolar

Mutasyon [30] aşamasında üzerine mutasyon yapılacak popülasyonundan bir kopya alınır ve sonra alınan kopya üzerine mutasyon işlemi yapılır.

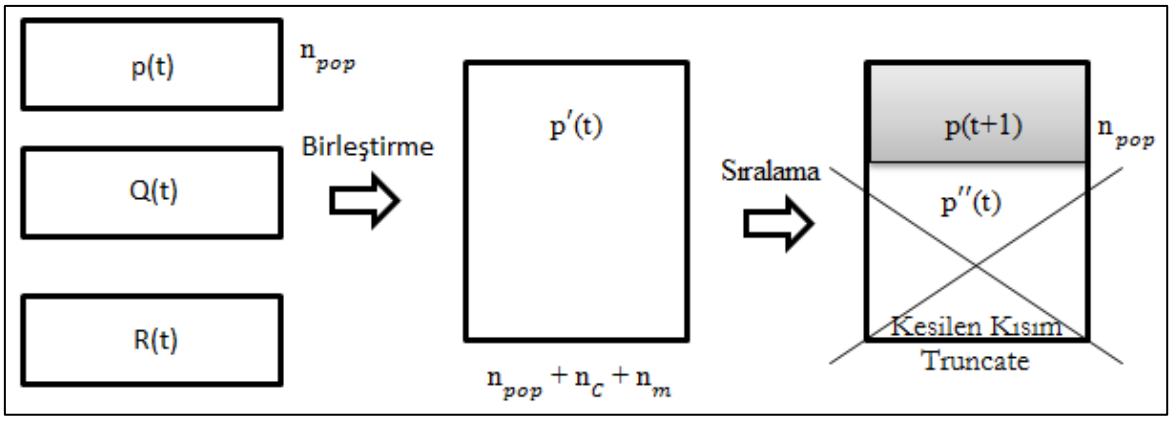


Şekil 20. Genetik algoritmasının popülasyon senaryosu

Şekil 20'deki popülasyonların toplamından n_{pop} kadarı Seçilir [28]. Bunun için de kaç Senaryo Vardır.

1.10.2. Birleştirme (*Merge*), Sıralama (*Sort*), Kesme (*Truncation*)

Popülasyonların toplamından n_{pop} kadarının en iyileri seçilir ve gerisi kesilerek (*Truncate*) [28] elenir. Seçilen en iyileri p(t+1) [29,30] olarak Şekil 21'de gösterilmiştir.



Şekil 21. Birleştirme, sıralama ve kesme senaryosu

Kesilmede birçok durum ortaya çıkabilir. Belki tüm ana popülasyon elenir, belki de mutasyonların popülasyonu elenir ve daha önemlisi belkide seçilenler yeni nesilin oluşumunda payı olmayacaktır ki bu ihtimal çok kötü bir durumu göstermektedir. İlk popülasyonu seçmek için belli bir kural yoktur. Fakat problemin boyutu bunu belirleyebilir. Örneğin: ikili (*Binary)* bir problemin n değişkenli olduğu varsayılsın. Bu problemin olası tüm cevapları 2^n sayısındadır. İlk popülasyonun seçiminde ilk başta 2^n boyutunda bir popülasyonu vardır ve 2^n kadar durumu incelemek mantıklı değildir. Burda problemin boyutu kendisi birazda olsa ilk popülasyonunu kısıtlamıştır. Sürekli problemlerde bu mantık olumlu değildir. Hatta eğer problem bir değişkenli olsa bile sonsuz kadar durum vardır ve yinede incelenmesi olumsuzdur. Genelde iyi bir algoritma arama uzayından 40 kere büyüklüğünde cevabı bulması gerekir. n_{pop}, n_C, n_m parametreleri birbirinden ayrı değerlerdir fakat bu genellikle n_{pop}'a göre belirlenir. Çocukların popülasyonu ana popülasyondan fazla olmaması gerekir, Ama yinede bunda sınır yoktur ve fazlada olabilmektedir.

$$n_c \simeq p_c * n_{pop}, 0 \leq p_c \leq 1 \; ya \; 0 \leq p_c < 1, p_{Crossover} \tag{2}$$

$$n_c = 2\left[\frac{p_c n_{pop}}{2}\right], n_{Crossover} \tag{3}$$

Denklem 2'deki p_c bir katsayıdır ve denklem 3'deki n_c'de çift sayı [30] olması gerekir. Çünkü çaprazlama (*Crossover*) operatörü hep iki ebeveyn alarak iki çocuk

oluşturur. Yani ebeveyn sayısı çocukların sayısıyla eşittir. Tabiki işin özünde bu böyle değildir yani doğada tek ebeveynli (*Single Parent*) de vardır. Aslında tabiatta cinsiyet meselesi de vardır. Bu konuda farklı farklı senaryolar olabilir, Ama burda en yaygın olanını (*Basic*) incelenmiş ve kullanılmıştır. Denklem 4 ve 5'deki mutasyona gelince böyle değildir ve şöyle ifade edilir.

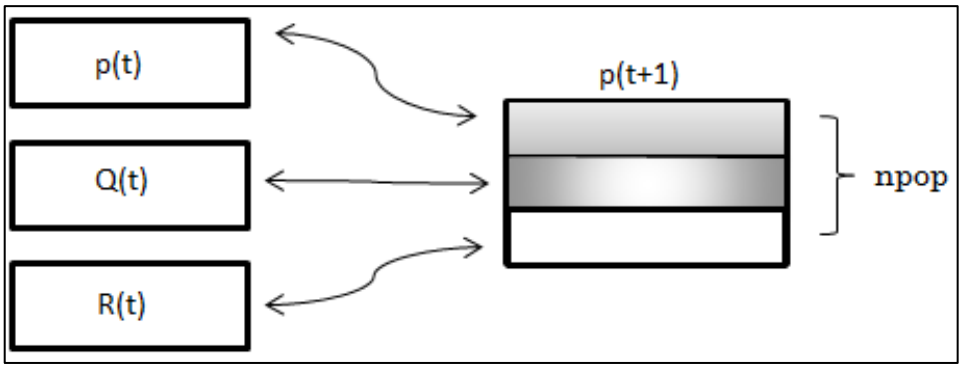
$$n_m \simeq p_m * n_{pop}, 0 \leq p_m \leq 1 \, ya \, 0 \leq p_m < 1, \, p_{Mutasyon} \tag{4}$$

$$n_m = \left[p_m * n_{pop}\right], n_{Mutasyon} \tag{5}$$

Şu hususun bilinmesinde fayda vardır, çaprazlama işleminin sonucu çocuk (*children*) ve mutasyon işleminin sonucu yinede çocuk (*Offspring*) [29] olarak adlandırılmaktadır.

1.10.3. Önceden Tanımlanmış Paylaşım (*Predifined Share*)

Mahrum bölgelerde ve önceliği az olan popülasyonlara yinede yeni popülasyonun oluşumunda payı olmalarına fırsat tanınır. İstenilen cevap bu mahrum bölgelerin içinde olabilir ve belki de durum bir mutasyon ile düzeltilebilir [32,30].



Şekil 22. Önceden tanımlanmış paylaşım senaryosu

Popülasyonun fazlalaşmaması için denklem 6'nın ifadesi kullanılır.

$$S_p + S_Q + S_R = 1 \tag{6}$$

P(t+1)'de, P(t)'nin payı Şekil 22'de gösterilmiştir ve şöyle yazılmıştır. Ayrıca P(t) önceki nesil demektir.

- $P(t)$ *'nin Payı* $= S_P * n_{pop}, 0 \leq S_P \leq 1$
- $Q(t)$*'nin Payı* $= S_Q * n_C, 0 \leq S_Q \leq 1$
- $R(t)$*'nin Payı* $= S_R * n_m, 0 \leq S_R \leq 1$

S_P'nin özel durumunun $S_P = 0$ olduğunda önceki nesilin şimdiki nesilde hiç temsilcisi olmadığı durumunu göstermektedir. Bu durumda *NFE* [55] sayısı değişmemektedir ve 7,8,9'daki matematik denklemleri ile şöyle yazılmaktadır.

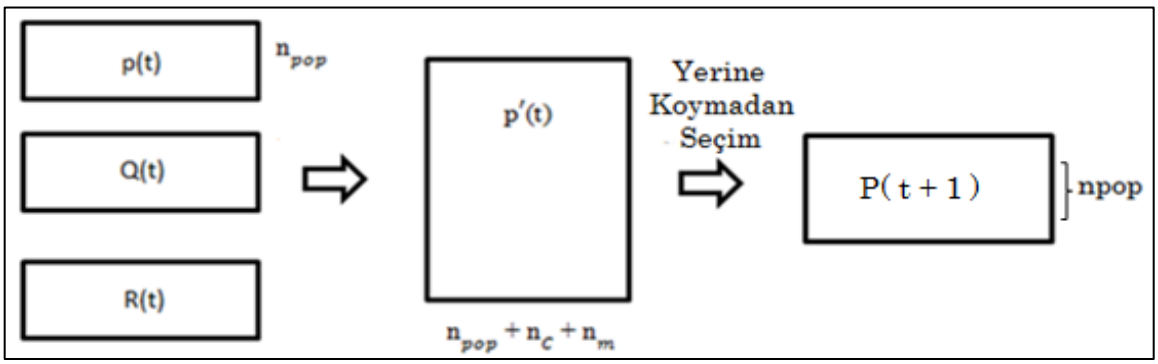
$$n_c = 2\left[\frac{p_c n_{pop}}{2}\right] \tag{7}$$

$$n_m = \left[p_m * n_{pop}\right] \tag{8}$$

$$NFE_{(t)} = n_{pop} + \left[n_C + n_m\right] * t \tag{9}$$

1.10.4. Birleştirme ve Rastgele Seçim (*Merge And Select Randomly*)

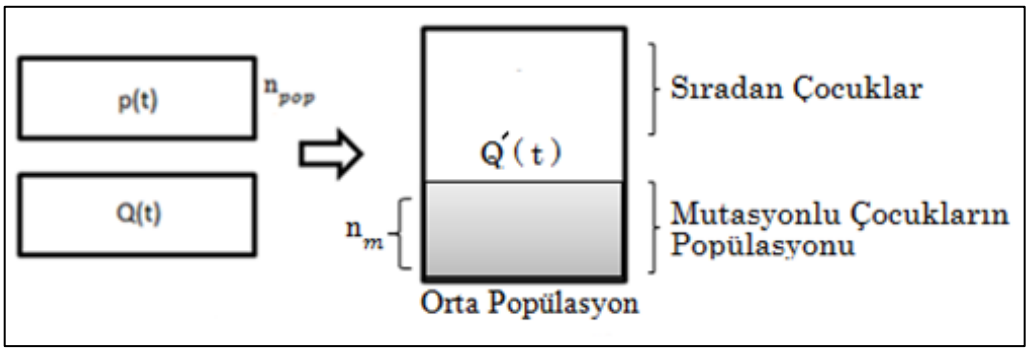
Burda Şekil 23'de gösterildiği gibi popülasyondan seçilen elemanlar elenmektedir. Buna yerine koymadan seçim denilir [27]. Böylece seçilmiş elemanların kaç kere tekrardan seçilme olasılığı ortadan kalkmış olacaktır. Ebeveyn seçiminde seçim, yerine koyarak yapılır ve ebeveynin tekrardan seçilmesi ihtimali vardır.



Şekil 23. Birleştirme ve rastgele seçimi

1.10.5. Mutasyonluların ve Çocukların Popülasyonlarının Bir Arada Olduğu Durumu

Şekil 24'te gösterildiği gibi orta popülasyonun senaryosunda *Crossover*'lerin üzerine de mutasyon [35,30] yapılmaktadır.



Şekil 24. Mutasyonluların ve çocukların bir arada olduğu durumu

Denklem 10 ifadesi, üzerine *c* ve *m* eklenilen orta popülasyonunu göstermektedir.

$$Q'(t) = n_{pop} + n_c \tag{10}$$

1.11. Nihai Popülasyonun (Sonraki Nesilin) Oluşumundaki Yöntemler

Ebeveyn seçimi ya üzerine mutasyon yapılacak popülasyonun seçimine kitaplarda ebeveyn denilir. Mesela bir ebeveyn üzerine mutasyon uygulanır ve çocuklar (*offspring*) [30] oluşur. Popülasyon elemanlarının yetkinliğini saydırılması gerekir ve yetkinliği fazla olan elemanların ebeveyn olmak için seçilme şanslarının yüksek olması gerekir. Genelde yıkıcı, hafif ve rastgele olmak üzere 3 mutasyon türü vardır. Yıkıcı mutasyon zayıf cevapları mutasyon uygulanmasına maruz bırakır yani yıkıcı etkisi vardır. Hafif mutasyon yerel arama (*local search*) [55] gibidir ve iyi cevapları mutasyona maruz bırakır. Üçüncü ise rastgeledir (*random*) ve adından da zaten bellidir.

1.11.1. Ebeveyn Seçiminin Yöntemleri

Popülasyonlar'dan eleman seçiminde [35,30] genel olarak kaç yaklaşım kullanılmaktadır. Aşağıda ebeveyn seçiminin yöntemleri şöyle sıralanmaktadır:

- Rastgele Seçim
- Yetkinci ya Rütbe Tabanlı Seçim
- Rekabetçi Seçim (*Tournament Selection*)

Koşul 1: seçilen ebeveynin numarası ile ilgili rastgele değişkeni **:**

$$I \in \{1, 2, 3, \dots, n_{pop}\} \rightarrow P_i = P_r\{I = i\}, \text{P}_r : \text{İhtimal} \quad (11)$$

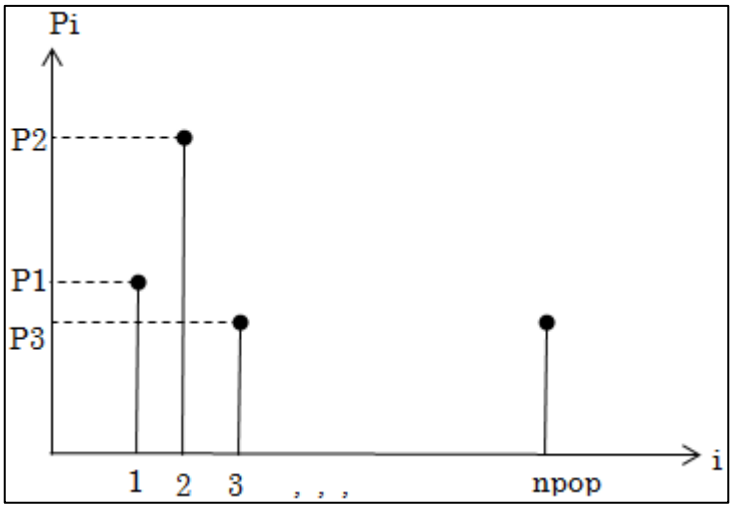
Tanımlanmış 11 ifadesinde ihtimal açısından I'dan bir örnek, i ile gösterilmiştir.

$$0 \leq P_i \leq 1 \quad (12)$$

$$\sum_{i=1}^{n_{pop}} P_i = 1 \quad (13)$$

Koşul 2: denklem 13'e bakarak ihtimal temellerine göre tüm olasılıkların toplamı 1 değerinde olması gerekir ve bu bir olayın kesin olacağını da göstermektedir.

Koşul 3: Yetkinci seçimi yönteminde $C_i \leq C_j \Leftrightarrow P_i \geq P_j$, i, j'den iyidir demektir. C_i ve C_j, i ya da j'inci popülasyonun elemanının *cost* değerleridir. Düşük maliyetler (*cost*) iyi sayılır ve $C_i \leq C_j \Leftrightarrow P_i \geq P_j$ ifadeleride gösterildiği gibi yüksek seçilme ihtimali küçük maliyetlere aittir. Yetkinci tabanlı seçiminde amaç fonksiyonun mutlak değeri (*fitness* ya da *cost*) [35] ve rütbe tabanlı (*rank base*) [31,55] yöntemler kullanılmaktadır. Amaç fonksiyonun mutlak değerine örnek olarak bir öğrencinin aldığı notu verebiliriz. Ancak rütbe farklı bir kriterdir. Dolayısıyla bu alanda hep maliyetlere göre inceleme yapılmıştır ve hep bir seçim kuralını tanımlamaya gayret edilmiştir. Bunları gerçekleştirmek için bir ayrık ihtimal dağılımı tanımlanması ve ondan örnekleme yapılması gerekmektedir.



Şekil 25. Ayrık ihtimal dağılımı

Burda tüm incelemeler maliyete göre yapılmaktadır. fakat rütbeye göre de yapılabilirdi. Şekil 25'deki ayrık ihtimal dağılımı fonksiyonu üzerinden nasıl örnekleme yapılacağına ilişkin yöntemler ilerliyen konularda incelenmiştir. P_i'ları Tanımlamakta Birinci Yöntem aşağıdaki gibi yazılmıştır**.**

$$P_i \propto C_{Max} - C_i \,,\; C_i \leq C_{Max} \tag{14}$$

tanımlanmış 14 ifadesinde C_{Max}'ın doğru tahmin edilmesi gerekir yoksa programda problemlere neden olacaktır. bahsedilen ebeveyn seçim yöntemleri konusunun üç koşulu sağlanması için tanımlanmış 15 ifadesi şöyle yazılmıştır:

$$P_i = \frac{C_{Max} - C_i}{\sum_j C_{Max} - C_i} \tag{15}$$

Böylece bir doğrusal orantı kurmuş olarak bahsi geçen üç koşul da sağlanmıştır. $0 \leq P_i \leq 1$ koşulu artık bölünün payı hep pozitif ve P_i sıfır ve bir aralığında olması için yazılmıştır. $\sum_{i=1}^{n_{pop}} P_i = 1$ koşulu da artık P_i 'nin sonucu hep 1 olduğuna göre yazılmıştır. Yetkinci koşulu zaten kurulmuştur. C_{Max} yerine bulunan en kötü maliyetin değeri hep güncellenerek kullanılabilir. İkinci yöntem şöyle aşağıda yazılmıştır :

$$P_i \propto \frac{1}{C_i + \varepsilon} \,,\; C_i + \varepsilon > 0 \;\text{ ve }\; C_i > -\varepsilon \tag{16}$$

tuhaflık (*Singularity*) durumu ortaya çıkmaması için tanımlanan 16 ifadesine bir ε eklenmiştir. Yinede bahsi geçen üç koşulun sağlanması için denklem18 şöyle yazılmıştır:

$$P_i = \frac{\frac{1}{C_i+\varepsilon}}{\sum_j \frac{1}{C_i+\varepsilon}} \tag{17}$$

Üçüncü yöntem boltzman (*Boltzmann*) [35,55] yöntemidir. İhtimalleri tanımlamak için Boltzman fonksiyonu denklem 19 ve 20'da gösterildiği gibi kullanılır ve tüm durumlar için geçerlidir ve çalışmaktadır.

$$P_i \propto e^{-C_i} \tag{19}$$

$$P_i = \frac{e^{-C_i}}{\sum_j e^{-C_j}} \tag{20}$$

Ama hala eksik bir şey daha vardır. Eğer bir elemanın diğeri diğer elemandan zayıf ise bunun ihtimali nekadar olacaktır. Burda ikisinin arasındaki çatlağı (farkı) azaltıp yükseltmesi istenilir. Bunun için maliyet üzerinde farklılıklar nekadar ihtimalleri etkileyebilir diye incelemeler devam etmektedir.

1.11.2. Seçim Baskısı (*Selection Pressure*)

Bunu gerçekleştirmek için seçim baskısı (*Selection Pressure*) [55,28,30] adında bir parametre tanımlanmıştır. Seçim baskısı = 0 olursa rastgele seçim gibi bir seçim işlemi yapılır ve eğer seçim baskısı $\rightarrow \infty$ olursa tekel bir şekilde yalnız bir eleman yani popülasyonun en iyisi seçilecektir ve her işlemin sonucu sadece 1 eleman olarak çıkacaktır. Aşağıda tanımlanmış 21 ifadesi ve denklem 22'deki P'_i ham ihtimallerdir.

$$P_i \propto (P'_i)^{\beta} . \quad \beta \geq 0 \tag{21}$$

$$P_i = \frac{(P'_i)^{\beta}}{\sum_j (P'_j)^{\beta}} \tag{22}$$

Örnek: Aşağıda P ihtimaller şöyle verilmiştir:

$$P'_1 = 0.1\ ,\ P'_2 = 0.2\ ,\ P'_3 = 0.3\ ,\ P'_4 = 0.4$$

$$\beta = 0 \rightarrow P_1 = P_2 = P_3 = P_4 = \frac{1}{4}$$

$$\beta \rightarrow \infty \Rightarrow \quad P_4 = \frac{1}{4}\ ,\ P_1 = P_2 = P_3 = 0$$

verilen örnekte β sıfır ve ∞ aralığında olması gerekir ($0 < \beta < \infty$) β'nın nasıl ayarlana bileceğinin üzerinde durulmuştur.

$$P_i \propto (C_{Max} - C_i)^{\beta} \Rightarrow P_i = \frac{(C_{Max} - C_i)^{\beta}}{\sum_j (C_{Max} - C_i)^{\beta}} \tag{23}$$

$$P_i \propto \frac{1}{(C_i+\varepsilon)^{\beta}} \Rightarrow P_i = \frac{\frac{1}{(C_i+\varepsilon)^{\beta}}}{\sum_j \frac{1}{(C_i+\varepsilon)^{\beta}}} \tag{24}$$

$$P_i \propto e^{-\beta C_i} \Rightarrow P_i = \frac{e^{-\beta C_i}}{\sum_j e^{-\beta C_j}} \tag{25}$$

β'nın yukarıdaki tanımlanmış 25 İfadesinde genelde hepsinden daha iyi çalışır ve kullanılmaktadır.

$$\beta \rightarrow 0 \Rightarrow \frac{1}{n_{pop}}\ , \forall\, i \tag{26}$$

$$\beta \rightarrow \infty \Rightarrow \begin{cases} 1, \text{ En iyi eleman} \\ 0, \text{ Diğerler} \end{cases} \tag{27}$$

Eğer kaç tane en iyi eleman olursa 1 değerindeki ihtimal aralarında bölünür. Örneğin: eğer 2 tane en iyi cevap olursa artı bide birbiriyle aynı olursalar ozaman her birinin ihtimali 1/2 değerinde olacaktır. Yukarıdaki ifadeden β → ∞ koyarak löpital (*l'hôpital*) kuralı ile limit alınırsa aynı cevaplara ulaşılacaktır. β'nın değeri şöyle uygun görülmüştür : $\sum_{i \in H} P_i$ = 0.8 ve en iyilerin yarısının kümesi demektir. *Dr. Mitsuo Gen* 'nin [32] kitabında β

değerini bulanık mantığı (*Fuzzy*) ile ayarlamıştır. Ancak yaygın olarak yazılan kural kullanılmaktadır.

1.11.3. Rütbe Tabanlı (*Rank Base*)

Aşağıdaki tanımlanan 28, 29, 30, 31 ifadelerinde r, Rütbe (*Rank*) anlamındadır ve 1. 11. 1 konusunda geçen üç koşulun sağlanması aynen burdada geçerlidir.

$$P_i \propto (n_{pop} - r_i + K)^{\beta} \tag{28}$$

$$P_i \propto \frac{1}{(r_i+\varepsilon)^{\beta}} \tag{29}$$

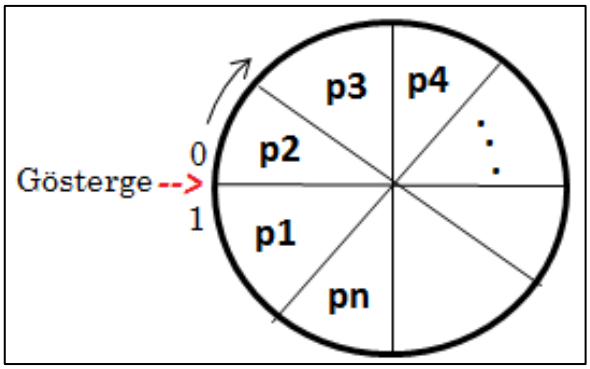
$$P_i \propto e^{-\beta r_i} \tag{30}$$

$$I \in \{1, 2, 3, \dots, n_{pop}\} \tag{31}$$

1.11.4. Rulet Tekerleği (*Roulette Wheel*) Örnekleme Tekniği

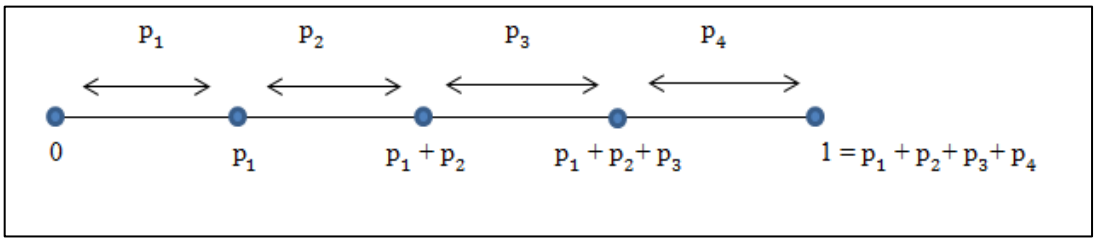
Daha önce bahs edildiği gibi ihtimal dağılımı fonksiyonundan örnekleme yöntemlerinden biri Rulet Tekerleği (*Roulette Wheel*) [55,31,35] tekniğidir. Bozuk para yada zar'ın sonuncu (*outcome*) ayrık dağılımlı fonksiyonu şeklindedir. Mesela bozuk paranın sonucunun değeri 2 tanedir yani seçilmesi ihtimali P_i ve seçilmemesi 1 - P_i ihtimalindedir. Burda bağımsız olarak seçim yapılabilmemektedir. Örneğin: zarın ihtimali hepsi 1/6 değerindedir. 6 gelmesi ihtimali 1/6 değerindedir ve gelmemesi 5/6 değerindedir. Zarın gelmemesi 5/6 ise hangisi gelecektir ozaman! yani bu durumda geriye zarın 5 tane gelmeye sayısı kalmıştır.

Eğer bir gösterge ve Rulet Tekerleği [55,31,35] Şekil 26'daki gibi olursa ve rastgele olarak tekerleği döndürürsek göstergenin p diliminin önünde durması ihtimali aynı p olacaktır.



Şekil 26. Rulet tekerleği

$$A_i = p_i \pi r^2, L_i = p_i 2\pi r \tag{32}$$



Şekil 27. Rulet tekerleğinin açılmış ve düzleştirilmiş biçimi

Uygulamaya kolaylık sağlanması için Şekil 27'de gösterildiği gibi Rulet Tekerleği [55,28] yuvarlağını açarak bir doğru oluşturulmuştur. Denklem 32'deki A_i tekerleğin alanı ve L_i çevresidir. Şekil 27'deki doğru dört kısma bölünmüştür ve örnekleme yapılması için bir rastgele düzgün dağılımlı (0,1) aralığında sayı tanımlanan 32 ifadesindeki gibi üretilmiştir.

$$r \sim u\,(0,1)\,,\ 0 \leq r \leq 1 \tag{32}$$

Her dilimin şansı uzunluğuyla aynı olmalıdır ve dilim ile eşit bir şekilde bölünmelidir. Yani p_1 uzunluğundaki bölge p_1 ihtimaliyle rastgele nokta kesin bu bölgede bulunması gerekmektedir.

- Eğer $r \geq 0 \,\&\, r \leq p_1$
- Eğer $r > p_1 \,\&\, r \leq p_1 + p_2$
- Eğer $r > p_1 + p_2 \,\&\, r \leq p_1 + p_2 + p_3$
- Eğer $r > p_1 + p_2 + p_3 \,\&\, r \leq 1$

Yukarıda bir ardışık koşul incelemesi yapılmıştır. Ve ya da *AND* (*conjunction*) önermesi r değeri üst sınırlar ile sırayla kıyaslanmıştır. $r \geq 0$ zaten hep doğrudur ve incelenmesine gerek yoktur. Dolayısıyla sadece $r \leq p_1$ doğru olursa r , 0 ve p_1 aralığındadır. $r > p_1$ önermesi incelenmesine gerek yoktur çünkü $r \leq p_1$ sağlanmadığı takdirde zaten $r > p_1$ demektir ve sadece $r \leq p_1 + p_2$ incelenir. Böylece *AND* (*conjunction*)'ların solundaki önermelerin incelenmesine gerek duyulmamaktadır. Yalnız sağdaki önermelerin incelenmesi yeterlidir ve bu rulet tekerleğinin çalışma mantığıdır.

- $i = min\left\{j \mid r \leq C_j\right\} \Rightarrow Find$ komutu
- $r \sim u\,(0,1) \Rightarrow rand$ komutu

$p_i = \sum_{j=1}^{i} p_j \Rightarrow$ CumSum Komutu (33)

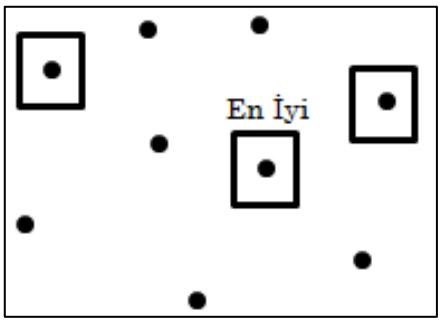
Rulet Tekerleği yukarıdaki komutlarla *MATLAB*'da [35] uygulanabilmektedir.

1.11.5. Turnuva Seçimi (*Tournament Selection*)

Buna *m-ary* turnuva seçimi adı da verilmiştir [55,28,35]. Önce eşit ihtimal ile popülasyondan m eleman rastgele seçilir ve sonra seçilenlerin arasından en iyisini rekabetim sonucu ünvanında belirlenir.

1.11.6. İkili Turnuva Seçimi (*Binary Tournament Selection*)

İkili turnuva seçiminde m = 2'dir, m =3 örneği için şöyledir:



Şekil 28. İkili turnuva seçimi

Şekil 28'de gösterildiği gibi m =3 olursa popülasyondan hep 3 eleman seçilecektir ve rekabet sonucunda en iyilerinden sadece bir tanesi seçilecektir. Burda en kötü eleman seçilemez çünkü zaten elenmiştir ve artık hiç bir rekabette yer bulmayacaktır. Başka bir deyişle turnuva yönteminde [55,30] m – 1 popülasyonun kötü elemanı hiç bir şekilde seçilme şansı yoktur.

- m = 1 → β = 0 : Rastgele seçimi durumu
- m = n_{pop} → β = ∞ : Seçim baskısı sonsuz

Yukarıda m = 1 , β = 0 durumu yani popülasyondan bir elemanı seçilir ve o seçilen bir tane de rastgele seçilmiştir demektir. m = 1 , β = ∞ durumu yani popülasyondan bir eleman seçilir ve o seçilen bir tane de mecburen en iyi (*Best*) ya da aynı kendisi demektir. Aslında m burda seçim baskısı (β) gibidir. Genel olarak β'nın ayarlanması daha kolaydır ve m bunun için uygun olmayabilir, Çünkü hakkında ortada seçim ihtimali ile ilgili bir duyu (*sense*) mevcut değildir, Ama en azından β hakkında bahsi geçen $\sum_{i \in H} P_i$ = 0. 8, H: en iyilerin yarısının kümesi kuralı elde mevcuttur. *NSGA-II* algoritmasında ikili turnuva seçimi [30] yöntemi ve *NRGA* (*Non-Revisiting Genetic Algorithm*) algoritmasında (*RWS*) [31] yöntemi kullanılmıştır. Artık konu olarak çaprazlama ve mutasyon işlemlerine sıra gelmiştir.

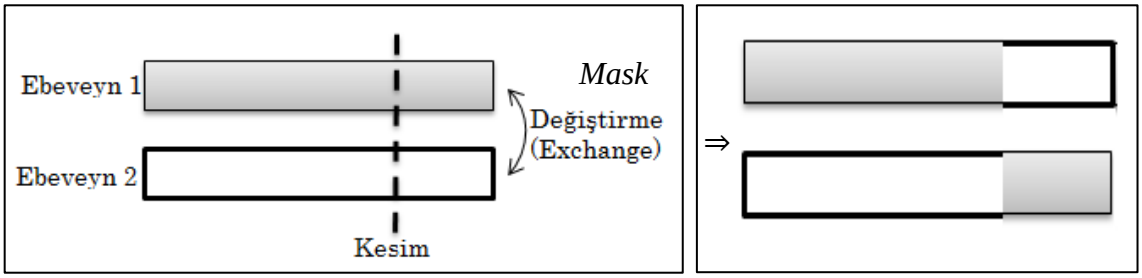
1.12. İkili (*Binary*) Problemleri

İkili bir problemin genel biçimi şöyle tanımlanmaktadır:

n_{var}: Karar değişkeni (*Decision Variable*) ve $x = (x_1, x_2, x_3, \ldots, x_{n_{var}})$ karar vektörü ya da kromozom olarak isimlendirilir. $x_i \in \{0, 1\}$ ikili sayılardır ve minimum Z = f(x) ifadesi amaç foksiyonudur. $f : \{0, 1\}^{n_{var}} \rightarrow \mathbb{R}$ kümesine aittir. Çaprazlama [30] ve mutasyonun kullanılma türü problemin türüne bağlıdır. Kromozom bir satır ya da sütun şeklinde gösterilebilir.

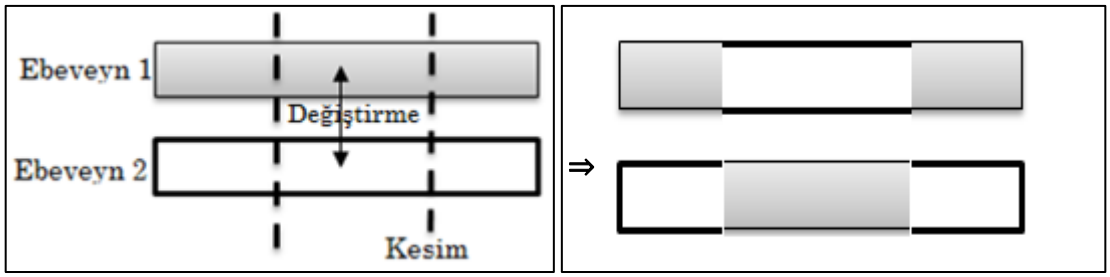
1.13. Çaprazlama (*Crossover*)

Tek Noktalı Çaprazlama (*Single Point Crossover*) Şekil 29'da gösterildiği gibi seçilen iki ebeveynin [55,30] üzerine bir kesim işlemi yapılarak kesilen kısımlar birbiriyle değiştirilir. Böylece ebeveyn ve çocukların arasında veraset ve intikal ya da kalıtım olayı gerçekleştirilmiştir.



Şekil 29. Tek noktalı çaprazlama

Kesim yerlerinin sayısı n_{var}–1 , çocukların birinci ebeveyn'den almış oldukları yerlere 1 koyulur ve ikinci ebeveyn'den almış oldukları yerlere 0 koyulur ve buna *Mask* işlemi denilmektedir. İki Noktalı Çaprazlama (*Double Point Crossover*) Şekil 30'da gösterilmiştir. Burda öncekine göre biraz daha iyi çeşitlilik (*Diversity*) sağlanmıştır.



Şekil 30. İki noktalı çaprazlama

Yani oluşturulan cevaplar tek noktalı çaprazlama [55,30,35] yöntemine göre biraz daha farklı olmuştur. Ama yinede istenilen çeşitliliğe hala ulaşılmamıştır. Düzgün Çaprazlama (*Uniform Crossover*) aşağıda şöyle tanımlanmıştır**:**

$$\text{Ebeveyn 1}: x_1 = (x_{1,1}, x_{1,2}, x_{1,3}, \ldots, x_{1,n}) \tag{34}$$

$$\text{Ebeveyn 2}: x_2 = (x_{2,1}, x_{2,2}, x_{2,3}, \ldots, x_{2,n}) \tag{35}$$

$$\text{Mask}: \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$$

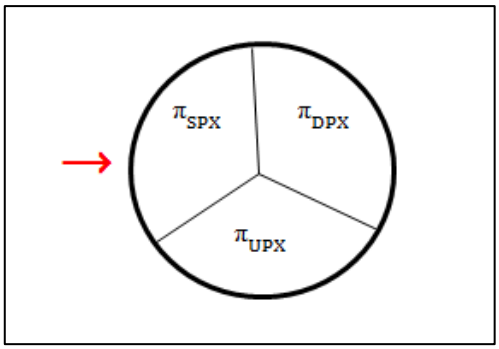
$$\text{Çocuk 1}: y_1 = (y_{1,1}, y_{1,2}, y_{1,3}, \ldots, y_{1,n}) \quad \text{Çocuk 2}: y_2 = (y_{2,1}, y_{2,2}, y_{2,3}, \ldots, y_{2,n}) \Rightarrow \begin{cases} y_{1,i} = \alpha_i x_{1,i} + (1-\alpha_i) x_{2,i} \\ y_{2,i} = \alpha_i x_{2,i} + (1-\alpha_i) x_{1,i} \end{cases} \tag{36}$$

$$\alpha_i \in \{0, 1\}$$

$$\alpha_i = 1 \rightarrow \begin{cases} y_{1,i} = x_{1,i} \\ y_{2,i} = x_{2,i} \end{cases}, \qquad \alpha_i = 0 \rightarrow \begin{cases} y_{1,i} = x_{2,i} \\ y_{2,i} = x_{1,i} \end{cases} \tag{37}$$

Eğer burda işlem ardışık sıfır ve bir'lere sınırlanmazsa, aslında hem tek noktalı çaprazlama hem de iki noktalı çaprazlama, düzgün çaprazlama [55,30,35] yönteminin birer özel durumlarından türemiştir. Düzgün çaprazlama ikili (*binary*) ve tam sayı (*integer*) uzayları için en iyi çaprazlama türü sayılır ve istenilen çeşitliliği çok iyi bir şekilde sağlayabilmektedir. Tek noktalı çaprazlama ve iki noktalı çaprazlama yöntemleri çok iyi sömürü ya da cıkarma (*exploitation*) [28] özelliklerine sahiplerdir. Düzgün çaprazlama yönteminin ise çok iyi arama (*exploration*) [28] özelliği vardır. Her optimizasyon algoritmasında bu iki husus vardır. *Exploitation* bir cevaptan mümkün olan en fazla çıkarımı elde edilmesi ve bu fikirin iyice beslenmesi demektir. E*xploration*, uzayı tamamen arama anlamına gelir. Bu iki husus arasında denge (*balance*) kurulması gerekir. Sırf *exploration* rastgele arama (*random search*) [55,30] gibidir ve bulduğu fikiri (cevabı) dallandırmadan ve beslemeden sadece aramaktadır ve bu kötü bir durumdur. Belki bulunan şeyin üzerinde durarak içinden çok iyi sonuçlar elde edilebilinir. Sırf *exploitation* yapmak da çok kötü davranıştır yani bulunan çözümün üzerinde gereğinden fazla durulmasıdır ve belki de üzerinde durulan çözüm temelden yanlıştır. Dengenin sağlanılması için şu üç çeşit çaprazlamanın bir kombinasyonu şöyle kullanılabilir. *Single Point Crossover:* (π_{SPX}), *Double Point Crossover:* (π_{DPX}) ve *Uniform Crossover*: (π_{UPX}) Şekil 31'de gösterildiği gibi rulet tekerleği seçim işlemi için kullanılabilir. Mesela bazı durumlarda 10% tek noktakı çaprazlama ve bazen 20% iki noktakı çaprazlama ve 70% düzgün çaprazlama kullanılabilir[30]. Çaprazlamaların rulet tekerleğindeki ihtimalleri π_{SPX} , π_{DPX} ve π_{UPX} , r ile gösterilmektedir.

$$\pi_{SPX} + \pi_{DPX} + \pi_{UPX} = 1 \tag{38}$$



Şekil 31. Düzgün çaprazlamada rulet tekerleği

1.14. Mutasyon (*Mutation*)

Mutasyon etkisi oranı $0 \leq \pi_m \leq 1$, mutasyon etkisi altındakilerin sayısı $\pi_m * n_{var}$ ikili (*Binary*) [30] havzasında mutasyon için sıfırlar bire ve birler sıfıra dönüşülür.

- Eğer $x_i = 0 \rightarrow x_{new} = 1$
- Eğer $x_i = 1 \rightarrow x_{new} = 0$

Yukarıdaki ifadeler bi nevi *not* operatörü ($1 - x_i \rightarrow x_i^{new}$) gibidir ve mutasyon yoğunluğunun ayarlanması gerekmektedir. $\pi_m = 1$ olursa bütün elemanların rastgele bir şekilde değiştiği ve $\pi_m = 0$ olursa mutasyon yapılmadığı anlamına gelir. $\pi_m = 0.1$ genlerin %10'nun rastgele değişmesi demektir. Mutasyon oranı düşük olduğunda etkisi azdır ve *Exploration* [28] hususu biraz sınırlanır demektir ve bu oran yüksek olursa rastgele arama gibi bir duruma dönüşür. Genelde mutasyon oranı %5~%20 aralığındadır ve bazen %1~2 daha da iyi sayılır. Değişkenlerin sayısı fazla ise mutasyon oranı, algoritmayı çalıştıranın dikkatiyle belirlenebilir.

1.15. Tam Sayılı Problemler (İnteger)

$x = (x_1, x_2, x_3, \ldots, x_{n_{var}})$ kümesinden rastgele bir eleman seçilir ve x kümesinden seçilen x_i değeri elenir [36]. Mesela seçim (1 ~ 10) aralığında yapılması var sayılsın, seçimde her sayı bir kereden fazla tekrarlanamaz ki burda permutasyon (*Permutation*) kullanılması gerekir [27]. n_{var}: değişkenlerin sayısıdır ve $x_i = \{x_{min\,1}, \ldots, x_{max}\}$ kümesindedir. Çaprazlama için bahsi geçen üç çeşit çaprazlama burda da kullanılabilir. Mutasyon : $x_i \rightarrow x_i^{new}$ şöyle $x_i^{new} \in X - \{x_i\}$ olarak tanımlanmıştır.

1.16. Sürekli Problemler

Burda önceki konularda incelenen çaprazlamalar kullanılamaz. Çünkü istenilen çeşitliliği sağlamakta yetersiz kalırlar. Hiç bir ayrık problemin sürekli bir problem [37] kadar değişkeni yoktur. Burda arama uzayı ayrık problemlerinden tamamen farklıdır. n_{var} : değişkenlerin sayısıdır ve aşağıdaki x_i kümesindedir.

$x = \left(x_1, x_2, x_3, \ldots, x_{n_{var}}\right)$, $x_i = [x_{min}, \ldots, x_{max}]$ ve x_i , $x_{min} \leq x_i \leq x_{max}$

Aritmetik çaprazlama (*Arithmetic Crossover*), düzgün çaprazlama [55,35] yöntemini biraz değiştirerek aritmetik çaprazlama [38] elde edilmiştir.

Ebeveyn 1 : $x_1 = (x_{1,1}, x_{1,2}, x_{1,3}, \ldots, x_{1,n})$ (39)

Ebeveyn 2 : $x_2 = (x_{2,1}, x_{2,2}, x_{2,3}, \ldots, x_{2,n})$ (40)

Mask : $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \rightarrow \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{1}$ **:** Sürekli

Çocuk 1 : $y_1 = (y_{1,1}, y_{1,2}, y_{1,3}, \ldots, y_{1,n})$

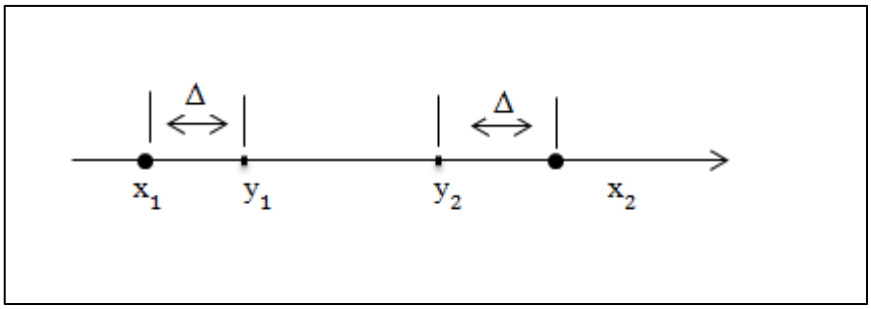
Çocuk 2 : $y_2 = (y_{2,1}, y_{2,2}, y_{2,3}, \ldots, y_{2,n})$ ⇨

$$\begin{cases} y_{1,i} = \alpha_i x_{1,i} + (1 - \alpha_i) x_{2,i} \\ y_{2,i} = (1 - \alpha_i) x_{1,i} + \alpha_i x_{2,i} \end{cases} \quad (41)$$

$y_1 = \alpha \; x_{1,i} + (1 - \alpha) x_{2,i}$, $0 \leq \alpha \leq 1$ (42)

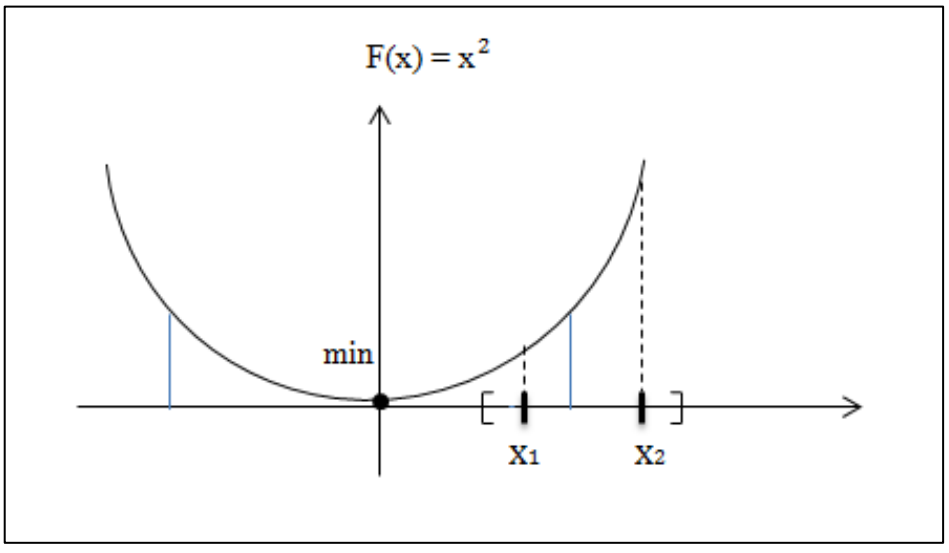
$y_2 = \alpha \; x_{2,i} + (1 - \alpha) x_{1,i}$, $x_1 \leq x_2$ (43)

$x_1 \leq y_1 \leq x_2, x_1 \leq y_2 \leq x_2, y_1 + y_2 = x_1 + x_2$ (44)



Şekil 32. Aritmetik çaprazlama

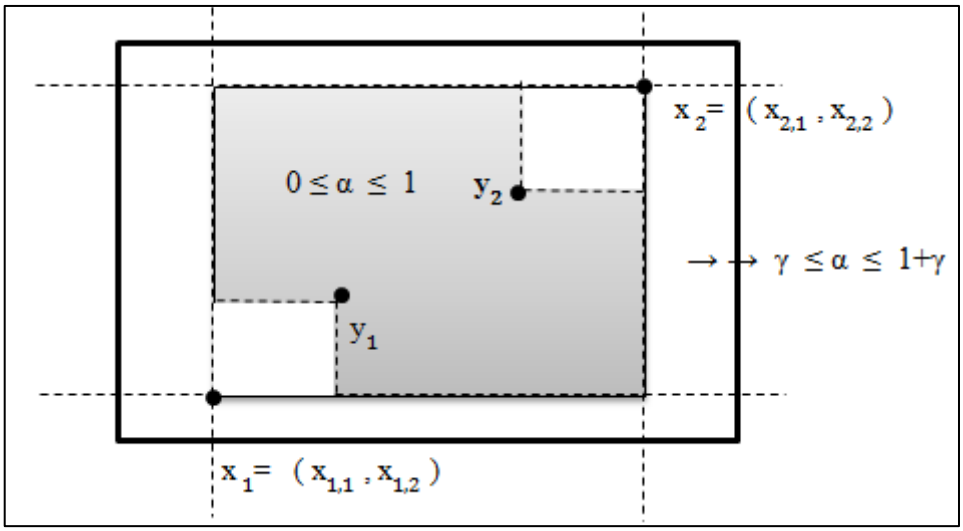
Çocuklar yani y'ler x_1 ve x_2 aralığından dışarı çıkamazlar. Yani ebeveylerine sınırlılardır ve çocuklar sonunda en fazla ebeveyn'leri gibi olabilirler. Yukarıda Şekil 32'de Δ aralıkları birbiriyle eşit durumdalar. Eğer α = 1 olursa y_1, x_1'in üstüne düşer ve y_2 de, x_2 'nin üstüne düşer. Eğer α = 0. 5 olursa her ikisi tam ortaya yani x_1 ve x_2 aralığının ortalamasının arasına düşerler. Anlatılan durumun dezavantajları vardır. Şekil 33'de çocuklar sonunda en fazla ebeveyn'leri kadar olabilirler, dolayısıyla diğer bölgeleri keşfetme olanağı yoktur ve buna göre $0 \leq \alpha \leq 1$ olması işe gelmez ve α bu aralığın dışına da çıkabilmelidir.



Şekil 33. $0 \leq \alpha \leq 1$ aralığının dışında bir durum

Bunun için bir γ Şekil 34'te gösterildiği gibi şöyle tanımlanır $-\gamma \leq \alpha \leq 1+\gamma$, Örneğin : $\gamma = 0.1 \rightarrow -0.1 \leq \alpha \leq 1.1$ olarak tanımlanmıştır. Şans eseri eğer sınırlandığı bölge yani ebeveyn'lerin aralığında çocuk üretilirse ozaman iyi çaprazlama

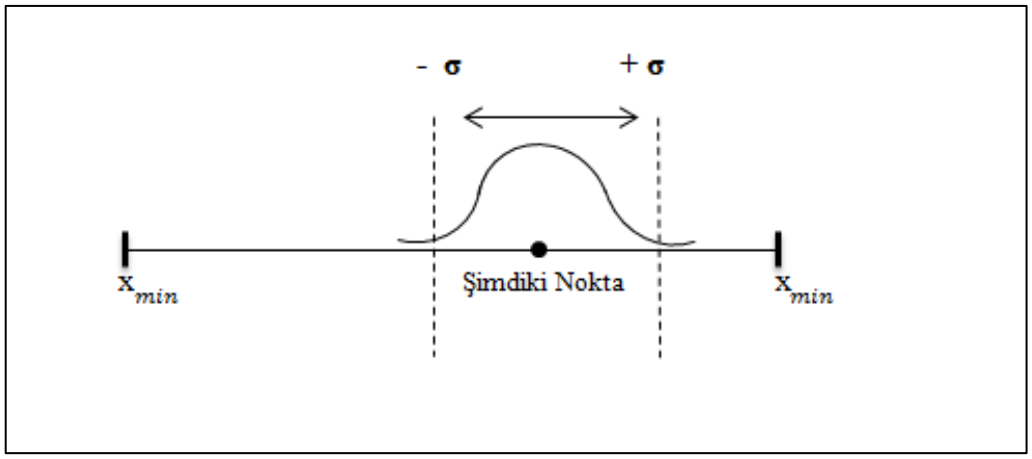
yapıldığı söylenebilir. Aritmetik çaprazlama [55,39] iki boyutlu uzayda üstten bakılırsa aşağıdaki gibi gösterilebilir:



Şekil 34. İki boyutlu alanda aritmetik çaprazlama

1.16.1. Sürekli Uzayda Mutasyon

Eğer dağılım düzgün ise x_{min} , x_{max} aralığında her sayı seçilebilir, Ancak biraz işlem yönlü olması istenilirse Şekil 35'de gösterildiği gibi Normal Dağılım [30] ya da Kuşi Dağılımı (*Cauchy Distribution*) [55] kullanılması gerekir.



Şekil 35. Normal dağılım

$$x_i^{new} \sim p(X),\ X \in [x_{min}, x_{max}] \tag{45}$$

$$x_i^{new} \sim N(x_i, \sigma^2) \sim x_i + \sigma N(0,1) \tag{46}$$

$$\sigma = \mu(x_{max} - x_{min}),\ \mu = 0.1 \tag{47}$$

Yukarıda N, normal dağılımıdır. x_i ortalamadır ve σ bir standart sapması olarak adım uzunluğunu değiştirir. Adım uzunluğu yüksek olursa *Exploration'*nin fazla olmasına neden olacaktır. Sigma σ düşük olursa *Exploitation* yükselir ve *Exploration* azalır. Adım uzunluğu ne büyük nede küçük olması gerekir ve bunun için *Schwefel'in* kuralı kullanılmaktadır.

1.17. Schwefel'in $\frac{1}{5}$ kuralı

Schwefel [34,33] kuralinda eğer başarılı mutasyonların oranı 1/5 'ten fazla olursa adım uzunluğu büyütülür ve eğer 1/5'ten az olursa adım uzunluğu küçültülür. Amaç fonksiyonun iyileşmesine neden olan an'a yani başarılı mutasyonu yüzdesi 20 % olursa bu durum hızlı ve küçük adımlarla yerel optimum'a doğru gidildiğini gösterir ve bundan kaçınılması gerekir. Eğer algoritma gereğinden fazla hızlı iyileşerek işin gidişindeyse ozaman bu durumdan merak ve endişe duyulur ki buna erken yakınsama (*Premature Convergence*) [33] denilir. Eğer ki algoritma ufak ve küçük adımlarla iyileşirse, demek algoritma adım uzunluğunun büyütülmesini gerektiren bir yolda ilerlemektedir.

1.17.1. Çok Amaçlı Optimizasyon

Çok amaçlı optimizasyon [42] konusu çeşitli isimlerle tanınır ve *multi objective optimization* adıyla daha ünlüdür. *Multi criteria* ve *vector optimization* isimleri ile özellikle uygulamalı matematikciler arasında yapay zekadan ziyade işin daha çok matematik boyutuyla tanınır. Bu konunun felsefesi amaç fonksiyonu artık skalar olmamasıdır yani amaç fonksiyonu vektör [41] olarak karşımıza çıkar. Amaç şurda tüm amaç fonksiyonları aynı anda optimize etmektir dolayısıyla ibareyi daha doğru çeviri yaparsak vektör

optimizasyonu olarak çok amaçlı optimizasyonuyla denk gelir. Optimizasyon problemi genel ifadeyle şöyle aşağıdaki gibi tanımlanır:

$Min\ f(x)\ F: X \rightarrow R$: Amaç fonksiyonu (48)

$g(x) = \geq 0\ \forall_j \epsilon\ j$: Kısıt (49)

$h(x) = 0\ \forall_k \epsilon\ K\ x \epsilon\ X$: Kısıt (50)

Optimizasyon yukarıdaki şekilde yazılırsa bir minimizasyon tekniğinin genel ifadesi olur ki içinde kısıtlar (*constraint*) [40] ve amaç fonksiyonunu içerir. Çok amaçlı optimizasyon tekniği ile yukarıdaki ifade arasındaki fark şöyledir:

$Min\ f(x)\ F: X \rightarrow R$: Skalar amaç fonksiyonu (51)

$F: X \rightarrow R^m$ (52)

$F = (f_1\ ,\ f_2\ , \ldots f_m)$ (53)

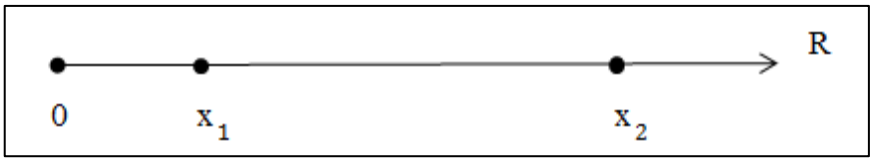
Yukarıdaki fonksiyonun değeri artık bir vektör şeklindedir. Asıl sorun R^m'nin vektör olmasıdır dolayısıyla R kümesi gibi bir sıralanabilen küme değildir.

$Min\ f_i(x)\ \forall_i \in I$ (54)

$F: X \rightarrow R^m$ (55)

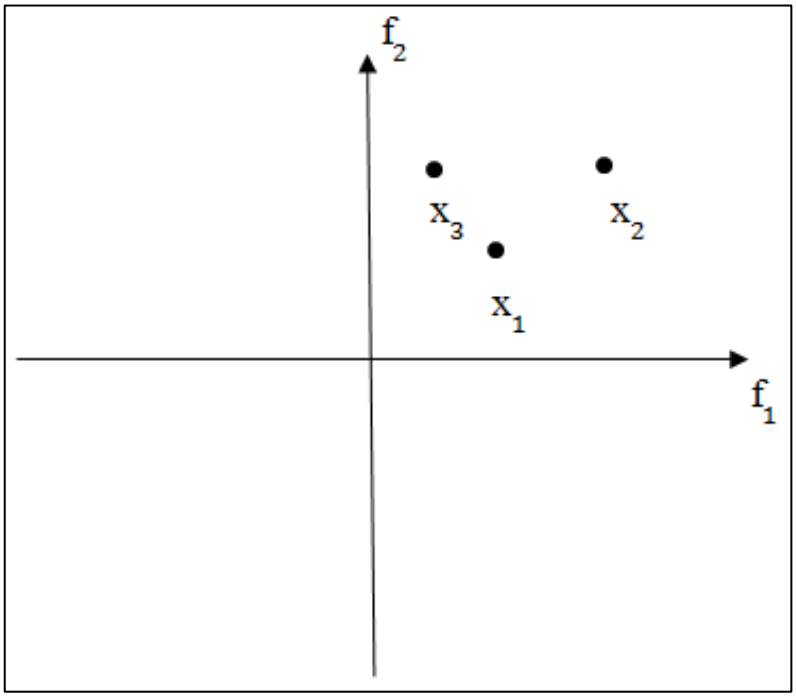
$I = \{1,2,3, \ldots m\}$ (56)

Problem genel olarak yukarıda tanımlandı. Çok amaçlı optimizasyondaki esas sorun $F:X \rightarrow R^m$ [42] sıralanabilmezliğidir. R^m kümesi büyükten küçüğe ya da küçükten büyüğe sıralanamaz yani $\leq \geq > <$ gibi operatörler bu kümenin içinde tanımlanamaz. Konunun daha iyi anlaşılması açısından Şekil 36'daki gibi gerçel sayılar eksenini göz önüne alalım.



Şekil 36. Gerçel sayıların ekseni

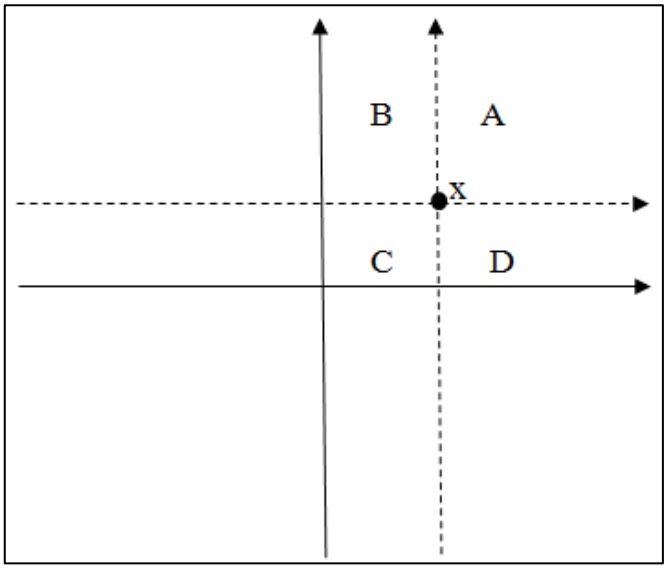
Yukarıdaki eksende x_1 ve x_2 arasındaki ilişki $\leq \geq > <$ operatörler [42] ile belirlenebilir. Bu sebeple *R* kümesi sıralanabilir dir yani $x_1 < x_2$ ifadesinin anlamı vardır. Öte yandan uzayı iki boyutlu bir düzlem düşünürsek farklı senaryolar karşımıza çıkar. Şunu vurgulamakta fayda var, uzaydan bahsederken amaç fonksiyonun uzayı kastedilmektedir. Bu genotip uzayı ile karıştırılmamalıdır. Genetik algoritmada amaç fonksiyonun uzayı *fenotip* (amaç fonksiyonların değerleri) ve değişkenlerin değerleri *jenotip* olarak adlandırılır. Bundan sonra bahsedeceğimiz konular hep *fenotip* ile ilgili olacaktır. Aşağıdaki şekilde gösterilen iki boyutlu bir uzayı göz önüne alalım:



Şekil 37. İki boyutlu fenotip uzayı

Eger x_1 ve x_2 arasında kıyaslama yaparsak x_1 hem yatay hemde düşey bileşenler açısından x_2'den küçüktür yani $x_1 < x_2$ olduğunu söyleyebiliriz. Ancak x_1 ile x_3 arasında kıyaslama yaparken sorun ortaya çıkmaktadır. Yatay eksenin bileşenini göz önüne

alırsak x_3 küçüktür ancak düşey eksen açısından x_1 küçüktür. Sonuçta bu iki nokta birbirne göre tam olarak ne küçük nede büyüktür yani mutlak olarak ikisinden birini büyük ydada küçük ifade edemeyiz. Buna sıralanamayan durum denilir. Aşağıdaki şekilde bir x noktasını göz önüne alalım. Şekil 38'deki alanı *A, B, C, D* ile isimlendirilen dört bölgeye bölelim.



Şekil 38. *A, B, C, D* ile dört bölgeye bölünen alan

Şekil 38'e göre aşağıdaki ifadeler yazılabilir.

- $a \in A \rightarrow a > x$
- $c \in C \rightarrow c < x$

Sorun *B* ve *D* bölgelerinin elemanlarındadır çünkü x ile kıyas yaparak bu bölgelere ilişkin bir yorum yapılamaz. Sonuçta *A* ve *C* bölgeleri için sıralama yapılabilir ancak *B* ve *D* için bu mümkün olmaz. Şimdi bir genetik algoritma geliştirdiğimizi düşünelim, genetik algoritmanın aşamalarından biri de sıralama (*sorting*) [43]. yani amaç fonksiyonu değerleri üzerinden popülasyon elemanlarını sıralama işlemidir. *Sorting* operatörü işlemi bahsi geçen sıralanamayan durumlarda skalar problemlerdeki gibi kullanılamaz. Bu noktada birtakım şeyler değişmelidir. Her *C* çeyreğin elemanı *x*'ten iyidir ve her *A* çeyreğin elemanı *x*'ten kötüdür denilebilir fakat *B* ve *D* çeyreklerinde durum kayıtsız (*indifferent*)'dır ve bu bölge elemanlarının büyüklük küçüklük açısından birbirine göre üstünlükleri yoktur. Buradan baskın (*dominance*) [42,55] kavramı ortaya çıkar.

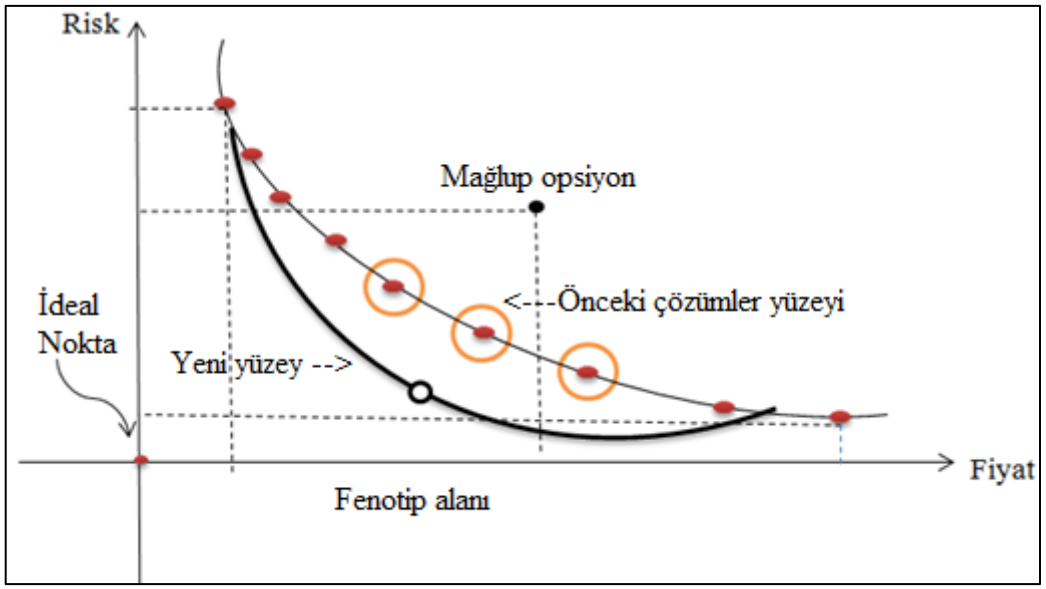
Çok amaçlı yöntemler ile optimizasyon problemlerini çözerken programın tek sefer koşmasında tüm cevapları bulmak mümkün olabilmektedir yani aynı anda tüm amaç fonksiyonları maksimize yada minimize edilebilir. Bu tekniğin yanısıra bazı *decomposition* yöntemleri ile amaç fonksiyonun uzayını sıralanabilir bir skalar uzayına dönüştürmek mümkündür.

$Min\ f_i(x)\ \forall_i i\ ,\ F: X \rightarrow R^m$: Vektör uzayı (57)

Yukarıdaki ifadenin yerine aşağıdaki ifadeyi kullanalım.

$Min\ f_D(x)\ ,\ F_D: X \rightarrow R$: $\mathrm{F}_{dekompozisyon}$: Skalar (58)

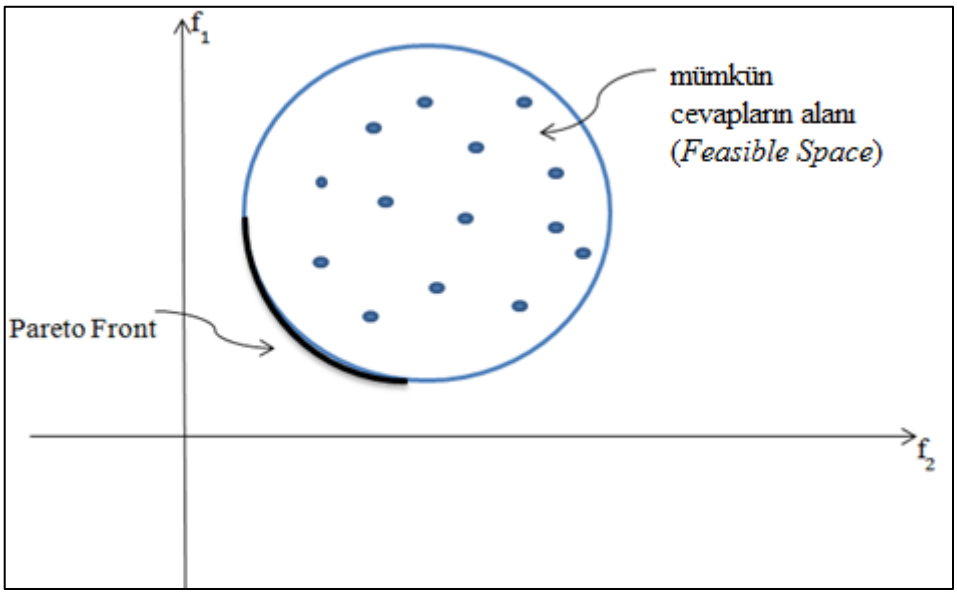
x cevap olursa gerçekten *C* mümkün *(feasible*) bölgesinde x'ten daha iyi cevap bulunmaz. Bu tür cevaplar deyim olarak bir *non-dominated* [42] cevabı ibaresiyle isimlenir. Bu cevap düzlem uzayında ondan başka daha iyi cevap iki eksen bileşeni açısından bulunamazdır. Bir dekompozisyon problemi tasarımı kolay değildir. *Non dominated* cevabı bulmak dekompozisyon yöntemiyle bir çok amaçlı optimizasyon problemi için çeşiti senaryolardan geçer. Şöyle bir durum vardır ki çok amaçlı problemini tanımladıktan sonra işin sonunda problemin bir cevaptan başka cevap olmadığı sonucuna varılır. Bu bilimsel olarak yanlış değil ve problemi baştan beri çok amaçlı çözmemeliydik anlamına gelmez. Bir çok amaçlı optimizasyon probleminin mutlaka birden fazla cevabı olması gerekmez [42,55]. Ancak çok amaçlı optimizasyon probleminden genel olarak beklentimiz birden fazla senaryosu olmasıdır. Birden fazla senaryoya neden ihtiyaç duyulduğunu anlamak için, bir araba satın alacağımızı düşünelim. Bizim için arabanın kullanma riskleri ve fiyatlar önemlidir.



Şekil 39. Arabanın risk ve fiyat grafiği

Şekil 39'a göre birtakım arabalar çok güvenliler ama fiyatları çok fazladır. Bedava ve sıfır riskli araba önerisi bir ideal senaryodur. Bu ideal noktaya hiçbir zaman ulaşılamaz. Şekil 39'daki grafiğe göre fiyat biraz düşer ama bu da riski biraz yükseltir. Bu şekilde devam edilirse birtakım senaryolar karşımıza çıkar. Bu senaryolara üzerine düşünüldüğünde bir yüksek fiyat ve yüksek risk senaryosu hiç düşünülmeden elenir. Buna mağlup (*dominated*) [43,55] çözüm denilir. Aranılan ise en önemli olan çözüm yüzeyleridir. Bu çözüm yüzeylerinin sürekli olması gerekliliği yoktur. Ayrık'ta olabilirler. Yeni araba teknolojisi meydana gelmediği müddedce senaryolar yüzeyinin ideal noktaya dönük tarafında başka çözüm olmadığından emin olunabilinir. Eğer bir araba fabrikası Şekil 39'da içi boş küçük siyah yuvarlakla gösterildiği gibi bir opsiyon sunarsa önceki opsiyonlar senaryolar yani Şekil 39'da üç tane yuvarlakla belirtilmiş olanlar devreden çıkar yada elenir. Artık önceki çözümler yüzeyi şekildeki gibi yeni koyu siyah gösterilen yüzeye dönüşür. Bu çözüm yüzeyleri yada senaryolar yüzeyine *pareto front* denilir [42,55]. Bu çözüm yüzeyi içinde senaryolar üzerinde düşünmeye değer opsiyonlar ve cevaplar mevcuttur. Bunlardan bütçe ve kabullenilebilecek risk'e göre bir opsiyon seçilir. Aslında aranılan mağlup olmayan (*non dominated*) çözümdür. Çok amaçlı bir optimizasyon problemini çözmek demek *non dominated* çözümleri yada *pareto front*'u bulmak demektir. Başka bir deyişle mümkün cevapların uzayındaki (*feasible)* [42] cevapların arasında bulduğumuz *pareto front*'u mağlup edecek daha başka bir cevap bulunmamasıdır. Var

sayılsın ki bir çok amaçlı problem için mümkün cevapları bulundu ve cevaplar şekilde gösterilen yuvarlak bölgede mümkün (*feasible*) bulunmaktalar.



Şekil 40. Fenotip alanında mümkün (*feasible*) uzayı

Non Dominated Pareto yukarıdaki Şekil 40'da koyu siyah ile belirtilmiştir ve aranılan *pareto*'dur ama tüm *feasible* uzayını elde etmek ve onlardan bir takımı seçmek çok zaman alır. Bu sebeple kısayollara başvurulmalıdır. Bunun için iki yola başvurulur.

- Problemi doğrudan çok amaçlı yöntemlerle çözen yöntemler
- Çok amaçlı problemini tek amaçlı problemine dönüştürerek çözen yöntemler

Çok amaçlı problemi tek amaçlı probleme dönüştürerek çözenler çoğunlukla klasik matematik alanında incelenmektedir. Bu yöntemler farklı ayarlar (*setting*) [42] ile koşturulup kullanılır ve sonunda bir *pareto front* tahmini elde edilir.

1.17.2. Çok Amaçlı Problemler ile Baş Etme Yöntemleri

Çok amaçlı problemler ile baş etme yöntemleri problemi dekompozisyon ve doğrudan çözen yöntemler ile kategorize edilmektedir.

1.17.2.1. Dekompozisyonun Çeşitli Yöntemleri

Aşağıda bahsedilen dekompozisyon yöntemleri incelendiğinde bunların ne kadar karmaşık olduğu anlaşılacaktır oysa çok amaçlı algoritmalar incelendiğinde bunların daha az karmaşıklıkla problemi etkin bir şekilde çözebildikleri anlaşılacaktır.

- Ağırlıklı Toplam (*Weighted Sum*)
- Hedef Programlama (*Goal Programming*)
- Hedefe Ulaşmalı (*Goal Attainment*)
- €- Kısıtlama (€-*Constraint*)

Dekompozisyon [44,55] yöntemlerinin işlem yükü fazlalığının sebebini anlamak için iki amaç fonksiyon olduğunu varsayalım. Bu amaç fonksiyonlar tek bir sayıya dönüştürülemez çünkü bir sayının bilgi yükü gerçekten bir sayı kadardır. Yani bir sayı ile iki sayı ifade edilemez. Amaç fonksiyonlarından elde edilen sayıları saklamak için tek boyutlu skalar uzayına eşleme (*mapping*) yapılırsa bilgi kaybı meydana gelir. Bundan dolayı bütün dekompozisyon yöntemlerinde birçok kere koşturularak bir *pareto front* elde edilmiştir. Örneğin: Ağırlıklı toplam (*weighted sum*) [45,55] dekompozisyon tekniği birçok kez çeşitli parametreler ve ayarlarlar (*setting*) ile koşturularak *pareto front* elde edilmiştir. Ama doğrudan çözüm yöntemleri bir kere koşturularak *pareto front*'un tümünü bulmaktadırlar.

1.17.3. Ağırlıklı Toplam Yöntemi

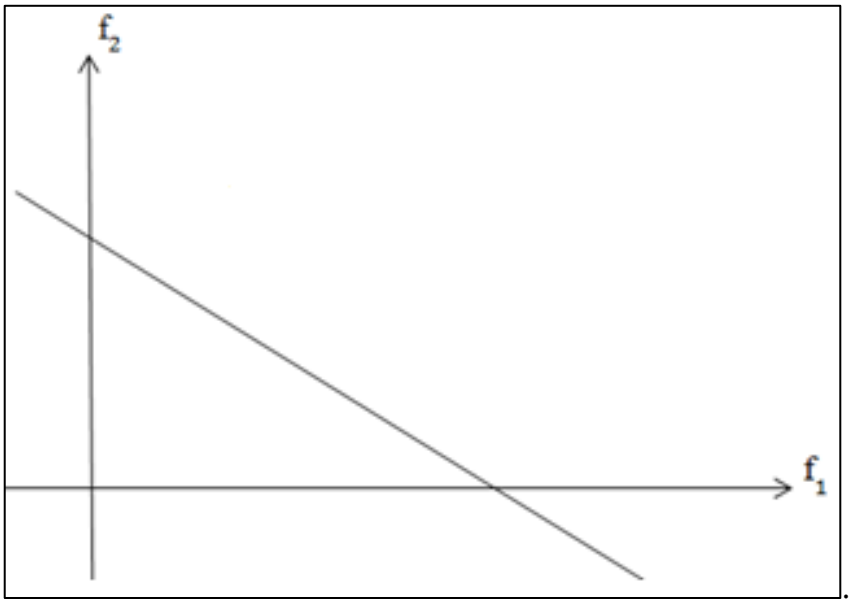
Çok amaçlı problem aşağıda ifade edilmiştir.

$$Min\ f_i(x)\ \forall_i\ i \in \{1,2,\dots,m\} \tag{59}$$

Bunu aşağıdaki gibi dönüştürerek yazmak mümkündür.

$$Min\ f_{Ws}(x) = \sum_{i=1}^{m} W_i\ f_i(x)\ W_i \geq 0 \tag{60}$$

M tane amaç mevcut ve bunların ağırlıklı toplamaları bir faktör olarak göz önünde bulundurulur. Bu toplamalar nekadar az olursalar bileşenleri bir o kadar azalır. Ağırlık [45] parametresi bileşenlerden her birinin önemini ve değerini ayarlamak içindir. Eğer $W = 0$ olursa amaç fonksiyonu $f_i(x) = 0$ olur ve artık $f_i(x)$ elenir. Konuyu daha iyi anlamak için aşağıdaki Şekil 41'i göz önüne alalım.

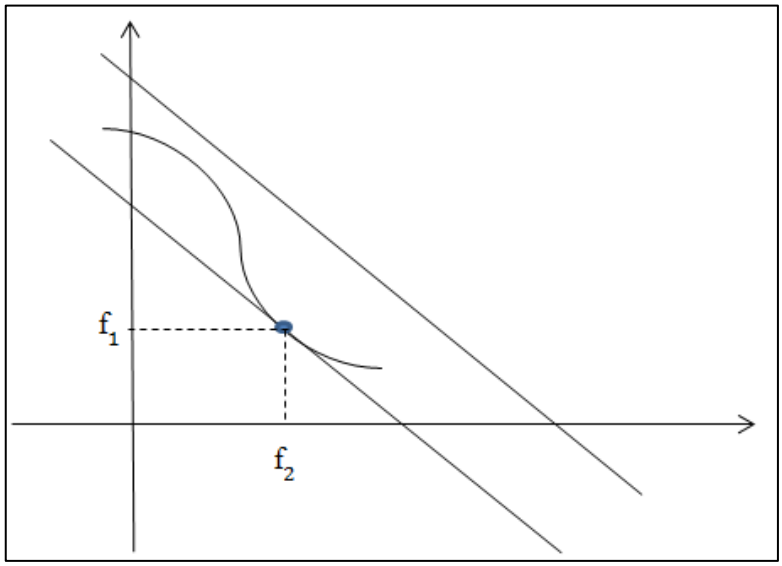


Şekil 41. Z, doğrusal denklemin eğirişi

$$\text{Min}\ W_1 F_1 + W_2 F_2 \ : W_1, W_2 \geq 0 \tag{61}$$

$$Z = W_1 F_1 + W_2 F_2 : \text{Doğrusal denklemin biçimi} \tag{62}$$

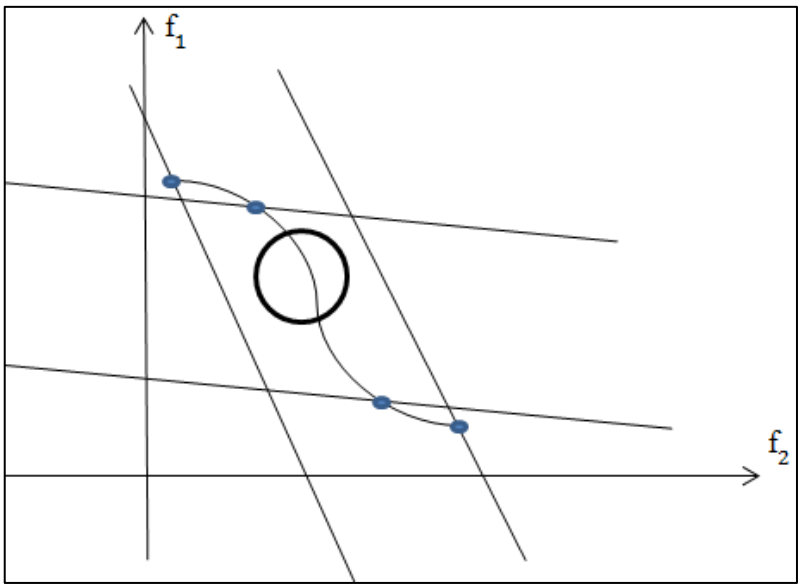
Bir iki boyutlu alan var sayılsın yani F_1ve F_2 amaç fonksiyonları minimize edilmesi isteniyor olsun. F_1 ve F_2 amaç fonksiyonları aşağıdaki Şekil 42'deki gibi minimize edilebilir. Ayrıca aşağıdaki şekilde iki doğru arasında bir *pareto* yüzeyi olduğunu varsayalım.



Şekil 42. Pareto yüzeyi ile kesişim noktası

Pareto cephesinin noktalarını bulmak için denklem 62'daki sabit Z değerini mümkün olan en minimum değere ulaştırması isteniyor. Problemin mümkün olan tüm cevapları *feasible* uzayında yani amaç fonksiyonların aralığında olduğu biliniyordur. Z değerlerinin ağırlıklı toplamlarını minimum etmekte aslında birinci doğruya göre bir paralel doğru bulunmasıdır. Sonunda *feasible* uzayı ile ideal noktaya bakan tarafında bir kesişim noktası elde edilecektir. Bu en sonunda *feasible* uzayı ile kesişim noktasını oluşturacaktır ve en az Z değerinin oynatılmasıyla bulunmuştur. Z'yi azaltarak *feasible* ile kesişim noktasından daha fazla ideal noktaya doğru ilerlenemez. Bulduğumuz noktanın f_1 ve f_2 bileşenleri vardır ve gerçekten mümkün olan en minimum değerlere sahiplerdir. Aslında anlatılan yöntem ile sadece *pareto*'nun bir noktası keşfedilmiştir. Bu yöntem farklı doğrular için denenebilir. Denklem 61'in ağrırlık katsayıları [45] olan w_1 ve w_2 değerleri değiştirilerek doğrunun eğimi ve eksenleri kestiği yerler değiştirilebilir. Dolayısı ile z değerinin değişmesi ile y kesişimi ve $w_1$ve w_2 değerlerinin değişmesi ile doğrunun eğimi değişir. En az y kesişimi ve çeşitli w_1, w_2'ler ile farklı başka doğrular çizerek *pareto*'nun farkı farklı

noktaları bulunur. Ağırlıklı toplam yöntemi *pareto*'nun içbükey (*concave*) [42,55] kısımlarını bulamaz. Bu durum Şekil 43'de içi boş yuvarlak ile gösterilmiştir. *Pareto*'nun (*concave*) olan kısımları ideal noktaya daha yakındır ancak ağırlıklı toplam yöntemi doğrusal yaklaşımı nedeniyle yetersiz kalmaktadır. Bu dezavantajına rağmen basit olmasından ötürü *vega* (*vector evaluated genetic algorithm*) [46,47,55] gibi birçok algoritmalar bu yönteme dayalı çalışmaktadır. Bir diğer sorun doğru eğimini değiştirmektir. Her doğru teğetlemesi ile pareto'dan sadece bir nokta bulunur. Eğer on nokta bulunmak istenirse on tane farklı konfigürasyon ayarlanmak zorunda kalınır.



Şekil 43. Pareto'nun içbükey kısmı

1.17.4. Hedef Programlama (*Goal Programming*) Yöntemi

Bu yöntem ideal noktadan olan uzaklığı minimize etmektedir. Bunun için bir ideal (*target,goal*) [48,49] nokta ihtiyacı vardır. Bu noktayı tanımlamak üzere her amaç fonksiyonu için aşağıdaki ifade çözülür.

$$\text{Min } f_i(x) \rightarrow t_i \tag{63}$$

Denk gelen her f_i için bir hedef tanımlanır. Yani amaç fonksiyonu değeri $f_i(x)$ idealize değeri anlamına gelir. Şimdi problemden önbilgi edinilebilir ve ona göre tahminde bulunabilinir. Her amaç fonksiyonu ayrı ayrı başkalarının yokluğunda optimize edilir ve

elde edilen sonuç hedef değeri olarak alınır. Genelde amaç fonksiyonlarının, birbirinin üzerine bozucu etkileri vardır. Dolayısıyla bir amaç fonksiyonunu optimize ederken diğerlerini hesaba katmayarak yani etkilerini ortadan kaldırarak her amaç fonksiyonu ancak kendi optimum değerine ulaşmış olur ki diğerlerin var oluşunda bu optimum değere ulaşamazdı. Mesela araba grafiği örneğinde en ucuz araba opsiyonu sıfır ve en az risk'de sıfırdır fakat her ikisi birlikte sıfır olamazlar. Ancak eğer tek bir eksen üzerinde minimizasyon yapılması düşünülürse idealize nokta bulunabilir. Bazı durumlarda bu yöntem yetersiz kalır ve problem'den daha fazla önbilgiye ihtiyaç duyulur. Bu durumlar uyarlamalı (*adaptive*) olarak *MOEA/D* (*Multiobjective Evolutionary Algorithm Based on Decomposition*) [44,47] algoritmasında olduğu gibi hedef, problemin durumuna bağlı hesaplanır.

$$\text{Min } f_i(x) \rightarrow t_i \; t = (t_1, t_2, \ldots, t_m) \tag{64}$$

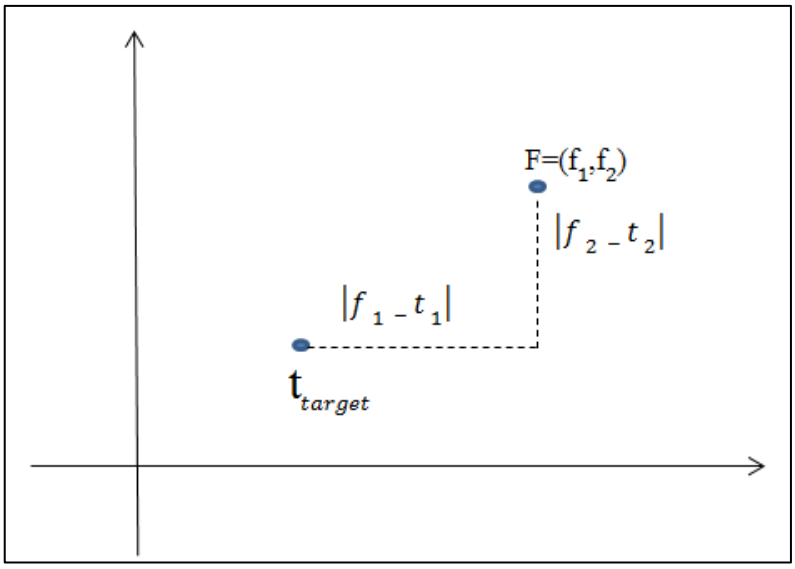
Yukarıda t kümesi idealize noktaları içerir. Uzaklık aşağıdaki gibi hesaplanıp en küçük değeri bulunur.

$$f_g(x) = \text{Min} \sum_{i=1}^{m} \left| f_i(x) \_ t_i \right| \tag{65}$$

$f_g(x)$ bir skalar fonksiyondur. Yukarıdaki ifadededе mesafe toplamalarından hedef değeri elde edilir. Öte yandan amaç fonksiyonu uzayında ölçekler değişirse sorun yaşanabilir. Araba örneğinde fiyatlar lirayla söylenirse 50 milyon $5*10^8$ ölçeğinde olur ve risk 0 ve 100 arasında yüzde olarak söylenebilir. Risk en fazla 100 olmaktadır ve 10^8 yanında göz ardı edilecek kadar küçüktür. Dolayısıyla *goal programming* [48,49] yönteminde $\sum$ toplamaları bu şekilde sonuçlanamaz. Bundan dolayı normalize işlemine yada başka yöntemlere başvurulmalıdır.

$$f_g(x) = \text{Min} \sum_{i=1}^{m} \frac{\left| f_i(x) \_ t_i \right|}{\left| f_i Max \_ t_i \right|} \tag{66}$$

Yukarıdaki denklemde W_i ağırlıkları ile normalize işlemi gerçekleştirilmiştir.



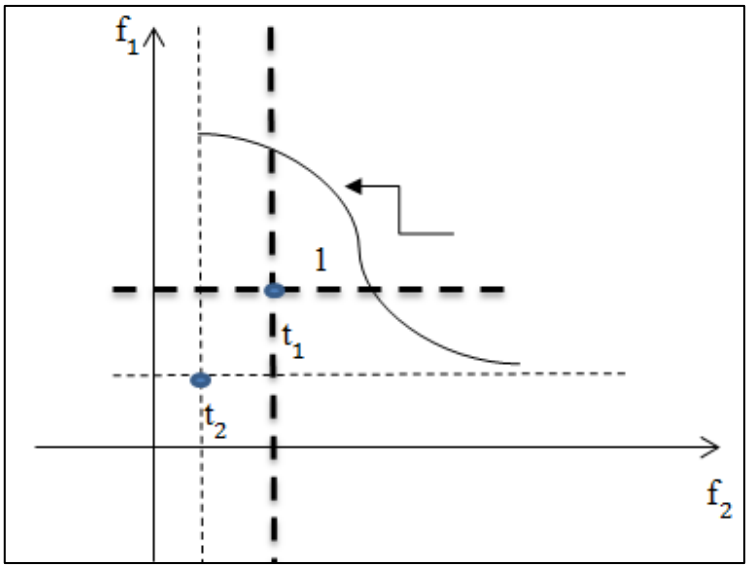
Şekil 44. Target noktasından olan uzaklık

Şekil 44'deki uzaklık hesaplanırken norm [55,49] aşağıdaki 67 ve 68 ifadeleri gibi kullanılır.

$$Min \sqrt{\sum_{i=1}^{m} w_i |f_i(x) - t_i|^2} : \textit{Euclidean Norm 2} \quad (67)$$

$$Min \sqrt{\sum_{i=1}^{m} w_i |f_i(x) - t_i|} : \textit{City Block Distance Norm 1} \quad (68)$$

Goal programming yönteminde norm 1 kullanılır. Bu yöntemdeki sorun hedef (*Goal*) [49] noktasının belirlenmesidir. Bazı durumlarda *target*'in tahmin edilmesi zor olmaktadır.



Şekil 45. Target'in belirtilmesi

Şekil 45'de gösterildiği gibi t_1 tahmin edilmesi ile sadece 1 ile işaretlenmiş alan bulunmuş olunur. Oysa t'nin biraz geriye kaymasıyla yani t_2 gibi bir nokta ile tüm *pareto* alanı keşif kapsamıda bulunabilir. Bu açıdan target'in tayin edilmesi oldukca önemlidir. Bazı durumlarda *feasible* bir ayrık uzayı şeklindedir ve cevap bulunmamaktadır. Eğer amaç fonksiyonu doğrusal ise ya da doğrusallaştırma yöntemleri ile iyi tahmin sağlanırsa bu yöntemler yüksek başarım sergilemektedirler.

1.17.5. Hedefe Ulaşmalı (*Goal Attainment*) Yöntemi

Bu yöntem hedef programlama yöntemine oldukça benzer yalnız matematik tanımlamasında [50] ufak farklılığı vardır.

$$Min\, f_i(x) \Rightarrow Min\, d, W_i \left(f_i(x) \_ t_i\right) \leq d \tag{69}$$

Yukarıdaki denklemde sol kısmıdaki problemin ifadesi sağdaki ifadelere denk gelir. Çeşitli i'ler için bu sınırlamayı (*constraint*) [50] açarsak aşağıdaki gibi olur.

$$W_1\left(f_1(x) \_ t_1\right) \leq d$$
$$W_2\left(f_2(x) \_ t_2\right) \leq d$$
.
.
.
$$W_m\left(f_m(x) \_ t_m\right) \leq d$$

d soldaki tüm ifadelerden büyüktür. Aranılan en küçük d değeridir. Elde edilen d öyle olmalıdır ki hem soldaki ifadelerden büyük hem de olabilecek en küçük değerde olsun. Bunun için d'yi onların maksimum değerleri ile eşleştiriyoruz. Başka bir ifadeyle aşağıdaki gibi olur:

$$Min\, d = Max\, W_i(f_i \_ t_i) \Rightarrow Min\, Max_i\, W_i(f_i \_ t_i) \tag{70}$$

Böylece bir min max problemi [50] tanımlanmış olmuştur. Önceki konudaki gibi $\left|f_1(x) \_ \mathrm{t}_1\right| \geq 0$, f_1 mümkün olan en küçük değerinde çıkması istenmiştir. Ayrıca t_1 ideal bir noktadır ve bu noktaya ulaşmak mümkün değildir. Bu sebeple hedef gerçekten istenilen değer ise $f_1(x) \_ \mathrm{t}_1 \geq 0$ olması gerekir. Bu vektörün çeşitli norm'ları bizim için iki nokta arasındaki uzaklığı bulmakta farklı kriterleri ifade edebilir. *Norm* 1, *Norm* 2, *Norm* ∞ [55,49,50] gibi Denklem 71 normun genel tanımıdır. Denklem 72 şehir çapı norm (*City block distance*) tanımıdır. Denklem 73 (*Euclidean Norm*) tanımıdır. Denklem 74 sonsuz norm tanımıdır.

$x \in R^m$

$$\|x\|_p = \sqrt[p]{\sum_{i=1}^{n} |x_i|^p} \tag{71}$$

$$\|x\|_1 = |x_1| + |x_2| + , \ldots , |x_n| \tag{72}$$

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2, \ldots, x_n^2} \tag{73}$$

$\|x\|_{\infty}$= Max $|x_i|$ (74)

Şimdiki cevaplar ve hedef arasındaki mesafe vektörü aşağıdaki gibidir.

$f - t = (f_1 - t_1, f_2 - t_2, \ldots, f_m - t_m)$ (75)

Hedef programlamada (Ağırlıklı Norm) olarak aşagıdaki tanım kullanılmaktadır.

$$Min \|f - t\|_{1.W} = \sum_{i=1}^{m} W_i |f_i - t_i| \quad (76)$$

İkinci mertebe Hedef programlamada aşağıdaki tanım kullanılmaktar.

$$Min \|f - t\|_{2.W} = \sqrt{\sum_{i=1}^{m} W_i |f_i - t_i|^2} \quad (77)$$

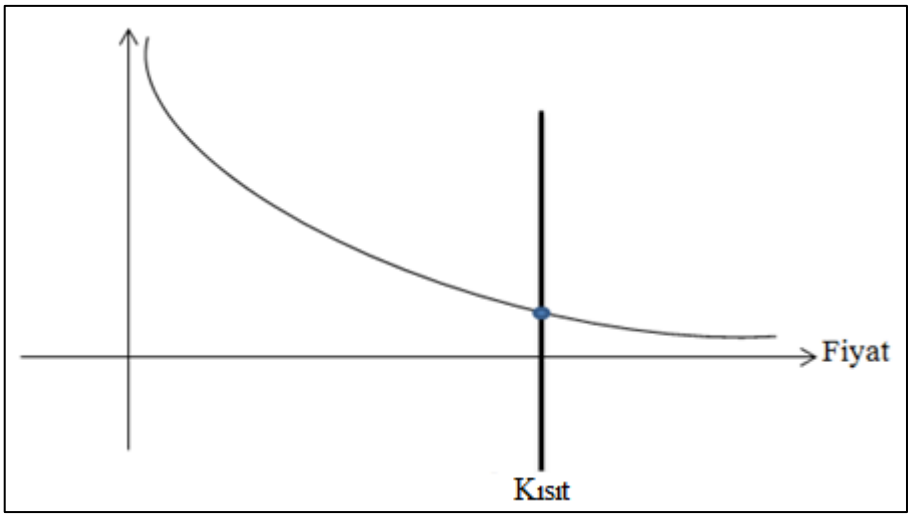
Hedefe ulaşmalı yöntemi içinde aşağıdai tanım kullanılmaktadır.

$$Min \|f - t\|_{\infty,W} = Max\, W_i |f_i - t_i| \quad (78)$$

Bir norm mertebesi değişimi ile *goal programming* üzerinden *goal attainment* yöntemi ortaya çıkmiş oldu. Buna *chebyshev* norm'u yada *chebyshev decomposition*'u da denilmektedir.

1.17.6. €- Kısıtlama (€-*Constraint*) Yöntemi

Bu yöntem amaç fonksiyonu uzayının karmaşıklığını azaltıp yerine bunu kısıtlar (*constraint*)'ların uzayına ilave etmiştir. Arabanın fiyat ve risk grafiği örneğine dönersek, araba satın almak için ayrılan bütçe mesela 10000-15000 TL fiyatlar üzerine konulan bir çeşit kısıtlamadır.



Şekil 46. Arabanın fiyat ve risk grafiğinde kısıt uygulaması

Belirlenmiş fiyata göre en iyi araba opsyonunu bulunabilir. Eğer fiyatın kısıttan fazla olmaması ve en az risk opsyonu istenilirse Şekil 46 gibi sadece bir nokta elde edilir. Aslında bir amaç fonksiyonuna kısıt koyarak hem amaç fonksiyonu açısından kolaylık sağlanır hemde diğer fonksiyon üzerine daha fazla odaklanılır. Yani örnekteki risk fonksiyonunu minimize etmek üzerine odaklanılır.

$$Min\, f_i(x)\, \forall\ i \rightarrow Min\, f_r(x)\, f_i(x) \leq \epsilon \in i, i \neq r \tag{79}$$

€-*Constraint* [50] yönteminin yukarıda tanımlanmış 79 ifadesindeki sorun € değerlerinin tahmin edilmesidir. Genelde kısıtlar ile çalışmak kolay değildir. Kısıtlı problemler genelde kısıtsız problemlere dönüştürülerek çözülür. Bu sebeple kısıtısız bir probleme kısıt koymak pek mantıklı gelmeyebilir. Bu yöntem pek etkin değildir. Ancak düzgün tasarlanırsa ve problemden önbilgi ile **€'**lar iyi ayarlanırsa genelde diğer yöntemlere göre en iyi performans ve cevapları sunmaktadır.

1.18. Mağlup Olmayan Sıralama Genetik Algoritma

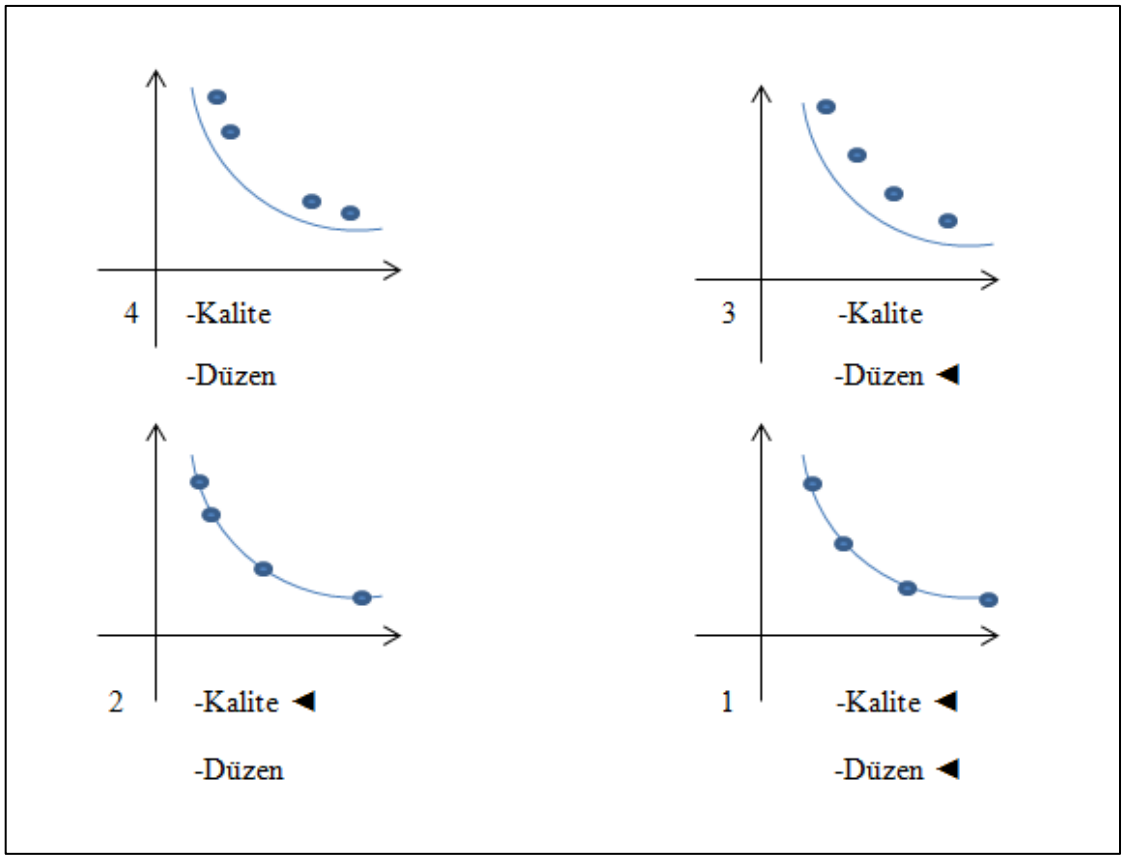
Önceki konuda dekompozisyon yöntemlerinden bahsedilmiştir. Bu kısımda ise algoritmanın bir kez koşturulması ile cevabı bulabilen, en az bilgiyle problemi çözebilen, doğrusal olmayan problemler için de kullanılabilen ve aynı zamanda *pareto front*'ın tüm noktalarını keşfedebilen yöntemlerden bahsedilecektir. Amaç fonksiyonun uzayı

sıralanabilir bir küme değildir ve sıralama operatörü burda çalışmaz. Baskın'lık (*dominance*) [42] kavramı Şekil 37'yi hatırlatarak aşağıdaki gibi tanımlanabilir.

- x, tüm A kümesinin elemanlarına galiptir
- x, C kümesinin elemanlarına mağluptur

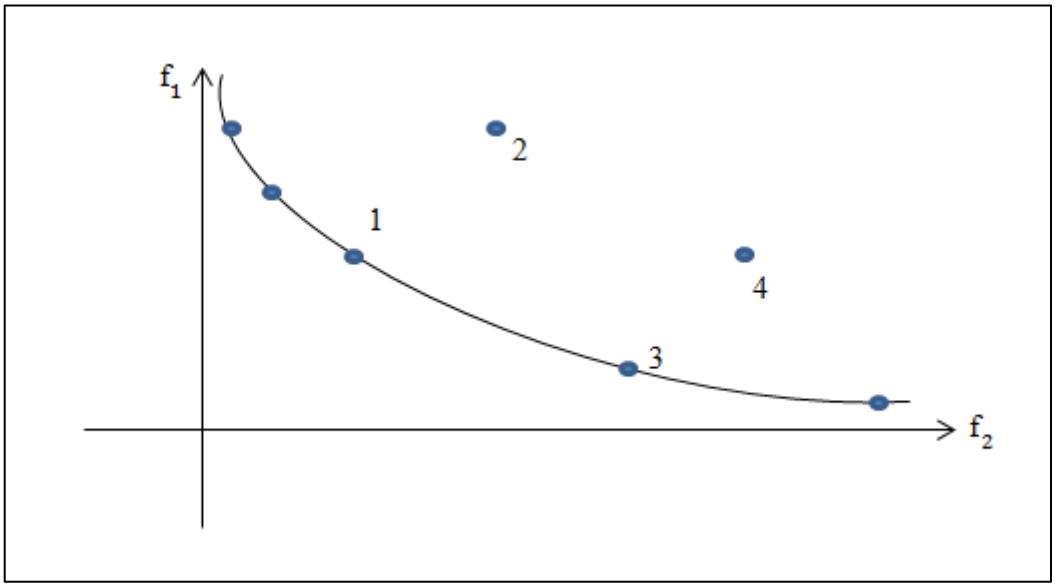
$$x \leq y \; or \; x \; dom \; y \; <=> \; \forall_i : x_i \leq y_i, \exists_i : x_{i_0} < y_{i_0} \tag{80}$$

$\forall_i : x_i \leq y_i$, yani y hiçbir açıdan x'ten iyi değildir anlamına gelir. Ayrıca $\exists_i : x_{i_0} < y_{i_0}$, x en az bir açıdan kesinlikle y'den iyidir anlamına gelmektedir. Baskın kavramı *MATLAB* dilinde tanımlıdır. Yukarıda tanımlanmış 80 ifadesindeki $\forall_i : x_i \leq y_i$ tanımına Evrensel Nicelik (*universal quantifier*) ve $\exists_i : x_{i_0} < y_{i_0}$ tanımına Varoluşsal Nicelik (*Existential quantifier*) denilir. $\forall_i : x_i \leq y_i$ için (*all*) ve $\exists_i : x_{i_0} < y_{i_0}$ için (*any*) komutu *MATLAB*'da mevcuttur. Bu Kavram *NSGA-II* ve *MOPSO* vs, gibi algoritmalarda kullanılmaktadır. *MOEA/D* [44] yöntemleri *Dominanace* kavramıyla çalışmıyorlardır. Buda önemli bir fark sayılır. Baskın kavramı doğrudan çözüm yöntemlerinde kullanılır. *NSGA-II* (*Non-dominated Sorting Genetic Algorithm*) [42,55] bir çok amaçlı, doğrudan çözen yaklaşımlı optimizasyon algoritması olarak popülasyon elemanlarını bu yöntem ile sıralayabilmektedir. Burda kıyaslama işlemi için bir kriter vardır yani x ve y arasında seçim yapılırsa galip olan (*Dominate*) seçilir ve (*Dominated*) mağlup olan elenilir; x ve y aralarında (*Dominance*) galip ya da mağlup durumu olmadığı yerde ikinci kriter (*Secondary Criteria*) [42,55] olmasına ihtiyaç duyulur.



Şekil 47. Cevapların kalite ve düzen faktörleri

Burda birçok amaçlı problemini çözmekteki amacın ne olduğunu incelemek için yola çıkılmıştır. Şekil 47'deki şekillerden hangisi istenilendir sorusuna cevap getirerek anlatmalar sürdürülmektedir. Bu konuda *Pareto Front*'a uygun yaklaşılması hedef koyulmuştur. Elde edilen cevapların mağlup olmayan (*Non-Dominated*) [42] ve nerdeyse tüm *Pareto Front*'ın örtüşmesini sağlayan cevaplar olması istenilmektedir. Şekillerde kalite (*Quality*) ve düzen (*Regularity*) [42] faktörleri vardır ve öncelik açısından kalite kriteri önde gelmektedir. Şekil 47'deki 1'de iki faktörler yani kalite ve düzenin var olması görülmektedir; 2'de kalite vardır fakat düzen yoktur; 3'te kalite yoktur, ama düzen vardır ve son olarak 4'te kalite ve düzen faktörlerinden hiç birisi yoktur. *Weighted Sum* yöntemi *Pareto*'nun içbükey (*Concave*) [42] kısımlarını keşfedebilmeme nedeniyle burda istenilmeyen yöntemlerdendir. Hedef kalite ve düzen faktörlerinden, şekilde ikisininde olmasının bir şeklini bulmaktır. İyi bir algoritma bulduğu cevaplarda önce kalite ve sonra düzen faktörlerini taşımaktadır. Bu açıklamalara dayanarak ana kriter (*Primary Criteria*) [42] cevapları kıyaslamakta tanımlanmış baskın kavramıdır. Eğer kalite kriteri ile iki cevaptan birisi seçilemezse ikinci kriter olarak düzen kullanılacaktır.



Şekil 48. Kalite ve düzen kriterlerine göre cevap seçimi

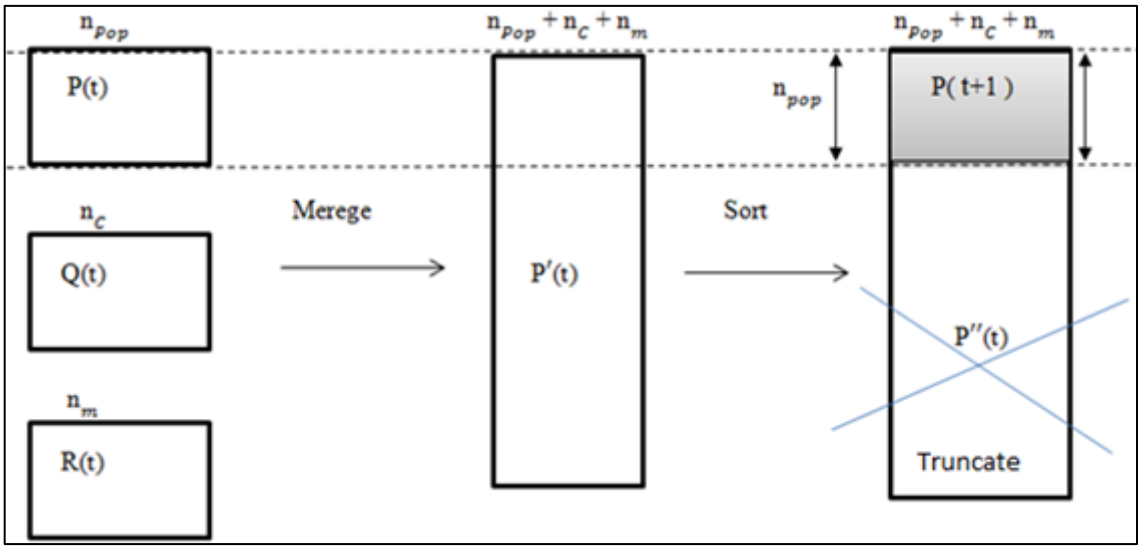
Şekil 48'de cevaplar arasında kıyaslama yapılırsa, 1 ve 2 cevapların arasından 1 seçilir. 2 ve 3 cevapların arasında eğer kendi çaplarında konuşursak hiç birisi seçilemez çünkü bibirine göre galip ya da mağlup (*Dominate or Dominated*) durumları yoktur. Ama diğer noktaların etkisini göze alarsak 2 daha kötü durumdadır çünkü 1 noktası gibi birileri 2'yi mağlup etmektedir. Ama 3'ü kimse mağlup etmemektedir; 1 ve 3 arasında yine baskınlık ilişkisi kurulmamaktadır. Ancak düzen faktörünü göze alınırsa 1'in olduğu bölgeye gereğinden fazla odak sarf edilmiştir. Ama 3'e bakılırsa, 3'ün o bölgede düzeni sağlayan bir temsilci olduğu görülür. Yani cevapların düzgün dağılmasında ve çeşitliliğin artmasında katkısı olduğu görülmektedir. Böylece tekrar araba girafiğinde fiyat ve risk örneğine dönersek çeşitli araba opsiyonlarına benzer yani noktaların (cevapların) *Pareto* üzerinde düzenli dağılması söz konusudur. Eğer 1 ve 3 arasında seçim yapılırsa düzen ve cevapların dağılmasına etkisi nedeniyle 3 seçilir. *Niched Pareto Genetic Algorithm* (*NPGA*) [51,55] ve *Vector Evaluated Genetic Algorithm (VEGA)* [51,55] gibi algoritmalarda ikinci kriterleri kullanmaktadırlar. Fakat iyi uygulanmamışlardır. *NSGA*'nın birinci versiyonunda yani *NSGA-I* algoritması düzen kriterini kullanmamıştır. Dolayısıyla etkin bir yöntemden faydalanmamaktadır. *NSGA-II* uygun ve güçlü bir algoritmadır. Şimdi *Non-dominated Sorting Genetic Algorithms* (*NSGA-II*) teorisine geçmeye sıra gelmiştir. *NSGA-II* konusu *Kalyanmoy Deb* [42] ve meslektaşlarıyla öne sürülmüş ve geliştirilmiştir.

NSGA-II en ünlü çok amaçlı evrimsel optimizasyonu algoritmasıdır. Bu algoritmaya dayalı yazılan makaleler *Paper*'ler diğerlerinden daha fazladır ve kullandığı fikir çok yaygındır. karınca kolonisi algoritması (*Ant Colony*), arı kolonisi Algoritması (*Bee colony*), *PSO* (*Particle swarm optimization*), *DE* (*Differential Evolution*) gibi algoritmaların da çok amaçlı versiyonları vardır ve *NSGA-II* fikirini kullanarak çalışmaktadırlar. Sonuçta her tek amaçlı algoritma *NSGA-II* fikrini kullanarak çok amaçlı algoritmaya dönüşebilmektedirler. *Pareto*'nun tahmin edilme yöntemi *NSGA-II* tekniğine münhasırdır. Bu konuda *Dr. Carlos A. Coello Coello*'un kitabı [42] çok kapsamlı sayılır. Bu kitap da normal genetik algoritma ile çok amaçlı genetik algoritmanın arasındaki fark cevapların sıralama tekniğinde olması üzerinde durulmuştur. Tek amaçlı normal genetik algoritma tekrar gözden geçirilirse :

Aşağıda genetik algoritmanın işlem adımları sıralanmıştır [20,28]:

- İlk popülasyonun oluşturulması ve değerlendirilmesi
- Ebeveyn *seçimi*, çaprazlama (*Crossover*) uygulanması ve çocuklar popülasyonun oluşturulması
- Ebeveyn *seçimi*, mutasyon yapılması ve mutasyon popülasyonun (*offspring*) oluşturulması
- Ana popülasyonu, çocuklar popülasyonu, mutasyon Popülasyonların birleştirilmesi (*Merge*) ve *yeni popülasyonun* oluşturulması
- Eğer sonlandırma koşulları sağlanmazsa, ikinci aşamadan tekrar başlanır
- Son

Normal genetik algoritma ile çok amaçlı genetik algoritmanın farkı yeni popülasyonun seçimindedir. Burda konu sıralanabilmezliktedir, Ayrıca tek boyutlu *Sorting* ya da sıralama operatörünü kullanılamazdır. Dolayısıyla yeni birçok aşamalı operatör tanımlanmaktadır. Bu operatör bir kere cevapların kalitesini baz alarak sıralama işlemini yapar ve sonra kalite faktörü belirleyici olmadığı durumlarda ikinci kriteri düzen olarak esas alır. Örneğin: Futbol maçlarında takım elemelerinde bazı durumlarda puan belirleyici olmayabilir bu durumda gol sayısı ikinci kriter olarak kullanılır. *Kalyanmoy Deb* [51] *NSGA-II* algoritmasının seçkinci (*Elitist*) [42] olmasının üzerinde durmuştur. Bu yüzden burda birleştirme, sıralama ve kesme senaryosu (*Merge & Sort & Truncate*) [42] senaryosu kullanılmıştır.

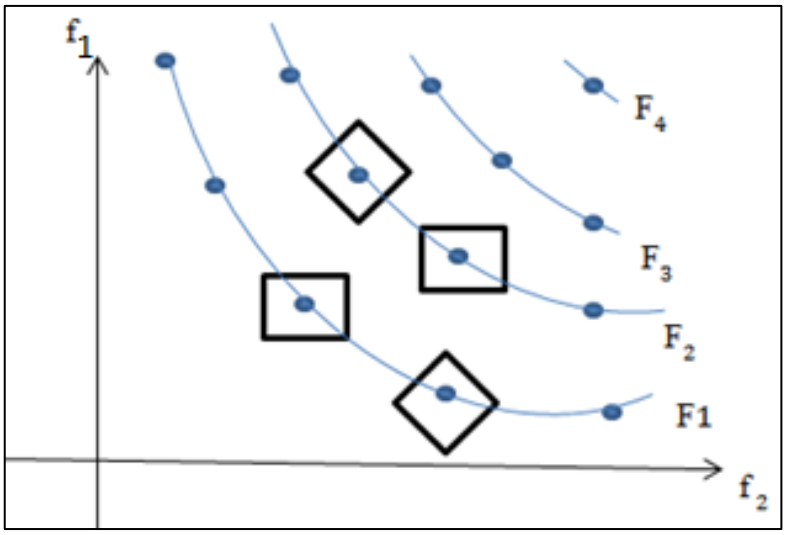


Şekil 49. Birleştirme, sıralama ve kesme senaryosu

Şekil 49'da gösterilen sıralama (*Sorting*) iki aşamalı işleme dönüşmektedir. Cevapları rütbelendirmek için bir rütbe (*Rank*) kavramı tanımlanır. Rütbenin belirleyici olmadığı yerlerde ikinci kriter olarak Yoğunluk mesafesi (*Crowding Distance*) [42] kavramı tanımlanarak devreye girer. Yoğunluk mesafesi daha sonralar *NSGA-II* algoritmasına eklenmiştir. *NSGA-I* ilk versiyonunda yoğunluk mesafesi yerine uygunluk paylaşımı (*Fitness Sharing*) kullanılmıştır.

1.19. Rütbeleme (*Ranking*)

Aşağıda Şekil 50'de popülasyon elemanlarının değeri birbiri ile kıyaslanır. Şu hususu tekrar vurgulamakta fayda var ki, f_1 ve f_2 alanları amaç fonksiyonların alanlarıdır (*phenotype*) ve hep bu uzaydan bahsedilmiştir ve bahsedilecektir.



Şekil 50. Cevapları cepheleyerek rütbeleme yapılması

Popülasyon elemanlarının değeri birbiri ile kıyaslanır. Her elemanın kaç kere mağlup olduğu bir sayaç yardımı ile sayılır. Bu duruma göre eğer diğerlerin aracılığıyla mağlup olmayan eleman ya da cevaplar bulunursa bunlar F_1 ya da birinci cepheyi (*Front*) oluşturmaktadırlar, Ayrıca bu cephenin içindeki cevapların rütbeleri 1 olmaktadır. Sonra popülasyon elemanlarından F_1'ler kaldırılır yani sonraki cepheyi bulmakata önceki aşamada bulunan F_1'lerin bozucu etkisi sonraki F_2 adayların üzerinden kaldırılır. Böylece geriye kalan popülasyonlardan mağlup olmayanlar ise F_2'yi oluşturmaktadırlar. Bu gidişi aynen böyle sürdürerek F_1ve F_2'leri kaldırarak geriye kalanların mağlup olmayanları F_3 ve böyle devam ederek F_4 ve F_5'ler ortaya çıkmış olacaktır. Şimdi bunlar matematik diliyle ifade edilecektir.

1.20. NS - Mağlup Olmayan Sıralama Algoritma

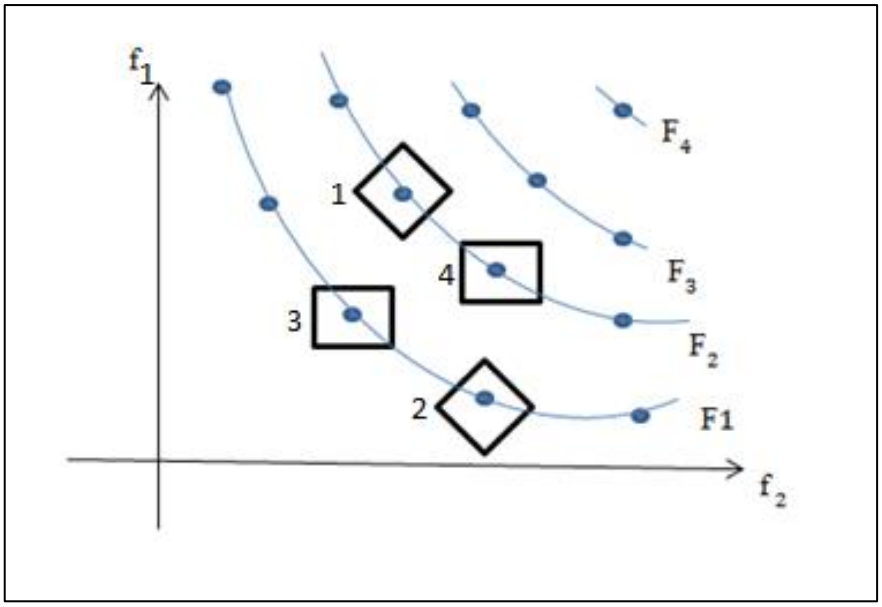
Mağlup olmayan sıralama algoritma (*Non-Dominated Sorting Algorithm*) [42] :

- P aracılığıyla mağlup olan popülasyonun elemanlarının kümesi → S_p
- P, diğerlerin aracılığıyla mağlup olmasının sayacı → n_p
- Tüm popülasyon elemanları için S_p= { } ve n_p= 0 değerleri alınır.
- P gibi popülasyondan her eleman için
- q gibi popülasyondan her eleman için

- Eğer p, q'ya galip gelirse → q elemanını S_p'ye ekle.
- Eğer q, p'ye galip gelirse → bir birim n_p'yi arttır.
- Tüm n_p'si sıfır n_p= 0 olan popülasyon elemanlarını F_1'e ekle.
- cephelerin sayacını K=1 değerine eşitle.

❖ *Q'yu F_{k+1}'in taslağı olarak kayıtla.

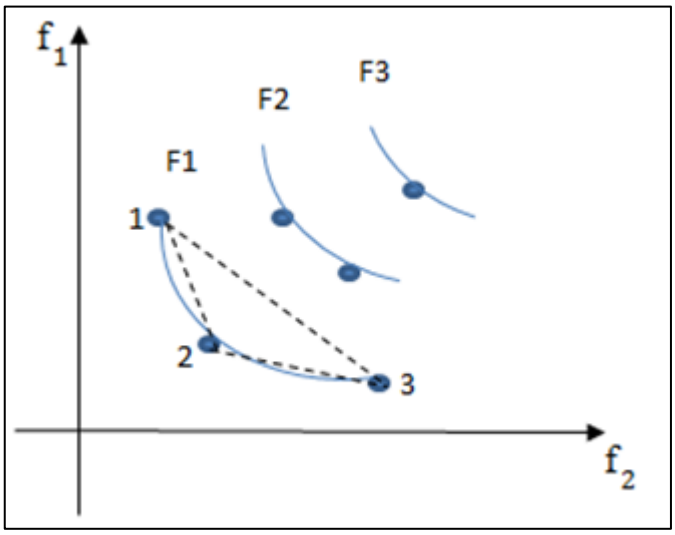
- F_k'dan P gibi her eleman için, S_p'den q gibi her eleman için (P aracılığıyla mağlup olan tüm q'lar) n_q'dan bir birim eksilt.
- Eğer n_q= 0 olursa q'yu Q'ya ekle. Eğer Q boş olursa sıralama işlemi sona ermiştir.
- Eğer Q boş olmadıysa F_{k+1}'ri Q ile eşitle.
- Bir birim K sayacına arttır ve * aşamasına geri dön.



Şekil 51. Rütbeleme

Rütbenin olmasının önemi şöyle yukarıdaki Şekil 51'de gösterilmiştir. f_1 ve f_2 uzayında rütbeleme olmadığı halde 3 ve 4 arasında seçim yapılabilirdi ve 3 noktası 4'ü mağlup ettiğinden dolayı 3 seçilebilirdi. Fakat 1 ve 2 cevapların birbirine göre galip ya da mağlup durumu yoktur. Yani aralarında baskın durumu yoktur, Ama rütbeleme bunu sağlamış oldu ve şunu görmekteyiz ki 2 noktasının, rütbesi 1 ve F_1 kümesindedir. 1 noktasının rütbesi 2 ve F_2 kümesindedir. Dolayısıyla 1 ve 2 arasında, rütbesi bir olduğuna

göre 2 seçilmektedir. Düşük rütbelerin önemi çoktur ve öncelik düşükten yükseğedir. Bazı durumlarda rütbeleme kriteri belirleyici olmayabilir. Şekil 52'ye göre 1 ve 2 ve 3 noktaların arasında nasıl seçim yapılacağı incelenecektir. Görüldüğü gibi bu noktalar F_1 cephesinin içindeler ve rütbeleri 1 olduğundan dolayı rütbe açısından seçim yapılamamaktadır.

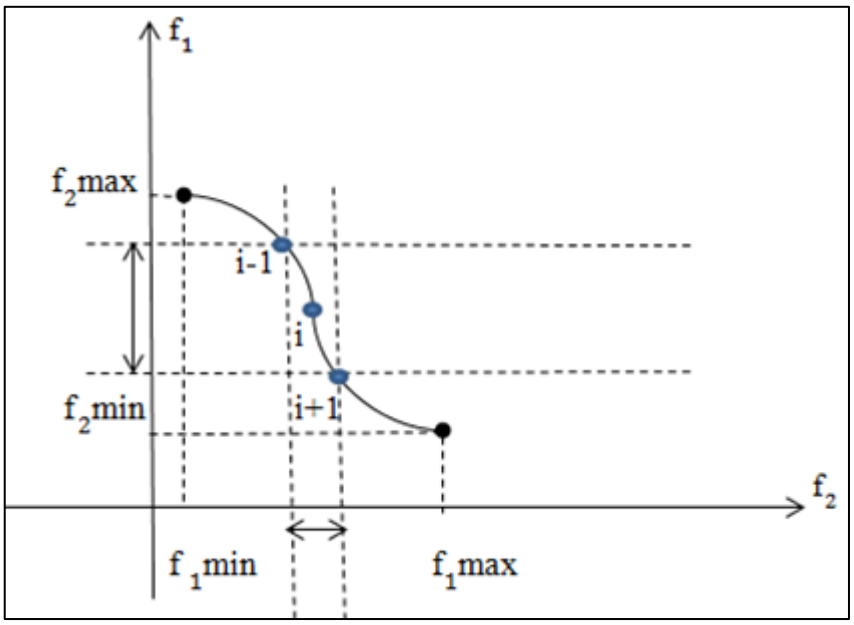


Şekil 52. Cevapların eşit rütbede olduğunda seçimi

Burda ikinci kritere ihtiyaç duyulur. 1 ve 2, 2 ve 3 noktaların örtüştükleri alan 1 ve 3 noktalarına göre daha azdır. 1 ve 3 yüksek örtüşme alanına sahip oldukları için seçilir. Geri kalan 2 noktası örtüşmekte düşük katkısı olduğundan dolayı elenmeye en iyi seçenektir. Bu duruma matematik tanımı vardır ve Yoğunluk mesafesi (*Crowding Distance*) [42] ikinci kiriter olarak isimlenmektedir.

1.21. Yoğunluk mesafesi

Şekil 53'te gösterildiği gibi yoğunluk mesafesi [42] cevabın bir önceki ve sonraki komşusu ayrıca popülasyonun ilk ve son elemanına göre hesaplanır.



Şekil 53. Yoğunluk mesafesi

Aşağıdaki 81, 82, 83 ve 84 ifadelerinde, j : *Objective* sayısı ve i : *Front*'ın eleman sayısıdır.

$$d_i^1 = \frac{\left|f_1^{i+1} - f_1^i\right|}{f_1^{max} - f_1^{min}} \tag{81}$$

$$d_i^2 = \frac{\left|f_1^{i+1} - f_1^i\right|}{f_2^{max} - f_2^{min}} \tag{82}$$

Yoğunluk mesafesinin genel ifadesi aşağıda tanımlanmıştır:

$$di = d_i^1 + d_i^2 \tag{83}$$

Yoğunluk mesafesi fazla olursa daha iyi sayılır yani daha fazla alan örtüşmekte demektir.

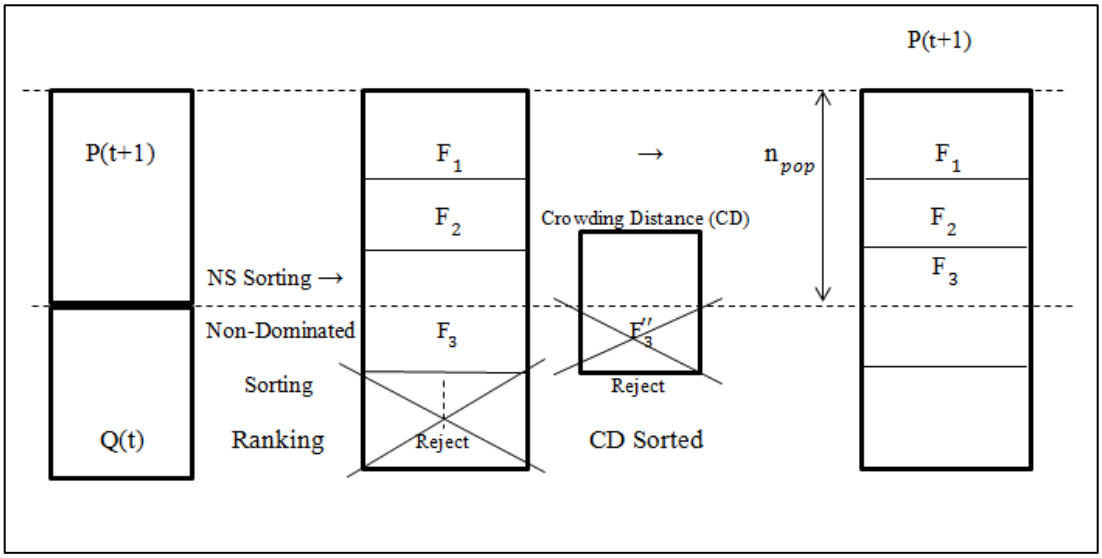
$$d_i^j = \frac{\left| f_j^{Önceki} - f_j^{Sonraki} \right|}{f_j^{max} - f_j^{min}} \tag{84}$$

$$d_i^j = d_i^1 + d_i^2 + \ldots + d_i^m = \sum_{i=1}^{m} d_i^j$$

Şimdi yeni sıralama operatörün tanımlamasının vakti gelmiştir.

1.22. Yeni Popülasyonun Sıralaması ve Seçimi

Şekil 54'de gösterildiği gibi F_1 ve F_2 yeni popülasyon p(t+1) içinde sığmaktadırlar, fakat F_3'tekiler rütbe açısından eşit durumdalar ve aynı zamanda F_3'ün bir kısmı yeni popülasyona sığmamakta ve elenmek zorundadır. Burda ikinci kriter olarak *Crowding Distance* (*CD*) devreye girer.



Şekil 54. Yeni popülasyonun sıralama ve seçimi

Böylece F_3 kendi içinde elemanları *Crowding Distance*'a göre kıyaslanarak sıralanır. *Crowding Distance*'ı yüksek olanlar yeni popülasyona geçerken *CD*'si düşük olanlar elenmektedirler. Klasik genetik algoritmasında ebeveyn seçimi ikili turnuva seçimi (*Binary Tournament Selection*) [42] yöntemi ile gerçekleşmektedir. Burada da *Binary Tournament*

Selection kullanılmıştır. Değişen tek şey, baskın mantığı ile seçimi gerçekleştirilmesidir. ikili turnuva seçimi için x *dom* y operatörü kullanılmaktadır.

- Eğer x'in Rütbesi y'den düşük Olursa → x *dom* y
- Eğer x ve y rütbeleri eşit durumda olursa → x'in *CD*'si y'den yüksek olursa

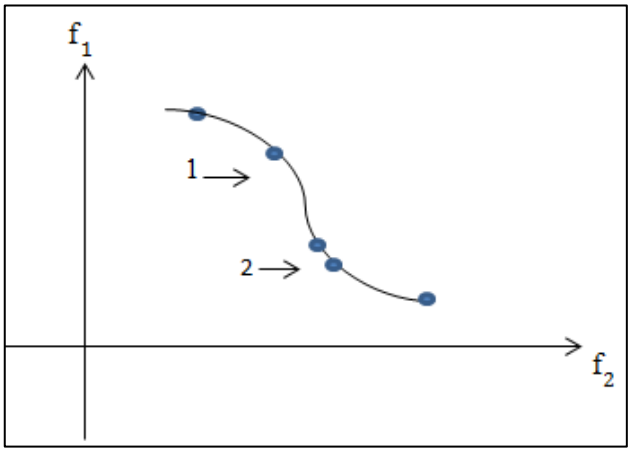
İkili turnuva seçimi [55] yönteminde popülasyon elemanlarından ebeveyn olmayan iki tane seçilir. İlk etapta bu iki tane elemandan rütbesi daha düşük olanını seçilir sonra eğer rütbe açısından eşit durumdalarsa *CD*'si yüksek olanını ebeveyn olacaktır. Eğer 10 ebeveyn seçimi yapılırsa ozaman 10 kere *Binary Tournament Selection* uygulaması yapılmalıdır. Rulet tekerleği (*Roulette Wheel*) [31] ile ebeveyn seçimi :

- Rulet Tekerleği Tekniği ile aşağıdaki ihtimalleri sağlayarak bir cephe seçilir.

$$P_r\{F_1\} \geq P_r\{F_2\} \geq P_r\{F_3\} \geq \ldots \tag{85}$$

- Seçilen cephenin bir elemanını Rulet Tekerleği Tekniği [55] ile rastgele seçilir.

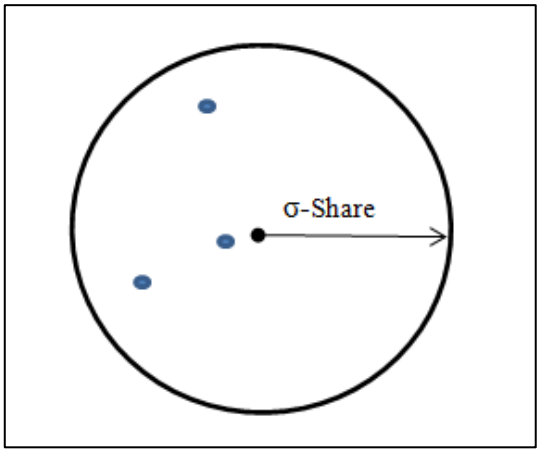
Elemanların seçim ihtimalleri *CD*'si yüksek olanlara aittir. *NSGA-I* ile *NSGA-II* algoritmaların farkı uygunluk Paylaşımı (*Fitness Sharing*) [42] adında bir yöntemdedir. Örneğin: varsayılan hoca çözmek için bir ödev vermiştir ve sonra iki kişi birlikte ödevi yapmışlardır. Eğer hoca titiz ve katı davranırsa ödev notunu 100 verir. Ama bu notu iki kişiye bölerek verir yani ödevi çözen her kişi başına notu 50 düşer.



Şekil 55. İki cevap yanyana birbirine bozucu etkileri

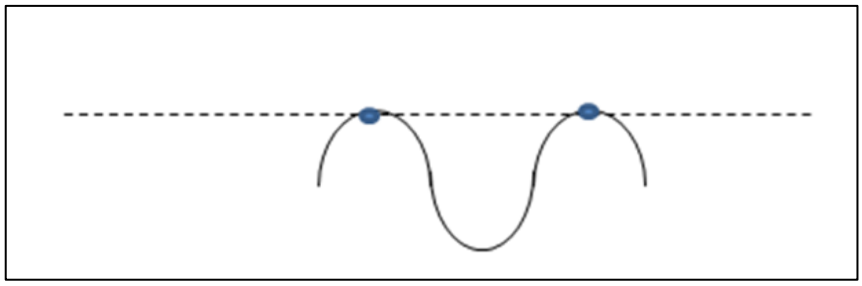
Şekil 55'de gösterildiği gibi iki cevap yanyana birbirine bozucu etkileri vrardır. Bu örnekteki iki kişinin beraber ödevi yapmasına benzer bir durumdur. Dolayısıyla Şekil 55'te

durum 2 önem taşımamaktadır. Durum 1 kalabalık olmayan alan olduğuna göre ve keşifsiz kalan bölgeleri bulduğuna göre daha fazla önem taşımaktadır. Burda her cevap için çapı *sigma* adında bir dairesel alan ile belirlenmiştir.



Şekil 56. σ-Share

Şekil 56'daki çemberin içinde bulunan cevaplar uygunluk (*Fitness*) üzerinde bozucu etkileri vardır. Bu rütbeyi yükseltmeye neden olur. Rütbenin olmasının yanısıra bunun etkisi alanında olan cevapların yani birbirine benzer cevapların bozucu etkileri yaklaşık hesaplanır. Bu işleme *NPGA* (*Niched Pareto Genetic Algorithm*) [42] ya da deyim olarak *Niching* söylenilir. Bu kavram *David E. Goldberg* öne sürülmüştür ve *σ-Share, NPGA* yönteminde de kullanılmıştır. Bir kürenin iç uzayını canlandırırsak bozucu etkileri olan noktalar kürenin içinde yer bulmuşlardır ve dışında olanlar etkisiz kalanlardır. *Niching* fonksiyonunda *σ-Share* [42] parametresini belirlemek kolay değildir ve bazen problemin tasarımından daha zor olmaktadır. *NSGA-I* algoritmanın esas güçsüzlüğü *σ-Share* parametresinin ayarlanmasındadır. *Multi Modal Optimization* konusunda parametrelerin ayarlamasına göre *Niching* yöntemleri vardır. *Multi Modal Optimization* yönteminde bir amaç fonksiyonu vardır ki burda benzer iki cevaba rastlamak ve tek boyutlu uzayda her ikisinide keşfetmek söz konusudur.



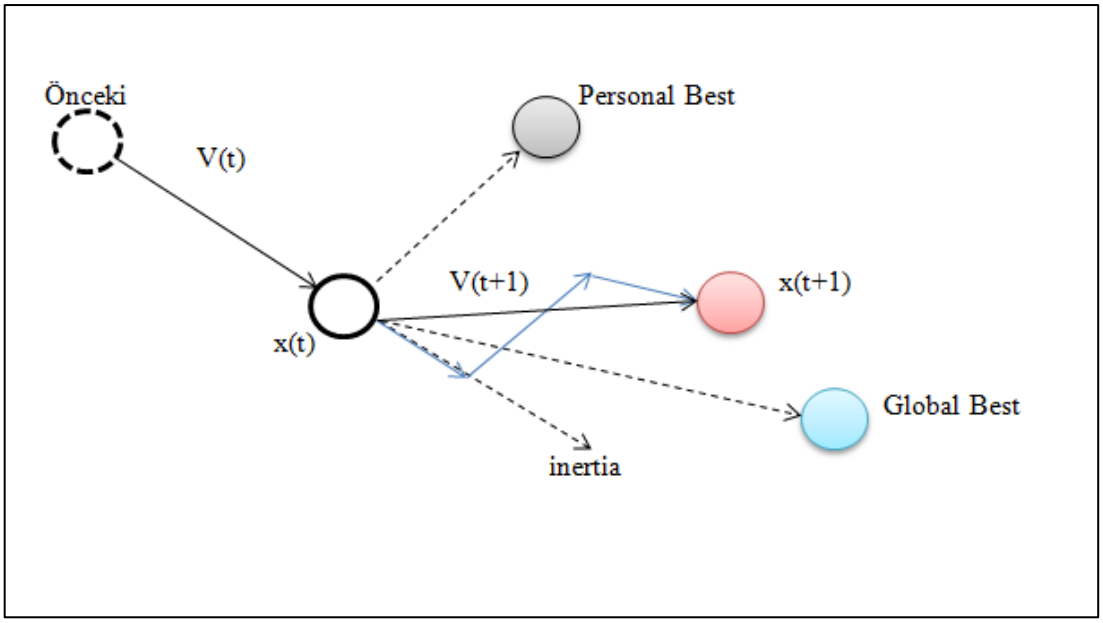
Şekil 57. Benzer cevaplar

Şekil 57'deki gibi bir amaç fonksiyonunu olduğunu var sayalım. Şekildeki iki nokta benzer ve birbirine denk gelmektedir. Ayrıca her ikiside keşfedilmek isteniyor. *Multi Modal Optimization* konusunda *Niching* yöntemleri çok kullanılmaktadır. Aslında bu yöntem benzer cevapları birnevi cezalandırır. *Variable-length Genetic Algorithm* (*MVGA*) [52] gibi algoritmalar bunu *Tabu Search* algoritmasına benzer *Tabu* ya da cezalandırarak sağlamaktadırlar. *Multi Modal* konusunda avantajı tüm *Pareto*'yu tanımak ve benzer cevapları cezalandırmaktır.

1.22.1. Çok Amaçlı Parçacık Sürü Optimizasyonu

MOPSO (*Multiple Objective Particle Swarm Optimization*) algoritması 2004 yılında *Dr. Carlos A. Coello Coello*'nun [53] üzerinde yaptığı çalışmalar ile ileri sürülmüştür. Tek amaçlı sürekli optimizasyonu problemlerini çözen *PSO* gibi *MOPSO da* çok amaçlı optimizasyon problemlerini çözen genelleştirilmiş bir *PSO* dur. *PSO* algoritmasına birtakım değişiklikleri uygulayarak onu *MOPSO'ya* dönüştürmek mümkündür. Aynen *GA* (*Genetic Algorithm*) genetik algoritmasını *NSGA-II* algoritmasına dönüştürülmesi gibi birtakım operatörlerde değişim yapılmıştır. Bu değişimin nedeni öncede bahsedildiği gibi problemin çok amaçlı olmasıdır. *MOPSO* teorisine geçemeden önce *PSO* algoritmasını gözden geçirmek gerekir. Böylece nelerin değişilmesi gerektiği daha iyi anlaşılır. Bu tezde *MOPSO*'nun teorisi ve uygulaması için ana kaynağı olarak birinci yazarı *Dr. Carlos A. Coello Coello* (*Handling Multiple Objectives With Particle Swarm Optimization*) [53] makalesi kullanılmıştır. Tabiki *MOPSO* adlandığı algoritma saf bir algoritma değildir ve bazı konuları *PESA-II* (*Pareto Envelope-based Selection Algorithm*) algoritmasından almıştır ve bu Algoritmanın temeli genetik algoritmasına dayanmaktadır. *MOPSO* ile optimizasyon problemlerini çözmekte işin genetik kısmını kaldırarak yerine *PSO* koyup kullanmışlardır. Bu açıdan ana katkı *PESA-II* algoritmasına aittir. *MOPSO* algoritmasını anlatmak için önce *PSO* algoritması gözden geçirilecek ve sonrada *MO* teriminin üzerinde durulacaktır. Parçacık Sürü Optimizasyonu (*Particle Swarm Optimization*) [54] algoritmasının özünde sürü zekası (*Swarm İntelligence*) [54] adlı bir paradigma vardır ve genetik algoritmasında gördüklerimizden daha farklıdır. Sürü zekasında bir takım etmenler (*agent*) tek başına iş yürütemezler. Dolayısıyla sürü içerisinde yer alarak ve işbirliği yaparak sonuca ulaşmaktadırlar. Sürü şeklinde öyle bir sonuç elde edilir ki tek başına yapılması mümkün olmamaktadır. Bunlara *PSO, Ant Colony, Bee Colony* vs, algoritmalarını örnek verebiliriz. Bazen *PSO* [54] algoritması yanlışlıkla deyim olarak kuşlar sürüsü algoritması adlanılır. Genetik algoritmasında kromozom ve burda her cevap parçacık (*Particle*) [54] olarak isimlenir ve kullanılır. Arama uzayında yeni cevaplar bulmak için birtakım kurallar ve ilkeler tanımlanmıştır. Bu konuda olan en önemlisi hareket kavramıdır. Bu hareketler bilgi alışverişi ile popülasyon elemanlarının arasında yakınsama kurulmasına neden olmuştur. Hareketlerin kaynağı üç şeyden ibarettir [54] :

- Parçacıkların önceden gösterdiği tepkiler
- Parçacıkların arama uzayında şimdiye kadar deneyim geçirdikleri en iyi pozisiyon
- Tüm popülasyon elemanlarının deneyim geçirdikleri en iyi pozisiyon



Şekil 58. Parçacığın hareketi

Şekil 58'de gösterildiği gibi bir parçacığın önceki durumunda olduğu var sayalım. Parçacık v(t) hız vektörü ile şimdiki pozisiyonu x(t)'ye aktarılmıştır. Parçacığın uzayda en iyi deneyim geçirdiği yer *personal best* [54] ile adlanmaktadır. Tüm popülasyon elemanlarının deneyim geçirdikleri en iyi pozisiyon *global best* [54] olarak isimlenmektedir. x(t)'den sonra bir hareket daha oluşturulması istenirse, eğer şimdiki x(t) tepkisine uyarak hareket ederse bu hareket atalet (*inertia*) doğrultusunda böyle sürecektir. Eğer bencilce yani sadece kişisel deneyim ya da nostaljik olarak hareket ederse bu *personal best* doğrultusunda devam edecektir. Eğer tüm popülasiyonun (toplum) modeline fazla uyum gösterilirse bu hareket *global best* doğrultusunda olacaktır. Birinci hareket bir çeşit atalet sayılmaktadır. İkinci hareket nostaljiktir ve tek başına işe gelmemektedir ve bunun gibi sırf topluma uyumlu hareketler işe gelmemektedir. Bu konuda bahsi geçen üç çeşit harekettin bir bileşimi uygun görülmektedir. Yani biraz ataletin doğrultusunda, biraz *personal best*'in ve birazda *global best*'in doğrultusunda hareket etmeleridir. Bu parçacığı yeni pozisiyona x(t+1) ulaştırır. Şimdi biraz matematiksel olarak konuşulursa üç vektörün bileşkesini hesaplamak için birinci vektörün başlangıcı sonuncu vektörün sonuna bağlınır.

Böylece yeni hız vektörün v(t+1) [54] oluşturular. Yani elde edilen bileşke bir bölümü önceki hızdan ve bir bölümü şimdiki pozisyonu *Personal Best*'e dönüştüren vektörden ve bir bölümüde şimdiki pozisyonu *Global Best*'e dönüştüren vektördendir.

$$V_i\,(t+1) = W\,V(t) + C_1 r_1 (p_i(t) - x_i(t)) + C_2 r_2 (G(t) - x_i(t)) \qquad (86)$$

Denklem 86'da *W* önceki hızdan bir atalet Katsayısı, $V_i\,(t+1)$ hız vektörüdür, C_1 öğrenim ya da nostaljik katsayısıdır ve r_1 (0,1) arasında rastgele sayılardır. Aşağıdaki denklem 87'de parçacığın yeni pozisyonunun gösterilmektedir.

$$x_i\,(t+1) = x_i(t) + V_i\,(t+1) \qquad (87)$$

Aslında anlatılan konu iki operatörlü matematik ifadeleri ile yazılmıştır. Bu işlemler her parçacık için ayrı ayrı yapılmalıdır. Dolayısıyla ifadeler bir i indeksi ile gösterilmiştir. Aslında *PSO* algoritmasının genel şeması şöyle yazılabilir [54] :

- İlk popülasyonun oluşumu
- Parçacıklardan her biri için en iyi parçacık (*Global*) ve en iyi *Personal Best*'in belirlenmesi

→ Tüm parçacıkların hız ve pozisyonlarının güncellenmesi

- Parçacıklardan her biri için en iyi parçacık (*Global*) ve en iyi *Personal Best*'in belirlenmesi
- Sonlandırma koşulları sağlanmadığı durumda, → aşamasına git aksi takdirde SON.

Algoritmanın ana çekirdeği altı çizilmiş üçüncü aşamas sayılmaktadır. *MOPSO* ve *PSO* arasındaki fark en iyi parçacık ve en iyi *Personal Best*'in belirlenmesindedir yani çok amaçlı (*Multi Objective*) konusunda en iyi teriminin tanımı üzerinde durulmuştur [53]. Çok amaçlı temellerini gözden geçirdiğimize göre mağlup olmayan cevabın birinci derecede kalitesi sonrada düzeni önemlidir. Dolayısıyla en iyi parçacığı ve en iyi *Personal Best*'i belirlemekte bu iki kriterleri göz önünde bulundurulması gerekmektedir.

1.22.2. Depo (*Repository*)

MOPSO algoritmasına depo (*repository*) [53] ya da *archive* kavramı eklenmiştir. *Pepository* popülasiyonuyla işimiz yoktur ve onu yalnız bir arşiv olarak kullanmaktayız. Bu arşiv *non-dominated* cevaplardan oluşur ve algoritmanın dışında saklanılır. *NSGA-II* algoritmasında böyle bişey söz konusu değildir. Yani bulunan en iyi cevaplar yinede popülasiyonun içindeler ve buna deyim olarak bağımsız arşivleme (*self-archiving*) [53] denilir. Ama birçok algoritmalar *SPEA-II* (*strength pareto evolutionary algorithm*) [53] ve *PESA-II* (*pareto envelope-based selection algorithm*) gibi buldukları mağlup olmayan cevaplarını bir arşivde saklamaktadırlar. Buna bazı kitaplarda örneğin: *engelbrecht* kitabında ünlülerin salonu (*hall of fame*) da denilmektedir [55]. Algoritma şimdiye kadar bulduğu en iyilerini ünlülerin salonuda sınırlı kapasitede saklamaktadır. Aslında bu *pareto*'dan bir yaklaşıktır ve popülasyon ile birleştirilmiyor çünkü işlem serbestliğini engellemektedir. Bu konuda iyi çözümleri saklamaya *elitizm*'in olması gerekir. Ama bu husus, fazla dikkatli davranmaya neden olmamalıdır. Yani eldeki çevapları kaybetmemk uğruna iyi arama (*explore*) yapılamayacaktır. Tüm algoritmalar arasında *NSGA-II* istisnadır. Çünkü elitizmi öyledir ki işlem esnasında tüm popülasyonu endişe duyulmadan ve bilgi kaybı olmadan değişimlerin etkisi altında bırakır. Bundan dolayı serbestlikle popülasyonun kendisi arşiv olarak kullanılmıştır. Bu *NSGA-II* 'yi istisna kılan elitizm türünün özelliğidir. Ama *MOPSO* algoritmanın böyle bir özelliği yoktur. Dediğimiz gibi hareketlerin kaynağı üç şeyden ibarettir :

- Parçacıkların önceki hızları, atalet terimi (bu ilk versyonlarda mevcut değildir).
- Nostalji Terimi
- Toplumun optimumuna yönelim yani *Global Best* terimi

Denklem 49'daki atalet ve *Personal Best* ya da nostaljik terimleri incelemelerde yer almayacaktır. Fakat *Global Best*'in kavramı burda değişilmelidir. Artık *Global Best* sabit nokta değildir ve her parçacık her hareket ettiğinde rastgele olarak depodan bir elemanını *Leader* olarak seçmektedirler. *Leader* Seçimi depodan seçilmesi zorunludur çünkü ancak muğlup olmayan şey *Leader* olabilir. Klasik *PSO* algoritmasında bir tane amaç vardır ve hepsi için *Leader* sayılır. Ama burda kaç sayıda mağlup olmayan eleman ve birbirlerine görede üstünlükleri olmayan durumlar söz konusudur. Dolayısıyla bunlara *Leader* olmak için eşit şans tanınmalıdır. Buyüzden *Global Best* yerine depo elemanlarının biri rastgele seçilmektedir. Bir başka fark da *Personal Best*'in belirlenmesindedir. *Personal Best* yeni

Pozisyona ulaşırsa, güncelleme aşamasında kıyaslama yapılacaktır. Yani yeni pozisyon öncekine göre iyi olup olmadığı için kıyaslama yapılır. Bu yeni Pozisyon iyi ise yeni *Personal Best* sayılacaktır. Aksi taktirde önceki *Personal Best* olarak yerinde kalacaktır. Genede kıyaslama işlemi söz konusudur ve çok amaçlı optimizasyon problemlerinin alanında yapılması gerekmektedir.

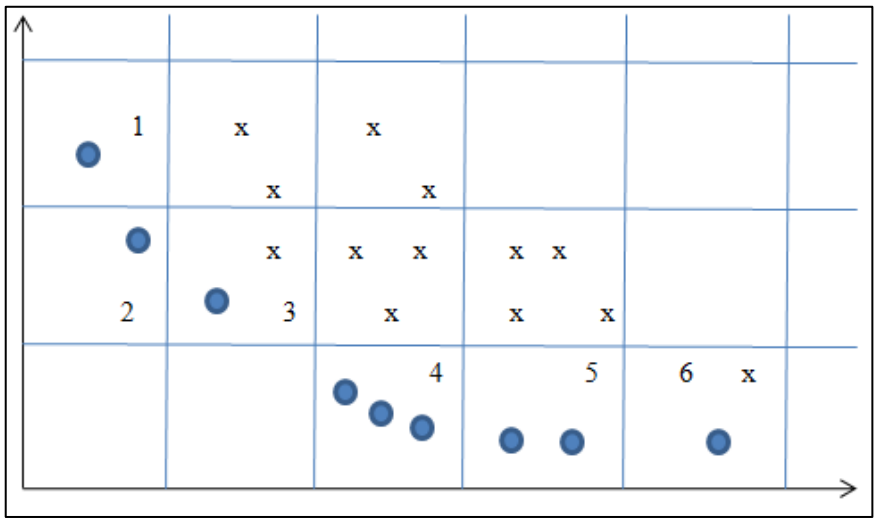
1.22.3. MOPSO Algoritma

Aşağıda *MOPSO* algoritmanın işlem adımları sıralanmıştır [53]:

- İlk popülasyonun oluşumu
- Mağlup olmayan elemanları bulup popülasyondan ayırıp ve onları depoda saklamak.
- Keşfedilen amaç fonksiyonun uzayınnın şebekelemesi (*Grid*) .
- Her parçacık depo elemanlarının arasından bir *Leader* seçerek hareket etmektedir.
- Her parçacığın *Personal Best*'i güncellenir.
- Şimdiki popülasyonun mağlup olmayan elemanları depoya eklenir.
- Deponun mağlup olan elemanları elenir.
- Deponun elemanları belirlenen kapasiteyi aşarsa fazlalıkları elenir.
- Sonlandırma koşulları sağlanmadığı takdirde üçüncü aşamaya aksi takdirde, SON.

1.22.4. Şebekeleme (*Grid*)

Var sayılan cevaplar şöyle bir alanda dağılmış olsunlar :



Şekil 59. Şebekeleme (grid)

Şekil 59'de gösterildiği gibi mağlup olmayanlar depo içinde yer alabilmektedirler. İkinci keriter, yani düzeni sağlamak için bu uzay şebekeleme yapılır. Yukarıdaki cevaplardan birini seçmek için artık birey tabanlı seçim (*individual base selection*) mantığı kullanılmıyor. *PESA-II* makalesine bakarsak bölge tabanlı seçim (*region base selection*) [53] önerilmektedir yani artık burda birey tabanlı seçimi mantığı geçerli olmamaktadır. Deponun hangi elemanı seçilme düşüncesi yerine şebekelemenin hangi hanesinin seçilmesi sorusu söz konusu olmaktadır. Sonra seçilen hanenin elemanlarından seçim yapılacaktır. Şimdi deponun elemanları kaç hanenin içinde yer almıştır sorusuyla Şekil 59'a bakarsak 6 hane bulunmuştur. Burda hanelerden hangisinin eleman yoğunluğu fazladır sorusuyla konuyu anlatmak için sorulara açıklama getirerek yola çıkıyoryz. Şekil 59'de numarası 4 hanenin eleman yoğunluğu en fazla olanıdır ve 1,2,3,6 numaralı hanelerin eleman yoğunluğu numarası 4 hanesine göre daha düşüktür. Bunların arasında şeçim önceliği 1,2,3,6 hanelerine ayittir ve 4 hanesinin önceliği en düşüğüdür. Başka bir deyişle önce depo elemanları belirlenir sonra fenotip uzayı şebekelenir ve sonrada deponun elemanlarından içinde bulunan, haneler (*cell*) belirlenir. Bunlardan bir hane seçilir ve seçilen hanenin elemanlarından birisi rastgele seçilecektir. Eleman yoğunluğu fazla olan hanelerin seçilme ihtimali düşüktür ve aksine eleman yoğunluğu düşük hanelerin seçilmesi

ihtimali yüksek olmaktatır. Seçim işlemi bir ayrık dağılımlı ihtimali ile gerçekleştirilmektedir ve bu ayrık elemanların örneklemesi rulet tekerleği ile gerçekleşmektedir. Yani her hanenin seçim ihtimali hesaplanır sonra bunlara *roulette wheel* [55] uygulanır. Aşağıda hanelerin indeksi i, seçim ihtimali p_i ve i'ncı hanenin eleman sayısı n_i ile gösterilmiştir.

- $0 \leq p_i \leq 1$
- $\sum_i p_i = p_1 + p_2 + \ldots + p_n = 1$
- $n_i \leq n_j <=> p_i \geq p_j$: Yetkinci seçim

Boltzman [55] Yöntemi :

$$p_i \; \alpha \; exp\,(-\beta \, p_i) \qquad : \tag{88}$$

Yukarıdaki β (*Selection Pressure*) [55] ifadesinde β = 0 olursa tüm ihtimaller birbirine göre eşit durumda olacaklardır. β = 1 olursa yalnız eleman yoğunluğu düşük olan haneler seçilmekte olacaklardır.

$$p_i = \frac{e^{-\beta n_i}}{\sum_j e^{-\beta n_j}} : \beta > 0 \tag{89}$$

Böylece yukarıdaki 89 ifadesinde her 3 şart kurulmuş olmuştur. Bahsi geçen makalede [53] ters orantılı (*inversely proportional*) [55] yöntemi önerilmiştir.

$$p_i \; \alpha \; \left(\frac{1}{n_i}\right)^{\beta} \rightarrow p_i = \frac{\left(\frac{1}{n_i}\right)^{\beta}}{\sum_j (\frac{1}{n_j})^{\beta}} \; : \; \beta > 0 \tag{90}$$

β = 0 olursa tüm haneler eşit ihtimaller ile seçilecektir. β = ∞ olursa kapasitesi 1 eleman olanlı haneler seçilme ihtimalinde olacaklardır. İhtimali 1 değeri onların arasında bölünür. Yani ihtimal 1 değeri eleman sayısının 1 olan hanelerin arasında bölünecektir. Rulet Tekerleği (*Roulette wheel*) [55] örnekleme tekniği Şekil 26'da gösterilmiştir. Eğer bir gösterge, rulet tekerleği olursa ve rastgele olarak tekerleği döndürürsek göstergenin p

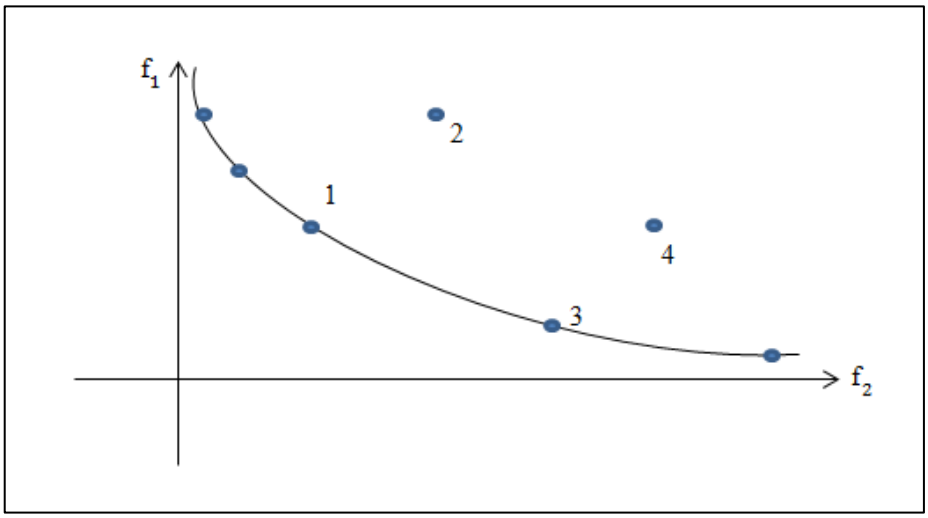
dilimi karşısında durma ihtimali aynı p olacaktır. Uygulamaya kolaylık sağlanması için Şekil 26 daki gibi rulet tekerleği yuvarlağını açarak bir doğru oluşturulur. Doğru dört kısma bölünür ve örnekleme yapılması için bir rastgele düzgün dağılımlı (0,1) aralığında şöyle r ~ u (0, 1), 0 ≤ r ≤ sayılar üretilir. Her dilimin şansı uzunluğuyla aynı olması gerekmektedir ve eşit bir şekilde bölünmesi gerekir. Başlığı 1. 11. 4 olan konuda bunlar anlatılmıştır ve rulet tekerleği *MATLAB*'da uygulanabilmektedir. Burda rulet tekerleği tekniği *Leader* seçimi için kullanılmıştır. Şebekeleme yapıldıktan sonra her hanenin içindeki depo elemanına bir *Boltzman* [55] ya da ters orantılı (*İnverse Proportional)* [53,55] ihtimali tanımlanılır. Şuna dikkat edilsin ki boş haneler yani içinde depo elemanı bulunmayan haneler hesaba katılmayacaklardır. Böylece rulet tekerleği ile bir hane seçilir ve sonra seçilen haneden bir eleman rastgele seçilmekte olacaktır ve genelde amaç *Leader* seçimi üzerinde durulmuştur. Cevapların diziliş düzeninini, çeşitliliğini (*Diversity*) korumak için popülasyonu düşük olan hanelere seçilme önceliği verilmektedir. Her parçacık hareketi rulet tekerleği tekniği ile bir hane seçerek ve *Leader*'ini belirleyerek ilerlemektedir. Yeni pozisyon ve *Personal Best*'i kıyaslamak için şöyle yapılır:

- Eğer yeni pozisyon *Personal Best*'i mağlup ederse ozaman yeni pozisyon *Personal Best*'in yerini alacaktır.
- Eğer *Personal Best* yeni pozisyonu mağlup ederse hiç birşey yapılmayacaktır.
- Eğer hiç biri birbirini mağlup etmeseler ozaman onlardan birini rastgele *Personal Best* olarak seçilmekte olacaktır.

$$P(t+1) = \begin{cases} P(t) & : p(t)\ dom\ x(t+1) \\ x(t+1) & : x(t+1)\ dom\ p(t) \\ x(t+1) \text{ ve } p(t) \text{ arasından biri rastgele seçilir: } Otherwise \end{cases} \quad (91)$$

Bu konuda rastgele eşit ihtimaller ile (0 ve 1) arasında sayı üretilir. Bu sayı 0. 5 altındaysa p(t) ve 0. 5 üstündeyse x (t + 1) seçilir ya da rastgele *binary* sayılar üretilir eğer 0 olursa p(t) aksi takdirde x (t + 1) seçilir aynı bozuk para atışı gibidir. *MOPSO*'nun çalışması için bir değişim daha yapılması gerekir. Eğer deponun elemanlarının sayısı belirlenen kapasiteyi aşarsa eleman fazlalığının kaldırılması gerekmektedir. Deponunn kapasitesi bilgisayarlarda donanım ve bellek ve hız vs. gibi sınırların olduğuna bağlıdır. Bu duruma göre eleme işleminde hangi elemanlar öncelikte olacaklar sorusunun meselesi için önceki konuları tekrar gözden geçireceğiz. Şekil 60'daki şekile göre 1 ve 3 arasında biri

elenmesi istenilirse, 1 elenecektir. Ama *leader* [53] seçilmesi söz konusu olursa 3 seçilmekte olacaktır. Eleme süreci *leader* seçiminin tam tersi olmaktadır. Bu sürec *leader* seçimine benzemektedir, Fakat ihtimaller ters olacaktır yani yetkinci koşulu tamamen ters anlamda olacaktır.



Şekil 60. Leader seçimi

- $0 \leq p_i \leq 1$
- $\sum_i p_i = p_1 + p_2 + \ldots + p_n = 1$
- $n_i \leq n_j <=> p_i \leq p_j$: Elenme ihtimali

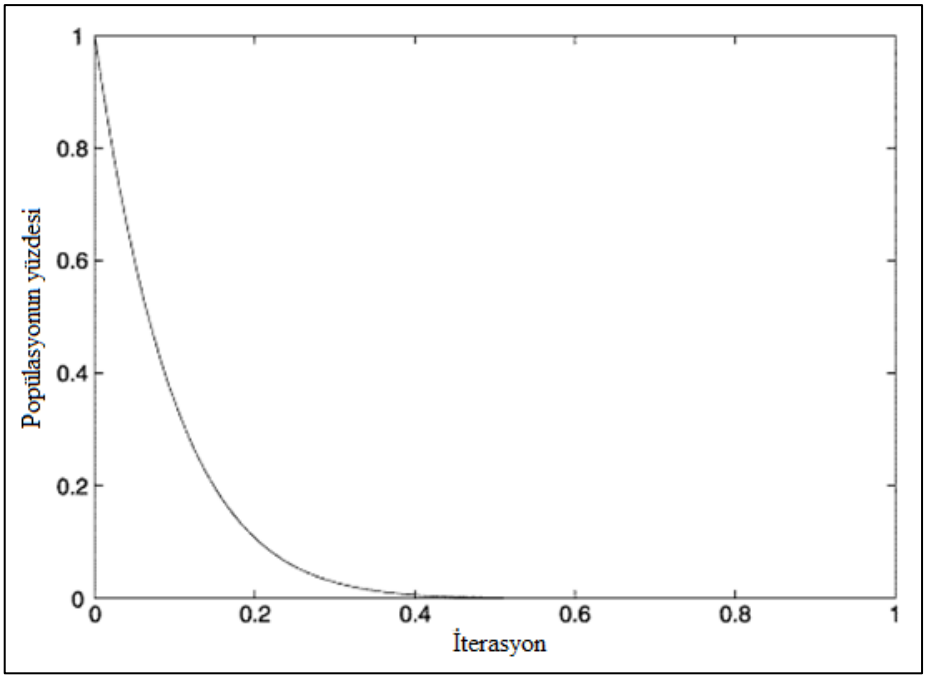
Burda eleman yoğunluğu fazla olan haneler eleme sürecinden geçmektedirler. Çünkü burda yinede önemli olan cevapların çeşitliliğinin korunmasıdır (*Diversity Preservation*) [55]. Şöyle $0 \leq q_i \leq 1, \sum_{i=1}^{n} q_i = 1$, $n_i \leq n_j <=> q_i \leq q_j$ yazılmış q_i, i'ncı hanenin elemanının elenmesinin ihtimalidir. Genede aşağıdaki gibi boltzman yöntemi kullanılmıştır:

$$q_i \; \alpha \; exp\,(\gamma \, n_i) \rightarrow q_i = \frac{e^{\gamma n_i}}{\sum e^{\gamma n_j}} \tag{92}$$

Ya da aşağıdaki gibi *İnverse Proportion* [55] yerine *proportion* Kullanılabilir:

$$q_i \; \alpha \; {n_i}^{\gamma} \rightarrow q_i = \frac{{n_i}^{\gamma}}{\sum_j {n_i}^{\gamma}} \tag{93}$$

Aynı *Leader* seçilmesi gibi hanelerden biri seçilir ve elemanlarından birisi rastgele seçilerek elenir. Bu eleme süreci deponun istenilen kapasitesi sınırına ulaşana kadar sürmekte olacaktır. Bazen eleme sürecinden sonra şebekelemeyi yenilemek gerekir. Yani *MOPSO* algoritmasının 8'inci maddesine şöyle husus daha eklenirse, *Depo elemanları belirlenen kapasiteyi aşarsa fazlalıklar elenir + şebekelemenin yenilenmesi.* Çünkü elenen elemanları algoritmanın çalıştığı esnasını ve hesaplamalarını etkileyebilir. Belkide şebekelemenin yenilenmesi gereği olmayabilir yinede bu problemin türüne bağlıdır. Bazen her 10 iterasyonda birkere yapılması söylenir ve her neyse *Coello'*nun makalesinde bundan pek açık bahsetmemiştir çünkü bu fikiri de başka yerden almıştır. *Dr. A. Coello Coello* (*Handling Multiple Objectives With Particle Swarm Optimization*) [53] makalesinde bir mutasyon operatörü kullanılmıştır. Parçacıklar popülasyonu hareketlerini yaptıktan sonra bir mutasyon operatörü bunların üzerine uygulanabilir. *PSO* algoritmanın doğası gereği yakınsama hızı oldukca yüksektir ve bu hızı biraz düşürmek için Mutasyon [53] kullanılmıştır. Algoritmanın tüm uzayı uygun biyere kadar arandığından emin olmak için özellikle çok amaçlı problemlerde uzayın her yerinde gereçekten temsilci olması gerekir ve uzayın çeşitli bölgelerinde her tür cevapların denenmesinden emin olmamız gerekmektedir. Bundan dolayı algoritmanın başlangıcında parçacıkların bir bölümüne yüksek oranda mutasyon yapılır ve sonra giderek azaltılır.

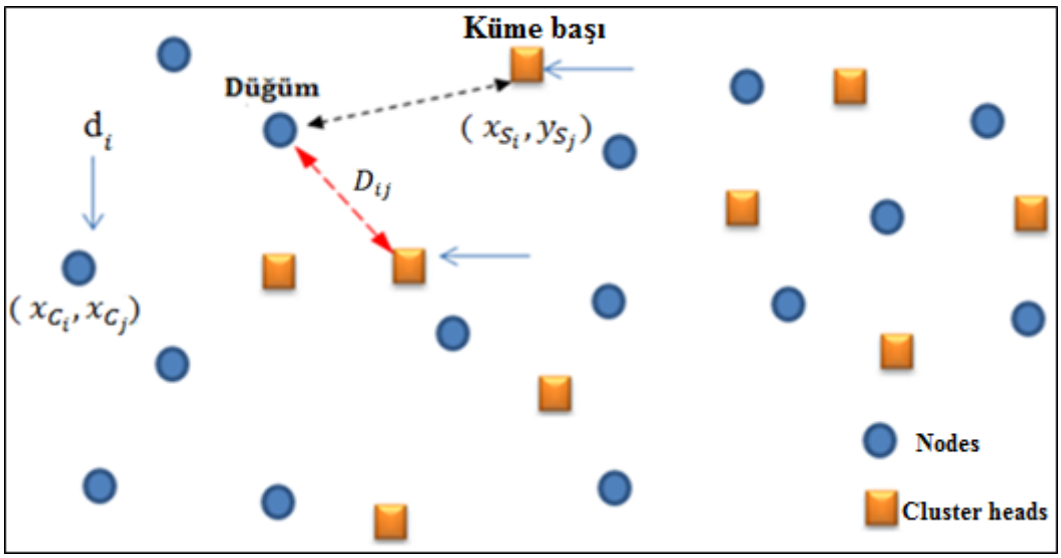


Şekil 61. *Dr. Coello'nun* makalesindeki mutasyon operatörü [53].

Makalenin Fig. 4 kısmı yukarıda Şekil 61'de de gösterilmiştir. Başlangıçta popülasyonun hepsinin üzerine mutasyon uygulanır ve kaç iterasyondan sonra popülasyonun yüzde 10'a ve sonrada giderek böyle azaltılmıştır. Buraya kadar *MOPSO* teorisi anlatılmıştır. Kalite ve Düzen kriterlerinin buradada söz konusudur. Mağlup olmayan elemanlar bulunup depoya eklendiğinde kalite esas alınır. Deponun elemanlarının elemesinde elitizmi yerine getirmeyenler kovulanlar ve bu kaliteyi ifade etmektedir. *Leader* seçiminde (Düzen) çeşitliliğe etkisi fazla olanlar önceliktedirler ve çeşitliliğe etkisi düşük olanlar elenme önceliğindedirler. *MOPSO* algoritması *NSGA-II* algoritmasına göre daha hızlıdır ve sürekli problemlerini çözmekte daha uygun cevap vermektedirler.

1.23. Problemin Tanımı

Tanımlanan problemi çözebilmek için bu çalışma sunucu Yeri Problemi (*Hub location problem*) mantığına uyumlu bir ikili (*binary*) problem olarak düşünülmüştür. Genelde sunucu Yeri Problemi çözümünde konum diye birşeyden bahsedilmemekte sadece birtakım potansiyelli yerleri açıp kapatarak sonuç elde edilmeye çalışılmaktadır. Dolayısıyla *Hub* Yeri Problemi kapsamında yer alan Kablosuz Algılayıcı Ağları Tasarımı konusu için kümeleme (*Clustering*) [55] ve aynı zamanda optimum küme başı (*Cluster head*) yeri seçimi konuları incelenmektedir. *HUB*, sunuş yapan bir servis birimidir ve *WSN* (*Wireless Sensor Networ*) alanında bunu Küme başı da temsil edebilir.



Şekil 62. Tanımlanan problemin şeması

Şekil 62'de düğümler daire ile küme başı potansiyelindeki konumlardaki düğümler ise kare ile gösterilmektedir. Düğümler *i* ile ve her düğümün küme başına yaptığı istek sayısı d_i (*demand*) ile gösterilmektedir. Amacımız düğümlerin en hızlı bir şekilde isteklerini küme başlarına ulaştırmak ve data paketleri hemen göndermektir. İstek sayısı d_i fazla olan düğümler daha çok önem kazanmakta ve onlara daha fazla ağırlık verilmektedir. Düğümlerin bir diğer dikkate alınması gereken özellikleri ise konumlarıdır. Düğümün bulunduğu konum (x_{ci}, y_{ci}) ve Küme başlarının koordinatları (x_{sj}, y_{sj}) gösterilmektedir. Bu aşamada matematikten Norm tanımı kullanılır ve uzaklıklar hesaplanarak düğümler en yakın küme başına bağlanacaktır.

1.24. Norm Tanımı

Uzaklıkları hesaplamak için *norm* [55] tanımından bir kaç yaklaşımı kullanılmaktadr.

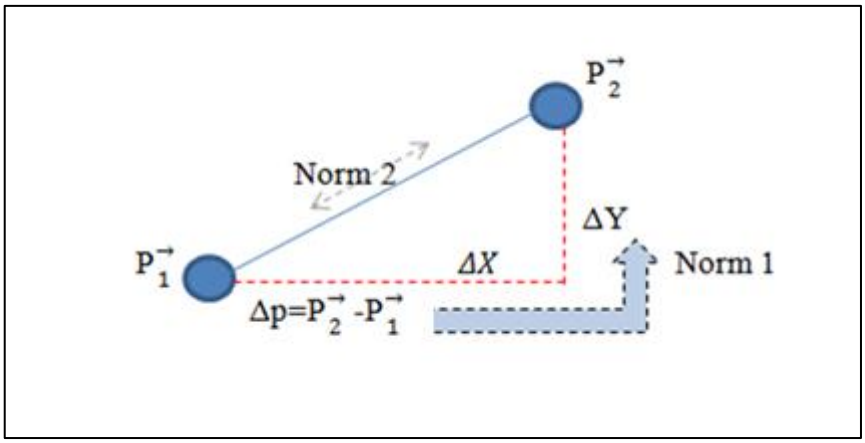
$$\|X\|P = \sqrt[p]{\sum_{i=1}^{n} |X_i|^p} \tag{94}$$

$$\|X\|_1 = |X1| + |X2| + \cdots + |Xn| \tag{95}$$

$$\|X\|_2 = |X_1| + \sqrt{(x_1^2 \ldots + x_n^2)} \tag{96}$$

$$\|X\|_\infty = \overset{Max}{i} |\mathrm{X_i}| \tag{97}$$

Yukarıdaki normun genel tanımında $P = 1$ olarak alınırsa denklem 95'deki gibi *norm*1 şehir çapında mesafe (*city block distance*) olarak adlandırılır. Eğer $P = 2$ alınırsa denklem 96'daki gibi öklidyen (*Euclidian)* [55] *Norm* olarak isimlendirilir. $P = \infty$ olursa denklem 97'deki gibi sonsuz norm elde edilir, P= ∞ durumu x'in büyüklüğünde en büyük bileşenin katkısı olması demektir. sonsuz norm $\lim_{p\to\infty} X$ problemi sayılır ve çözülür. Bu denklemler farklı problemlerde farklı şekillerde kullanılır. Aşağıda Şekil 63'de *city block distance* mantığı gösterilmektedir.



Şekil 63. Şehir çapında mesafelerin hesaplaması

Şekil 63'de $\overrightarrow{P_1}$ ve $\overrightarrow{P_2}$ noktaları arasındaki uzaklığın norm 1'e göre hesaplanması gösterilmektedir. ΔX yatay düzlemdeki uzaklık ve Δy düşey düzlemdeki uzaklıktır. İki nokta arasındaki *norm 1* mesafesi $\Delta p = \overrightarrow{P_2} - \overrightarrow{P_1}$ şekilde kalın mavi okla gösterilmiştir. Norm 2 şekilde görüldüğü gibi doğrudan iki noktayı birbirine bağlamaktadır ve *Euclidean* [55] mesafesi olarak isimlendirilmektedir. Örneğin: Boztepeden *KTÜ'ye* gitmek istersek norm 2 yolundan gidemeyebiliriz çünkü arada binalar olabilir. Yani norm 1'den yola çıkarak önce boztepe-meydana ve oradan *KTÜ'ye* gitmemiz gerekebilir. Bu çalışmada şehir çapında yani *city block distance* üzerinden mesafeler hesaplanmıştır. Yukarıdaki şekilde**,** *i*'ncı düğüm ve *j*'nci küme başı arasındaki uzaklıktır. Aşağıda amaç fonksiyondaki bilinmeyen değişkenler şöyledir:

- $F_j = 0 \implies$ Küme başı düğümü kapalıdır
- $F_j = 1 \implies$ Küme başı düğümü açıktır

Tanımlanan problem bu sebeple bir ikili (*Binary*) problemi türünden sayılmaktadır. Her düğüm kendine en yakın küme başını bulurken city block distance'a göre D_{ij}'leri hesaplar ve aşağıdaki gibi D_i^{min} yani düğümlerden en yakın açık küme başı bulur.

$$D_i^{min} = Min\{D_{ij} \mid Fj = 1\} \tag{98}$$

$$D_i^{min} = \frac{D_{ij}}{Fj} \tag{99}$$

Yukarıdaki ifade daha doğru çalışır çünkü F_j ya 0'dır yada 1, eğer 0 olursa ifade çıkışı ∞ olur dolayısıyla hesaba alınmaz başka bir deyişle eğer bir server kapalı var sayılırsa bu server ∞ konumunda demektir. Amaç fonksiyonu ifadesindeki birinci terimi yani birinci *Objevtive*'de düğümlerin istek sayısının D_i^{min} mesafeleri ile birlikte minimize ediliyor. Ayrıca ikinci terimde de düğümlerin açık tutma enerjisi göz önünde bulundurlarak minimize edilecektir. İsteği fazla olan düğümlerin uzaklığı bizim için daha çok önem taşımaktadır. Amaç fonksiyonunu aşağıdaki denklem 100'daki gibi tanımlanmaktadır.

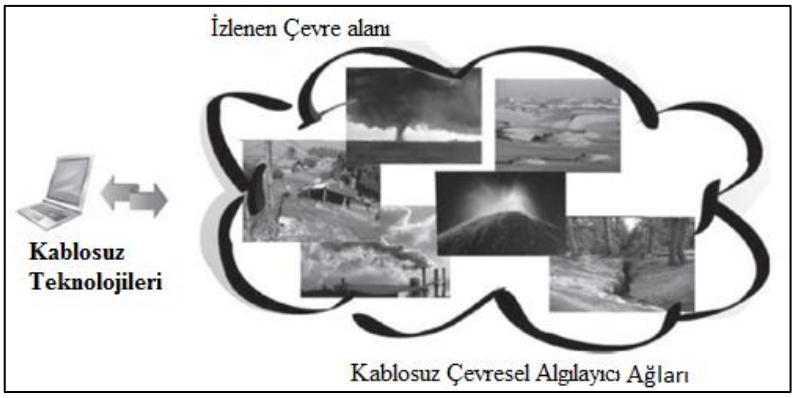
$$Min\ Z\ =\ \sum_{i=1}^{n} d_i.\ D_i^{min} + \sum_j F_j.C_j \qquad (100)$$

- d_i = Düğümlerin Küme başına gönderdiği istek sayıs ya da müracaat sayısı
- D_i^{Min} = Düğümlerden en yakın açık Küme başı mesafesi
- Fj = 0 (*On*) Ya 1 (*Off*)
- C_j= Düğümlerin açık tutma (*Set up or Activation Energy of Node*) enerjisinin maliyeti

1.25. Senaryo

1.25.1. Akıllı Çevre İzleme

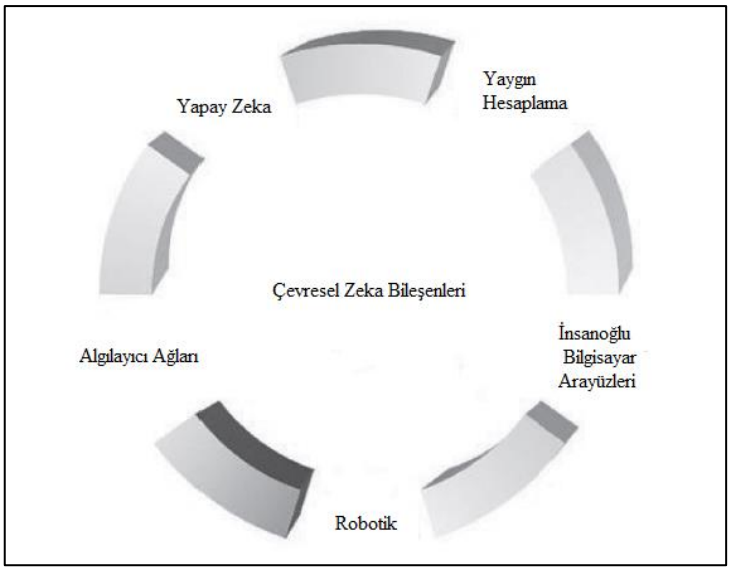
Akıllı Çevre İzleme (*Smart Environmental Monitoring*) [56] konusu günümüzde kablosuz teknolojilerin gelişmesiyle ileri sürülmüştür. Şekil 64'da gösterildiği gibi otomasyon sistemlerinde hasta, yaşlı ve özürlülerin takibini sağlayan, yaşamı kolaylaştıran, kablosuz cihazların yer aldığı akıllı ortamların oluşturulması yönünde ilerlemeler başlamıştır. Kablosuz biyolojik ve çevresel Algılayıcı'lar, akıllı uzaktan izleme sistemleri, sağlık bakım uygulamaları ve hasta takip sistemlerinin birleştirilmesiyle güvenilir ve gerçek zamanlı birçok uygulama geliştirilmiştir. Teknoloji ve hizmetlerin bütünleştiği akıllı sistemler toplumun gereksinimleri doğrultusunda çalışmalarını sürdürmektedir.



Şekil 64. Akıllı çevre izleme

1.25.2. Çevresel Zeka

Çevresel zeka (*ambient intelligence*) [57] çeşitli elektronik cihazlar (algılayıcılar, mobil cihazlar vb) yardımıyla kullanıcılara akıllı ortamlar sağlar. Bir ortamda bulunan bu cihazların birbirleri ile uyumlu olarak çalışıp, zeki bir sistem oluşturması amaçlanmaktadır. Çevresel zeka yaygın bilişim üzerine oturan çeşitli yapay zeka (*Artificial İntelligence*) metodlarını destekleyen çok disiplinli kullanıcı merkezli yeni paradigma olarak tanımlanır. Çevresel zeka ortamları, hazır ve nazırdır (*Ubiqutious* türkçesinin çevirisi zor olan bir kelimedir), teknolojik araçlar arka plana entegre edilmiştir. Sistem çevrede yaşayan insanları tanıyabilir ve onların davranışlarını öğrenerek kendi kendine ihtiyaçları karşılayabilir, akıllıdır. Şekil 65'da çevresel zeka sisteminin bileşenleri gösterilmiştir.

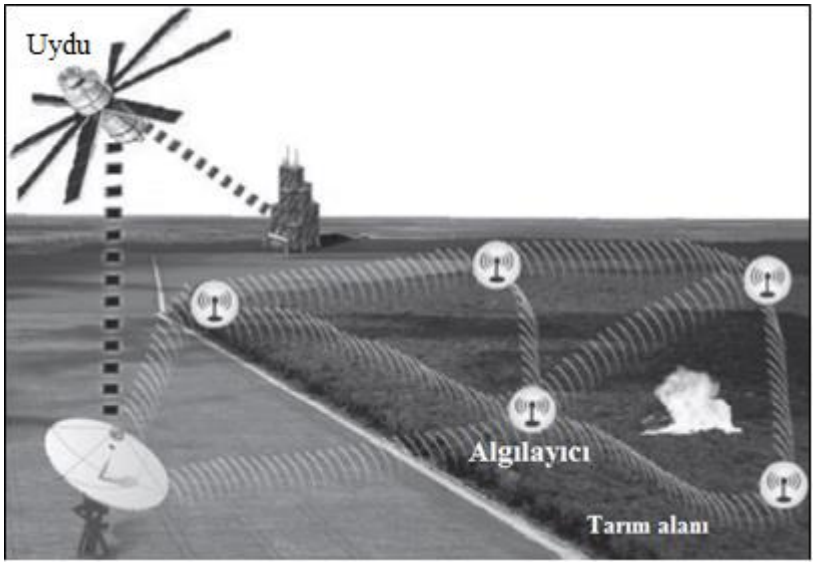


Şekil 65. Çevresel zeka bileşenleri

1.25.3. Hassas Tarım

Bu çalışma çevresel zeka bileşenlerinden yapay zeka ve algılayıcı ağları konuları kapsamaktadır. Akıllı Çevre İzleme ve Çevresel Zeka konuları incelenmesi açısından oldukca geniş bir alandır. Dolayısıyla hassas tarım (*precision farming*) [58] konusu gibi uygun bir iki boyutlu (*Two Dimensional*) Problemini baz alarak yola çıkıyoruz. Uzun yıllardır bilişim sektörünün ilgi alanı dışında kalmış olmasına karşın, son yıllarda ve özellikle gelişmiş ülkelerde bilgi teknolojilerinin gelişimiyle insana, bitkiye, hayvana ve çevreye duyarlı, üretimde kalite ve verimlilik faktörlerini ön planda tutan bir evrim geçirmektedir. Tarımsal üretimde insan gücünden hayvan gücüne ve daha sonra da traktör gücüne geçiş sürecinin devamı olarak değerlendirilen ve hassas tarım (*precision farming*) olarak adlandırılan teknolojiler de bu evrim süreciyle ortaya çıkmıştır. Hassas tarım, ekonomi ve çevre koruma ilkelerini göz önünde tutarak bilişim çağının gelişen teknolojilerinin tarımsal üretimle bütünleştirilerek kullanılmasını ifade etmektedir. Hassas tarım sisteminde kullanılmak üzere örnek toprağın içerdiği bitki besin elementlerinin ve pH düzeyinin ölçülmesi amacıyla kablosuz Algılayıcı'lar geliştirilmiştir. Bu algılayıcı'ların çalışma prensipleri aşağıdaki gibi özetlenebilmektedir :

- İyon seçici elektrot (*ISE*)
- İyon seçici alan etkili transistor yöntemi (*ISFET*)
- Gama ışını yansıtma,
- Kızıl ötesi ışın yansıtma,
- Elektriksel direnç.



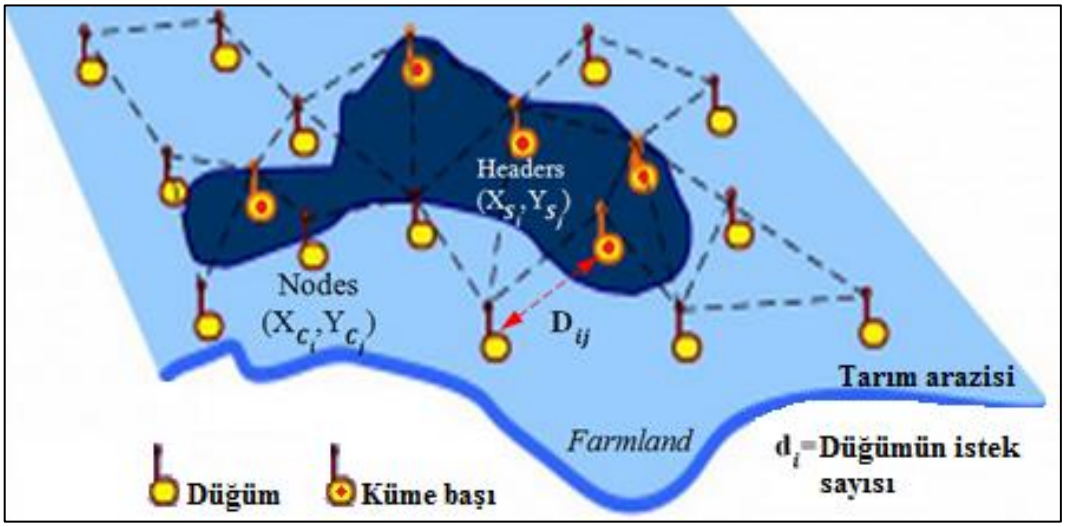
Şekil 66. Algılayıcıların tarım alanında yerleştirilmesi

Şekil 66'de görüldüğü gibi Algılayıcı'lar bir karesel alanda dağılmışlardır. Senaryoda inişler ve çıkışlar göz ardı edilmiştir. Ayrıca düğümlerin tüm koordinatları elde ve biliniyor var sayılmıştır. İncelemeler sadece işin Kablosuz Algılayıcı Ağları üzerinde durulmuştur. İlk başta düğümlerden belli sayıda Küme başı olmak üzere seçilerek senaryodaki alana dağıtılıyor. Normal düğümlerde aynen belli sayıda bahsi geçen alana dağıtılır. Optimum Küme başının yeri seçimi için senaryoda düğümlerin açık tutma (*Set up*) enerjisinin maliyeti ve Küme başları ve düğümler arasındaki mesafeler ve düğümlerin küme başlarına gönderdiği istek sayısı gibi etkenleri problemin çözümünde göz önünde bulundurulmuştur. Çalışmada yapılan önemli bir husus daha kümeleme (*Clustering*) olduğu da vurgulanmaktadır. Yani sedece optimum küme başlarının yerleri seçilmiyor. Dolayısıyla çalışmanın benzetim programının girdisi değişirse kümeleme yeni duruma göre yapılacaktır. Düğümlerin koordinatları rastgele olarak'da dağıtılabilmektedir ve yapılan çalışma dinamik bir senaryoya dönüşme kabiliyetine sahiptir. Ayrıca bu problemin çözümü ile enerji tasarrufu ve ağ ömründe artış sağlanmış olacaktır

2. YAPILAN ÇALIŞMALAR

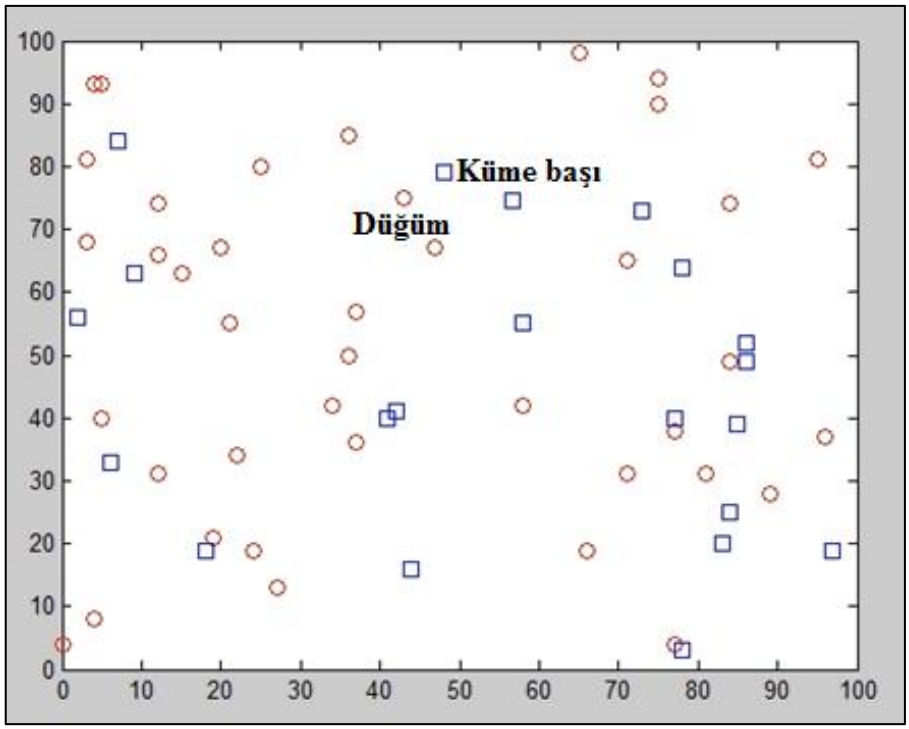
2.1. Problemi Radyo İletişim Modeli ile Matlab Uygulaması

Algılayıcı'ların tarım alanında yerleştirilmesinin şeması, çalışmaya daha iyi bir görsellik katılması için Şekil 67'de gösterilmiştir.



Şekil 67. Problemin görsel şeması

Şekil 67'deki tarım arazisinin senaryosunun benzetimi için *MATLAB* programı kullanldı. Benzetimde düğümlerin sayısı 40 ve küme başlarının sayısı 20 tane olarak alındı. Buradaki 20 tane aslında olabilecek en yüksek küme başı sayısı anlamına gelmektedir. Buradaki 20 tane küme başı açma (*Set up*) potansiyeline sahip olan yer koordinatı anlamına gelmektedir. Bu koordinatlar 100*100 kare alanında rastgele dağıtılmıştır. Yani 20 tane x ve y koordinatı küme başları için benzer şekilde 40 tane x ve y koordinatı da diğer düğümler için 0 ile 100 arasında rastgele üretilmektedir.



Şekil 68. Matlab ortamında problemin şeması

İstekler (d_i) her düğüm için 5 ve 50 arasında rastgele düğümlerin sayısının buyutunda bir dizide üretilip ve saklanmıştır. *MATLAB* ortamında *model* adında bir *structure* tanımlanmıştır ve *model* içinde aşağıdaki değerleri barındırmaktadır

.

>> model=CreateModel()

model =

N : 40 : Düğümlerin sayısı

M : 20 : Küme başının sayısı

d : [1x40 double] : İsteklerin sayısının dizisi

xc: [1x40 double] : Düğümlerin x koordinatının dizisi

yc: [1x40 double] : Düğümlerin y koordinatının dizisi

c : [1x20 double] : Düğümlerin aktivasyon Enerjisinin dizisi

xs: [85 78 18 9 86 86 7 44 73 77 41 6 42 58 78 48 84 83 2 97]: Küme başının x koordinatının dizisi

ys: [1x20 double] : Küme başının y koordinatının dizisi

c= [11243 8647 11029 8745 10192 10921 10411 9060 10430 8635 8127 11813

9090 10691 8081 8459 10938 11797 11954 9444] : Düğümlerin aktivasyon

Enerjisinin değerlerinin dizisi

D: [40x20 double] : Hesaplanan tüm mesafelerin dizisi

Model değişkeni oluşturulduktan sonra rastgele çözüm (*Kromozom*) üretme aşamasına gelinir. Hangi küme başının açık ve hangi küme başının kapalı olduğunu rastegele belirlemek için bir *f=createrandomsolution*(*model*) isimli bir fonksiyon tanımlanmıştır. Problemin akış diyagramı Şekil 70'da gösterilmiştir. Bu fonksiyonun görevi rastgele çözümler üretmesidir ve probleme göre 20 tane rastgele 1*20 boyutunda (0,1) random değerleri üretiliyor. Üretilen her çözüm kromozom olarak adlandırılmaktadır. Böylece ikili genetik algoritmaya (*Binary-GA*) göre popülasyon sayısı (n_{pop}) kadar şöyle kromozom oluşuyor :

F_1 = [0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 1 1 0]

F_2 = [0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0]

.

.

.

$F_{n_{pop}}$ = [0 0 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1]

Model ve *CreateRandomSolution* (Rastgele Çözümler = Kromozomlar = F) ürettikten sonra çözümler denklem 101'de *Cost Function* isimli değerlendirme prosedürüne gönderilmektedir.

$$Amaç\ Fonksyonu \rightarrow Min\ Z\ = \sum_{i=1}^{n} d_i . D_i^{min} + \sum_j F_j . C_j \qquad (101)$$

Küme başlarının açık tutma yada kuruluş maliyeti hesaplanırken 5 ile 50 arasındaki istek (*demand*) sayısısının maksimum değeri alınır ($d_i = 50$). Diğer taraftan problemin alanı 100*100 olduğundan düğüm ile küme başı arasındaki en uzun mesafe kare çapı yani $100\sqrt{2} \cong 141$'dir ve aslında $141x50$ bir düğümün maksimum maliyeti sayılır. Dolayısıyla bir düğümün maksimum maliyeti 7050'dir. Ayrıca bir düğümün ortlama maliyeti de 3525 olur. Böylece 3525*40 =141000 tüm düğümlerin maliyeti var sayılır. Küme başı sayısı 20 tane olduğuna göre 141000/20 = 7050 tüm küme başlarının aktivasyon maliyeti 7050

etrafından biraz fazla tutularak ve programda kullanmak üzere 20 tane rastgele maliyet toleranslı bir şekilde 8000 ile 12000 arasında rastgele üretilip bir dizide saklanır. Ayrıca küme başlarının açık tutma enerjisini daha gerçek bir model içerisinde hesaplamak için *Heinzelman* et al'ın [11,62] Radyo İletişim Modeli (*Radio Communication Model*) kullanılmıştır. [62]'de açıklandığı gibi burada bir radyo modeli kullanılmıştır. Küme içerisindeki düğümler için kısa mesafe iletimi d^2 (*distance*) ile orantılı bir şekilde ve daha uzun mesafe iletimi d^4 ile orantılı olarak aşağıdaki gibi ifade edilmiştir.

$$Ech\_elect = l * (Ee * (n/k) + Ebf * ((n/k) - 1) + es * Distance^{2}) \quad (102)$$

Denklem 102'deki seçim aşamasında (*Election Phase*) tüm düğümler birbiri ile haberleşerek her düğüm en yakın küme başına bağlanır ve bu problemin D_i^{min} aşamasında gerçekleşmektedir. Dolayısıyla k sayısında küme başı oluşmuştur ve her kümenin içinde m düğüm bulunmaktadır. *WSN* ağının tüm düğümleri ise n ile gösterilmektedir. Ee, Ebf ve es değerleri aşağıda gösterilmektedir [62]:

Tablo 1. Radyo iletişim modelin niceliksel analizinin örnek parametre değerleri

Tanımlama	Sembol	Değerler
Kısa mesafelere iletmek için yükselteç tarafından tüketilen enerji	ϵ_s	10 pJ/bit/m^2
Uzun mesafelere iletmek için yükselteç tarafından tüketilen enerji	ϵ_l	0.0013 pJ/bit/m^4
Elektronik devre 'de enerji tüketimi	E_e	50 nJ/bit
Işın şekillendirme için tüketilen enerji	E_{BF}	5 nJ/bit

$$Enon_{ch_{elect}} = l * (Ee * (1 + k) + (k * Ebf) + es * ((M * M)/((2 * 3.14159) * k))) \quad (103)$$

Küme başı olmayan düğümlerin enerji tüketimini ise (*non-cluster head*) 103 ifadesi ile hesaplanmaktadır. Veri transferi (*Data Transfer Phase*) aşamasında düğümler topladığı

verileri küme başlarına iletmektedirler ve bunun enerji tüketiminin maliyeti her küme başı için denklem 104'da şöyledir.

$$Ech_{frame} = l * (((n/k) - m) * Ebf + (el * Distance^{4}) + ((n/k) - m + 1) * Ee) \quad (104)$$

$$f1 = 1/(k * ((n/k) - m + 1)) \quad (105)$$

$$f2 = ((n/k) - m)/(k * ((n/k) - m + 1)) \quad (106)$$

$$Ech\_data = f1 * Nf * Ech\_frame \quad (107)$$

$$Enon\_ch\_data = f2 * Nf * Enon\_ch\_frame \quad (108)$$

Her iterasyon sayısında tüketilen enerji, seçim ve data transferi aşamalarının toplaması ile hesaplanmaktadır.

$$Ech\_iter = Ech\_elect + Ech\_data \quad (109)$$

$$Enon\_ch\_iter = Enon\_ch\_elect + Enon\_ch\_data \quad (110)$$

Bu çalışma *LEACH* protokolü gibi iteratif mantığı ile çalışmaktadır. Dolayısıyla her iterasyonda tüketilen düğümüm *start* enerjisi denklem 111'deki gibi hesaplanmaktadır :

$$Estart = (Ech\_iter/m) + ((n/(k * m)) - 1) * (Enon\_ch\_iter/((n/k) - m)) \quad (111)$$

Enerji =[11243 8647 11029 8745 10192 10921 10411 9060 10430 8635 8127 11813 9090 10691 8081 8459 10938 11797 11954 9444] + Model

Böylece önceki hesaplanan maliyetlere bu yeni gerçek hesaplamaları katarak problem minimize edilmiştir. Tüm bunlara göre amaç fonksiyonu aşağıdaki gibi 112'de tanımlanmaktadır:

Amaç fonksyonu → $Min\ Z\ =\ \sum_{i=1}^{n} d_i . D_i^{min} + \sum_j F_j . Energy_j$ (112)

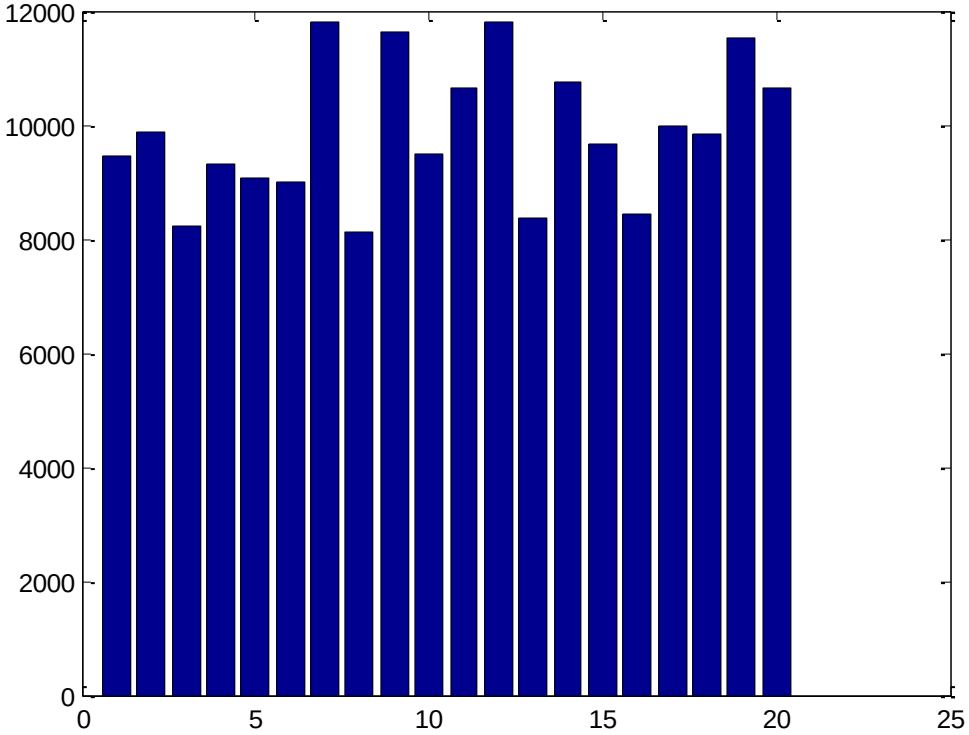
Program radyo iletişim modeli ile bir iterasyon için aşağıdaki gibi koşturulmuştur: n düğüm sayısı: 40, c düğümlerin kuruluş maliyetleri programın iterasyonunda rastgele 8000 ve 12000 değerleri arasında üretilmektedir. Böylece c matrisine radyo iletişim modelinin tüketilen enerjisi hesaplanarak eklenmiştir. Bu matris sürekli değişmektedir.

c =[11259 11624 8508 11654 10530 8390 9114 10188 11830 11860 8630 11883 11829 9941 11201 8567 9687 11663 11169 11838]

Küme sayısı: k = 11, data frame sayısı: *Nf* =10000 ve *Distance* = 28 değeler ile enerji şöyle Tablo 2'de gösterildiği gibi raporlanmıştır:

Tablo 2. Radyo modeli ile düğümlerin kuruluş enerjilerinin maliyetleri

Energy = Estart + c ⇒ $Energy_{1*20}$ =			
[11259.0882950223	11624.0882950223	8508.08829502227	11654.0882950223
10530.0882950223	8390.08829502227	9114.08829502227	10188.0882950223
11830.0882950223	11860.0882950223	8630.08829502227	11883.0882950223
11829.0882950223	9941.08829502227	11201.0882950223	8567.08829502227
9687.08829502227	11663.0882950223	11169.088295022	11838.08829502]



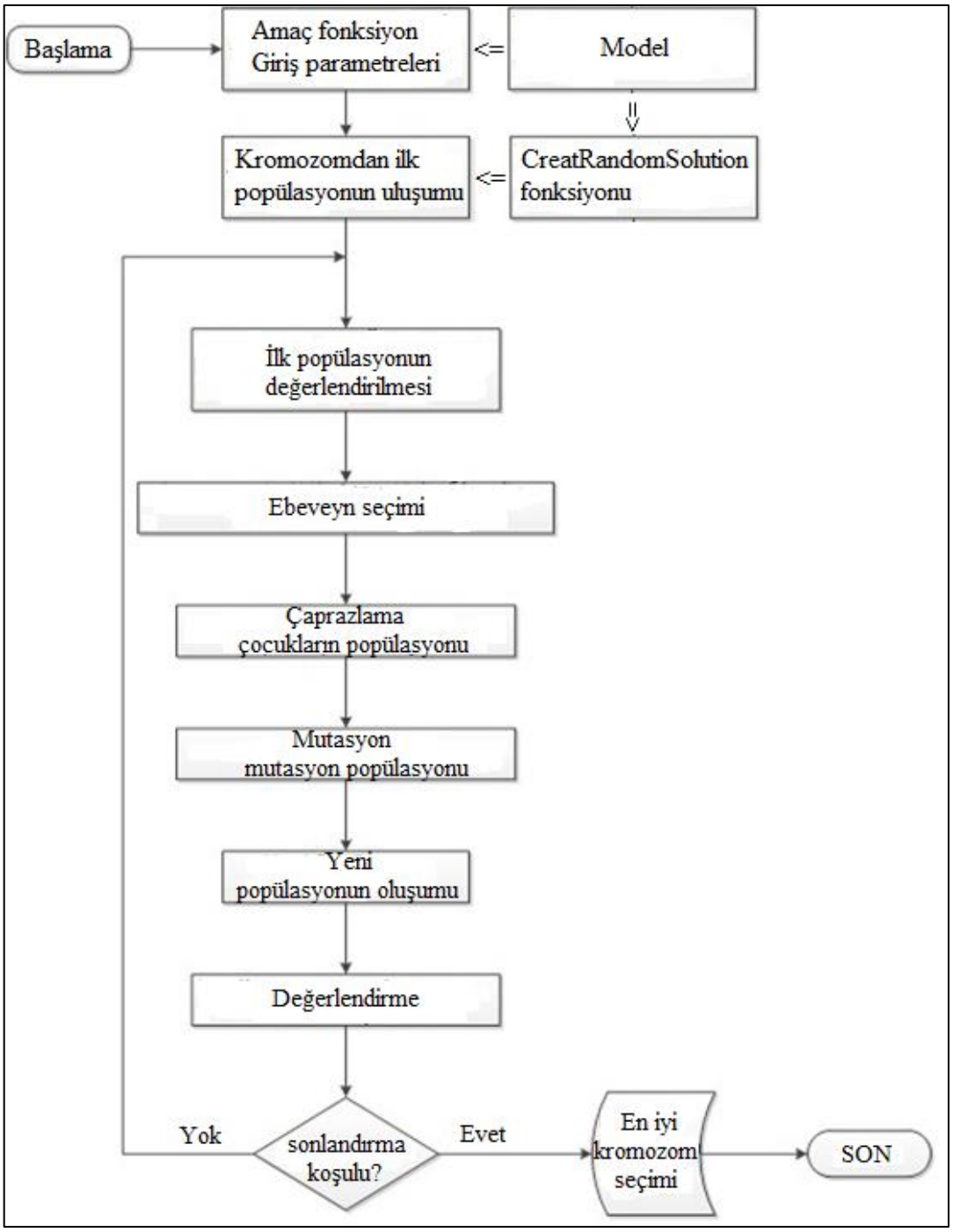
Şekil 69. Her iterasyonda sürekli değişen enerji maliyetleri

Şekil 69'de görüldüğü gibi küme başının sayısı 20 tanedir ve yaklaşık 8000 ve 12000 arasında değişmektedir. Radyo modeli ile düğümlerin kuruluş maliyetleri yani enerji matrisini senaryoyu çok kötü bir şartlarının olmasını düşünerek enerjiyi sürekli değişen bir dizi şeklinde ifadesi 112'deki amaç fonksiyonuna verilmiştir. Genelde ancak bir simulasyon çok kötü şartlar içerisinde uygulanır. Bu çalışmada enerji maliyetlerinin sürekli değişmesine rağmen algoritma problemin pareto'suna iyi bir yakınsamayla çözüme ulaşmıştır. Yukarıdaki enerjinin çubuk grafiği (*bar graph*) zamanın geçmesi ve itersyonun artmasıyla sürekli değişmektedir.

2.2. Problemin Tek Amaçlı Yaklaşımının Akış Diyagramı

Şekil 70'de akış diyagramında ilk başta oluşturduğumuz model fonksiyonundan parametreler çıkarılır ve sonra amaç fonksiyonu tanımlanır. Sonra modelden çıkarılan bilgiler CreatRandomSolution fonksiyonuna verilir ve kromozomlar oluşur. Sonrada üretilen koromozomlardan algoritmanın ilk popülasyonu oluşturulur. Oluşturulan ilk popülasyon maliyet fonksiyonuna girerek değerlendirme sürecinden geçer ve daha sonra

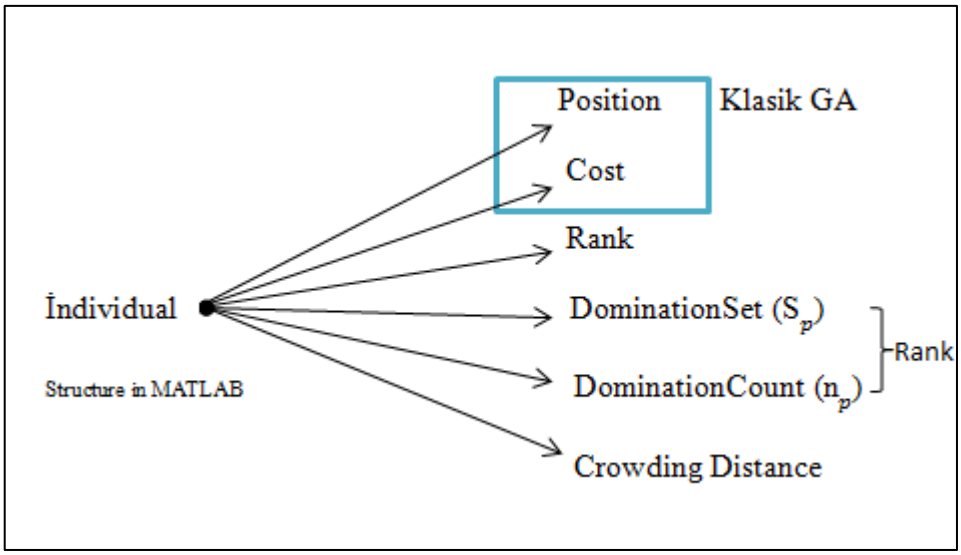
algoritmanın iteratif döngüsüne girip seçim işlemleri, çaprazlama, mutasyon, sıralama ve kesme sürecinden geçerek sonunda en iyi kromozom ya da optimum çözüm elde edilir.



Şekil 70. Problemin ağırlıklı toplam genetik algoritması ile çözümü için akış diyagramı

2.3. Problemin NSGA-II Algoritma ile Çözümü

Tezin önceki kısımlarında *NSGA-II* teorisi üzerinde konuşuldu ve tek amaçlı genetik algoritma çok amaçlı algoritmaya dönüşmesi için nelerin değişilmesi gerektiği anlatıldı. Görüldüğü gibi bu değişimler hep seçim operatörleri üzerindedir çünkü iyi ve kötü cevapların teşhisinde artık kriterler değişmiştir. Normal genetik algoritmasında sadece bir kriter vardır. Ama burda 2 yada 3 kriter olabilir ve bu kriterler problemin *objective* sayısına göre farklı olabilmektedir. Bu kriterler kalite ve düzen adı altında bahsedilmiştir. (*evolutionary algorithms for solving multi-objective problems*) [42] kitabın yazarı *dr. Carlos A. Coello Coello ve emoo repository* web sitesi [60] *multi objective* havzasında araştırmak için uygun ve kapsamlı kaynaklardır. Şimdi tüm anlattıklarımıza dayanarak kablosuz algılayıcı ağlarındaki küme başı yeri problemini çok amaçlı olarak *NSGA-II* algoritması ile incelemekte ve çözmekte olacağız. Daha önce çözdüğümüz problem iki (z_1 ve z_2) amaç fonksiyonu olan çok amaçlı problemdir. Problem ikili genetik algoritması, tek amaçlı yaklaşımı *weighted sum* dekompozisyon yöntemi ile çözülmüştür ve farklı $w_1$ve w_2 ayarlarıyla pareto'dan farklı çevaplar elde edilmiştir. Kullandığımız yöntem *pareto*'yu doğrusal olarak yaklaşık tahmin etmiştir. Yapılanları toplarsak konuyu geliştirme doğrultusunda dezavantajları söylemek gerekir. Şöyle tezin ilk kısmında kullandığımız *B-GA* (*binary genetic algorithm*) ile (*weighted sum*) yöntemi *pareto*'nun içbükey (*concave*) kısımlarını keşfedebilmemiştir. Ayrıca algoritma her koştuğunda *pareto*'dan yalnız bir noktayı cevap ünvanında bularak çözmüstür. Tüm noktaları bulması için algoritmanın defalarca farklı *setting* ya da *w*'lar ile koşturulması gerekir. Elde edilen sonuçlar kesin olmayan, zaman alıcı ve yaklaşık cevaplardır. klasik genetik algoritmasında popülasyondan her eleman (*individual*)'ın [55] *position* ve *cost* adında bileşenleri vardır, Ama *NSGA-II* bunlarla sınırlı değildir.



Şekil 71. Yapılanmış her bireyin bileşenleri

Programda her bireyin *Position* ve *Cost* bileşenlerinin Yanısıra rütbe (*Rank*), galip durumunun dizisi (*DominationSet*), mağlup durumunun sayacı (*DominationCount*), yoğunluk mesafesi (*Crowding Distance*) Şekil 71'de gösterildiği gibi programlanmıştır. İlk başta bireyler (*İndividual*) başlatılır (*İnitialization*). Sonra *Position* başlanır, *Position* bir bağımsız özelliktir ve diğer bileşenler ona bağlı olarak belirlenir. *Cost, Position*'a bağlı ve diğerler *Cost*'a bağlı olmaktadırlar. *Position*'ler *CreateRandomSolution(model)* fonksiyonun çıkışının ürettiği değerlr ile doldurulur.

% Cost Function

z_1=sum(d. *Dmin) : *Objective* 1 (f_1)

z_2=sum(f. *c) : *Objective* 2 (f_2)

w_1=model. w_1

w_2=model. w_2

z=w_1*z_1+ w_2*z_2 : *Weighted Sum*

Artık yukarıdaki w_1 ve w_2 ağırlık katsayılarına gerek yoktur. Amaç fonksiyonları doğrudan algoritmaya verilir. z_1 ve z_2 (2*m) boyutlu bir matriste kayıtlanıyor yani z_1 değerleri birinci satırda (1, :) ve z_2 derğerleri ikinci satırda (2, :) kayıtlanıyor. $f_1$ve f_2 *Objective Function*'ları amaç fonksiyonun uzayında (*Fenotip*) *NSGA-II* algoritmasına girmeden önce grafiği çizilmiş ve şekil Şekil 72' de gösterilmiştir.

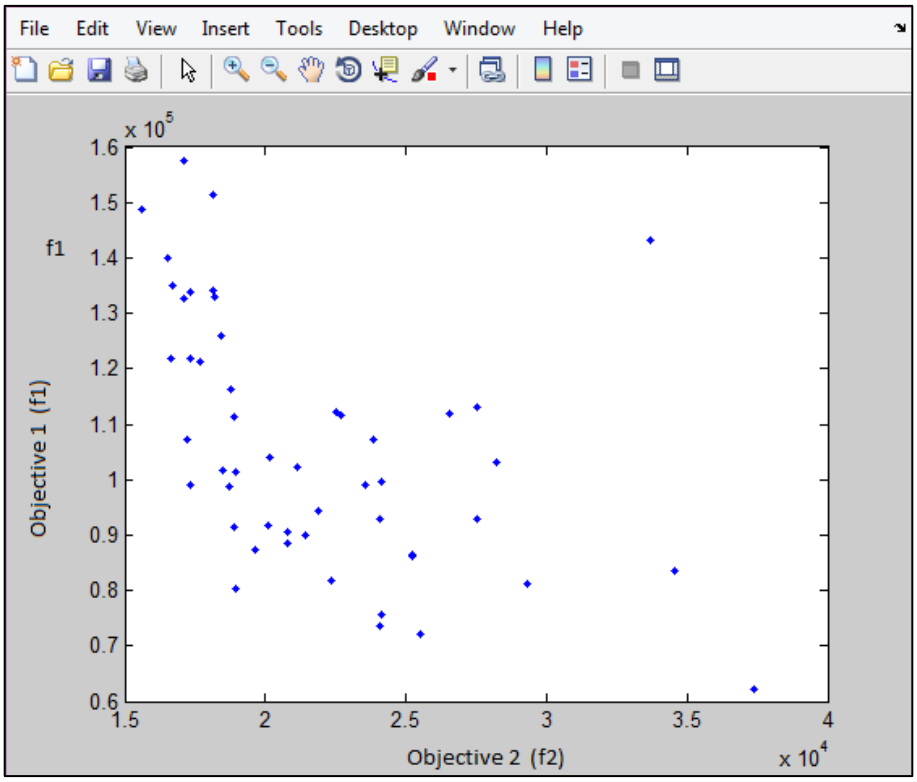
Burda amaç fonksiyonun değerleri ilk popülasyonun 50 bireyi için çıkarılıp şöyle raporlanıyor. >> costs=pop. Cost, >> costs=[pop. Cost] :

Tablo 3. Problemin *NSGA-II* yaklaşımında ilk popülasyonun 50 bireyinin maliyet değerleri

costs = Columns 1 through 50 : 17701 29326 20132 25220 18915 121160 81106 104019 86274 111402
23837 15617 24056 16674 18927 107119 148796 92876 134858 80289
24142 17129 22543 25500 37397 75589 132750 112104 72245 62286
18943 17329 20775 20106 24046 101376 121915 90682 91803 73488
27539 16545 17094 19651 23558 113127 139989 157376 87293 99124
18784 21860 18908 25208 20790 116226 94230 91297 86514 88578
17334 21412 18167 18114 17192 133921 89871 151483 134070 107329
26555 27528 22663 18476 34544 112042 92846 111705 101557 83576
33683 22372 24129 18714 18221 143179 81690 99516 98656 132888
18456 21158 16618 28199 17324 125863 102313 121984 103194 98911

Bu aşamada amaç fonksiyonların değerlerini fenotip alanında nasıl bir durumda olduklarını görmek için şöyle çizilmiştir.

>> plot (costs(1, :), costs(2, :), '. ')



Şekil 72. Fenotip alanında cevapların ilk durumları

Şekil 72 grafiğinde şunu görmekteyiz ki cevapların ilk durumları amaç fonksiyonu alanında (*Fenotip*) dağınık ve istenilmeyen şekildeler. Algoritmanın marifeti bu cevapları ideal *Pareto Front*'a ulaştırmasıdır. Mağlup olmayan cevapların bulmasını gerçekleştimek için sıra *Rank* kriterini belirlemektedir. *DominationSet* ve *DominationCount* tek başına işe gelmezler. burda önemi olan *Rank* [42] ve *Crowding Distance* [42] kriterleridir, *DominationSet* ve *DominationCount* etkenleri *Rank*'ı belirlemekte destek faktörleri sayılmaktadırlar. Bu işlemler *NS* (*Non-Dominated Sorting*) [42] algoritmasında şöyle gerçekleştirilmektedir.

2.4. Mağlup Olmayan Sıralama (*Non-Dominated Sorting*) NS Algoritma

MATLAB'da mağlup olmayan sıralama algoritması *NonDominatedSorting. m* adında dosyaya programlanmıştır. Aşağıdaki *NS* [42] algoritmanın işlem sıralarıdır :

- P aracılığıyla mağlup olan popülasyon elemanların kümesi → *DominationSet* (S_p)
- P'nin diğerlerın aracılığıyla mağlup olma sayısı → *DominationCount* (n_p)
- Tüm popülasyon elemanları için $S_p = \{\}$ ve $n_p = 0$ değerleri alınır.

- P gibi popülasyondan her eleman için ve q gibi popülasyondan her eleman için :

Eğer p, q'ya galip gelirse → q elemanını S_p'ye ekle.

Eğer q, p'ye galip gelirse → Bir birim n_p'yi arttır.

Bu aşama aşağıdaki *Dominates* fonksiyonu ile şöyle gerçekleşmektedir.

function b=*Dominates*(x,y)

b=all(x<=y) && any(x<y); % Evrensel Nicelik ve Varoluşsal Nicelik

- Tüm n_p'si sıfır olan n_p= 0 popülasyon elemanlarını F_1'e ekle.
- Cephelerin sayacını K=1 değerine eşitle.

☼ Q'yu F_{k+1}'in taslağı olarak oluştur.

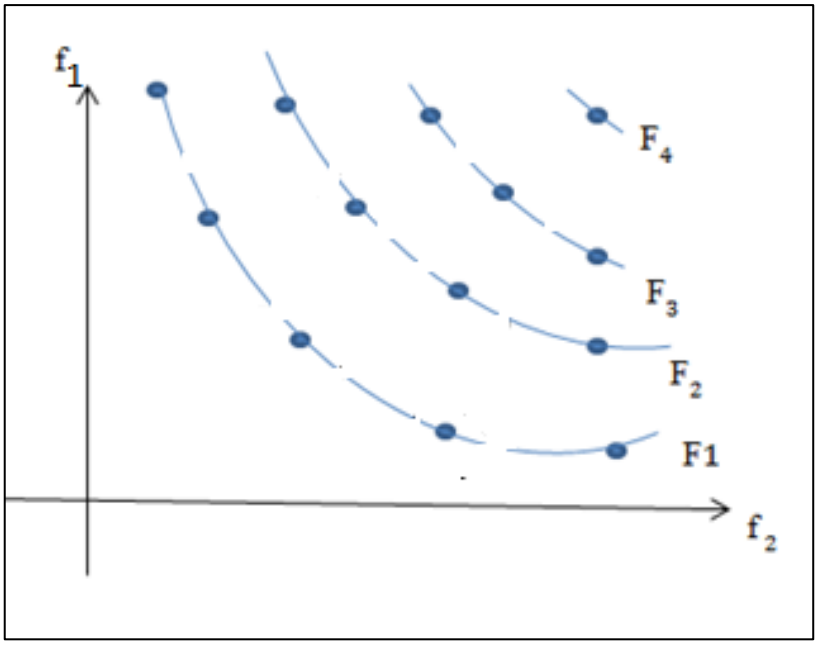
- F_k'dan her eleman için P gibi:

S_p'den her eleman için q gibi (P aracılığıyla mağlup olan tüm q'lar) n_q'dan bir birim eksilt.

- Eğer n_q= 0 olursa q'yu Q'ya ekle. Eğer Q boş olursa sıralama işlemi sona ermiştir.

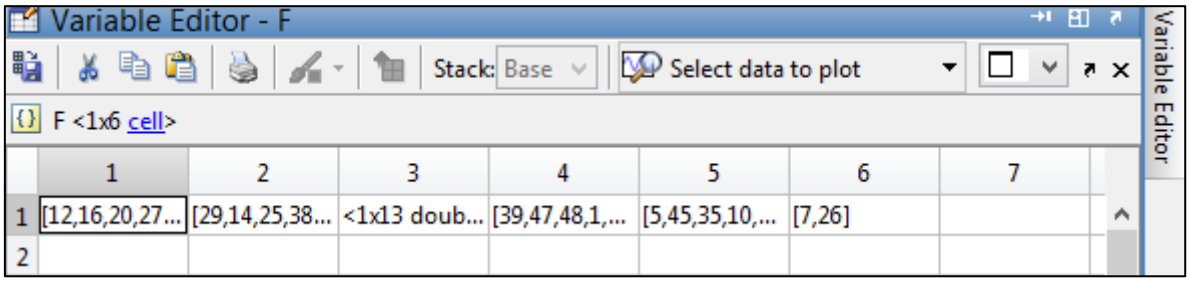
Eğer Q Boş olmadıysa F_{k+1}'ri Q ile eşitle.

- Bir birim K sayacına arttır ve ☼ aşamasına geri dön.



Şekil 73. Cephelerin oluşumu

NS algoritmasının sonucunda *Rank*'lar belirlenir ve *Front*'lar *Rank* ile uyumlu olarak Şekil 73'deki gibi sıralanır. Burda farklı farklı uzunlukta diziler (*Array*) olmasına ihtiyaç duyulur yani vektör ya da matrislerden bir dizi (*Array*) olması söz konusudur. F'ler ya da *Front*'lar dizi sayılmaktatırlar ve bazılarının içinde 1 eleman ve bazılarının içinde birkaç eleman vardır. Başka bir deyişle özetlersek bir dizinin her elemanı gene bir dizi ya da *Array* sayılmakta denilebilir. Böyle birşey *MATLAB*'da *Cell Array* [59] olarak adlanmıştır. Şimdi buraya kadar programı koşturursak Şekil 73'da gösterildiği gibi *WorkSpace*'te F adında bir *Cell Array* oluşmuştur ve onu açarsak içinde 6 tane *Cell* (*Front*) olduğunu göreceyiz.



Şekil 74. *WorkSpace*'te altı tane cell, 6 tane cepheyi içinde barındırmıştır

Tablo 4. Cephelerin kümesinin içindeki elemanlar

Front ya da Cell 1 içinde {12 16 20 27 32 34 36 41 44 } = 9 Eleman
Front ya da Cell 2 içinde { 29 14 25 38 2 13 18 24 42 46 } = 10 Eleman
Front ya da Cell 3 içinde {19 37 43 17 3 28 33 49 50 15 31 6 22 } = 13 Eleman
Front ya da Cell 6 içinde { 7 26 } = 2 Eleman

Böylece *Cell*'lerin çeşitli uzunlukta olduklarını görmüş olduk. Tablo 4'de *Cell* 6'nın 2 elemanı vardır ve *Rank*'ı da 6'dır. Böylece Tablo 5'de de gösterildiği gibi Popülasiyonun 7 ve 26'ıncı elemanının *Rank*'ı ya da Rütbesi 6 olmalıdır.

Tablo 5. *NSGA-II* bireylerinin içindekilerinden örnek raporları

>> pop(7) ans = Position: [1 0 0 1 0 0 1 1 1 1 0 0 0 0 1 1 1 0 1 1] Cost: [2x1 double] Rank : 6 DominationSet: [ ] DominatedCount: 0 CrowdingDistance: [ ]
>> pop(26) ans = Position: [1 1 1 0 1 1 0 0 1 1 0 1 1 0 1 0 0 1 1 0] Cost: [2x1 double] Rank : 6 DominationSet: [ ] DominatedCount: 0 CrowdingDistance: [ ]

Aşağıdaki *NonDominatedSorting* fonksiyonunun girişi popülasyondur ve çıkışı cephe ve sıralanmış popülasyonudur.

% Non-Dominated Sorting

[pop F]=NonDominatedSorting(pop);

NS fonksiyonun işleminin sonunda hepsinin *DominatedCount*'ları sıfır olması gerekir ve nedeni *Ranking* konusunda bahsedildiği gibi rütbesi 1 olanlar *Front* 1 olarak mağlup olmayan cevaplardır ve tabiki mağlup olma sayaçı *DominatedCount* = 0 olması gerekmektedir. *NS* algoritmasında F_1 ortaya çıktıktan sonra diğer elemanlara bozucu etkisini kaldırarak yani F_1'in yokluğunda diğer mağlup olmayan elemanlar F_1 sayılır ki

aslında F_2 olmuş oluyorlar. Burda *Non-Dominated Sorting* algoritması bitirmiş olmuştur. Test etmek için *DominationSet'*lere şöyle bakabiliriz:

```
>> pop(13)
 ans =
  Position: [1x20 double]
  Cost:  [2x1 double]
  Rank: 2
  DominationSet: [1 3 4 9 28 35]
  DominatedCount: 0
  CrowdingDistance: [ ]
```

Yukarıda görüldüğü gibi popülasiyonun 13'üncü elemanı [1 3 4 9 28 35] kümenin elemanlarını mağlup etmektedir. *Rank*'ı da 2'dir demek ikinci cephededir, yani aşağıda gösterildiği gibi F{2}'yi açarsak şöyle :

```
>> F{2}
ans = 29 14 25 38  2 13 18 24 42 46
```

Popülasyonun onüçüncü elemanı yani Pop(13)'de yukarıdaki cephenin içinde olması gerekir. Yapılacak diğer işlemlerden biri *Crowding Distance*'ın hesaplamasıdır ve bunlar 1.21. Yoğunluk mesafesi isimli başlıkta ve denklemleri 81-84 ile beraber ilgili konuda anlatılmıştır. *CD* ifadesine göre önceki ve sonraki elemanların arasındaki farkı hesaplanarak *Objective*'lerin aralarındaki maksimum fark'a bölünür. d_i^j tüm *Objective*'ler için hesaplanır ve sonra toplanır aşağıdaki ifadeyi kullanarak total *Crowding Distance* elde edilir.

$$d_i^j = d_i^1 + d_i^2 + \ldots + d_i^m = \sum_{i=1}^{m} d_i^j \qquad (113)$$

pop=CalcCrowdingDistance(pop,F)fonksiyonu F popülasyonun elemanlarını alır. Çünkü *CD* işlemi bir cephenin elemanlarının arasında yapılmaktadır. F önceki aşamada hesaplanmıştır ve *CD* tane tane her cephe için ayrı ayrı hesaplanacaktır. Her cephenin

elemanları maliyet üzerinden sıralanır ve tüm *Objective'*lere göre elemanlar için tane tane d_i^j'ler hesaplanır sonra bu işlemler aynen bir sonraki cepheye geçmektedir. Bir önemli hususu açıklama gerekir şöyle listenin başında olanlar ya da sonunda olanlar Şekil 53'de gösterildiği gibi cephenin sonuncu elemanın birsonrası yoktur ve birincinin bir önceliği yoktur. Budurumda sonuncunun birsonrasını ∞ ve birincinin biröncesini'de ∞ olarak var sayılmaktadır. Buna göre kenarlarda olan popülasyonun elemanları *CD*'si ∞ sayılmaktadır. *Crowding Distance* açısından bunların arasında bulunan her cevap kesin kötü bir cevap sayılmaktadır. Şimdi buraya kadar programı koşturuyoruz:

```
>> F1=pop(F{1})
F1 = 7x1 struct array with fields:
  Position
  Cost
  Rank
  DominationSet
  DominatedCount
  CrowdingDistance
```

F_1'in elemanları 7x1 struct biçiminde olmuştur. Birinci cephenin birinci elemanının *CD*'si ve aynı şekilde Birinci cephenin ikinci elemanının *CD*'si aşağda raporlanarak gösterilmiştir.

```
>> F1= pop (F{1})
>> F1(1)
ans =  Position: [0 0 0 1 0 1 1 0 0 0 0 1 1 0 1 1 1 0 0 0]
   Cost: [2x1 double]
   Rank: 1
   DominationSet: [1x34 double]
   DominatedCount: 0
   CrowdingDistance: 0. 6446
```

>> F2=pop(F{**2**})

>> F2(12)

ans = Position: [1x20 double]

  Cost: [2x1 double]

  Rank: **2**

  DominationSet: [25 29 34 46]

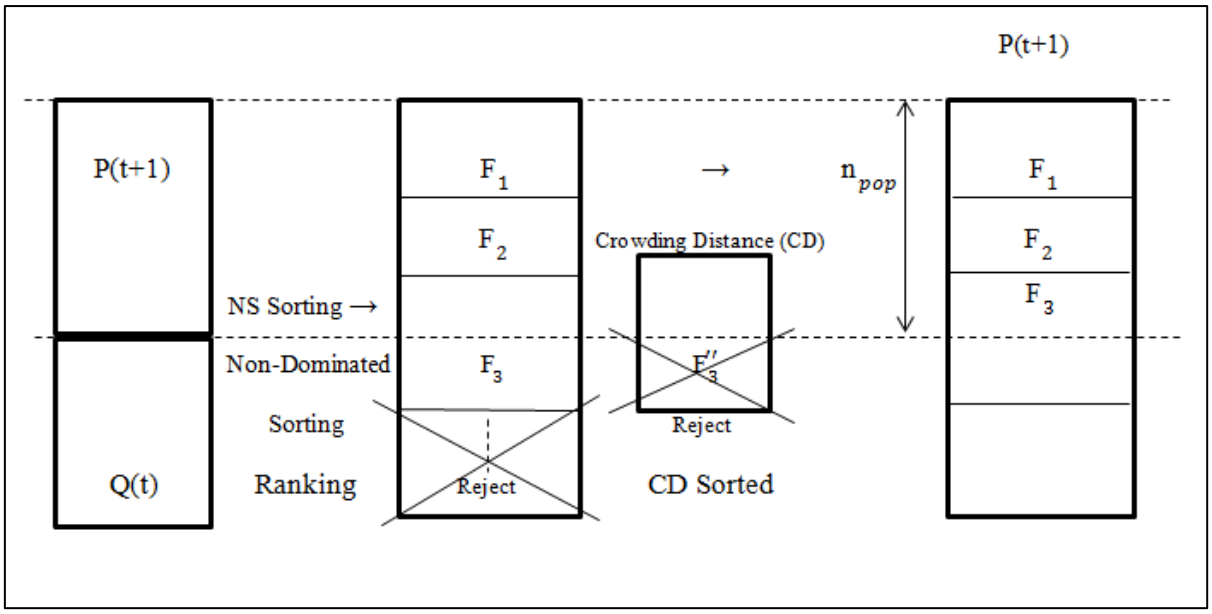
  DominatedCount: 0

  CrowdingDistance: 0. 7631

Böylece *Crowding Distance* işlemi burda bitmiş Olur.

2.5. Popülasyon Sıralaması (*Sort Population*)

Birleştirme, sıralama ve kesme senaryosuna geri dönelim. Aşağıda Şekil 75'de gösterildiği gibi burda sıralama yapılıp birtakımı yeni popülasyona P(t+1) aktarmak için seçilmekte ve sonra rütbeleme yapılıp yeni popülasyona doğrudan gidebilenler aktarılmakta ve gidemeyenler ise kalmaktadırlar. Aslında cepheleri *Crowding Distance*'a göre sıralarsak yinede işin mantığı doğru olur sonuçta aktarılacakların iyileri listenin üst kısmındalar ve emin olabiliriz popülasyon aşağıya doğru kötüleşmekte olacaktır. Popülasyon *CD*'ye göre sıralanabilir.



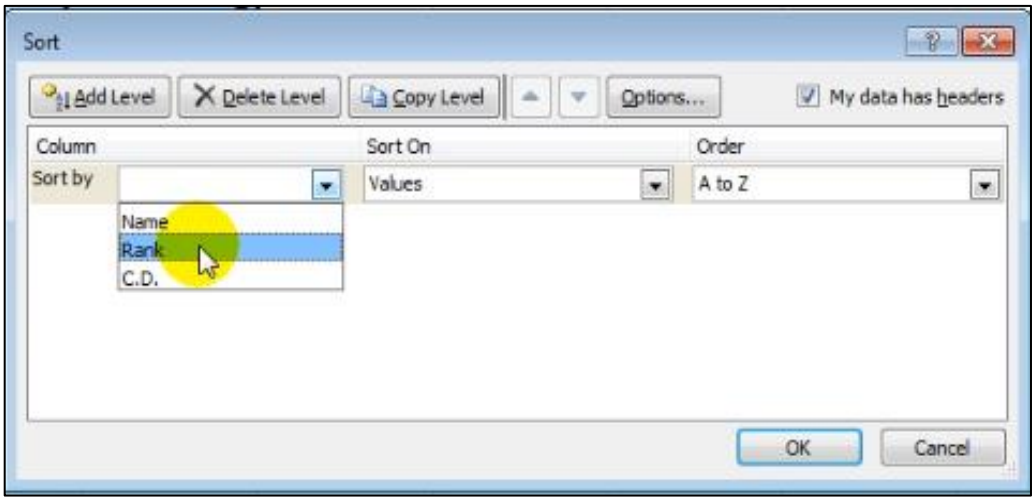
Şekil 75. Bir cephenin elemanları CD kriterine göre sıralanması

Fakat tanımlanan problem gibi iki kritere göre sıralamak istersek ne yapılabilir! Burda asıl kriter *Rank* ve sonraki kriter *Crowding Distance*'a aittir. Yani popülasyon elemanlarını önce birinci derecede önemi olan kritere göre sıralanır ya da önemi ikinci derecede olan kritere göre sıralanır sorusuna açıklık getirerek yola çıkalım. Önce rütbeye göre sonra *CD*'ye göre ya da tam tersi sıralama durumlardan hangisine göre yapılacağını inceleyelim. Önce rütbeye göre sıralama yapılırsa yanlış olur. Önce *CD*'ye göre sıralama yapılacaktır. Bakıldığında burda rütbeye göre sıralanmış gibi gözükmektedir ancak unutmayalım ki F_1, F_2, F_3, … *NS* algoritması aşamasında bozucu etkilerinden dolayı birbirinden izole tutlmuşlardır. Ama *CD* bir içsel gruplu ya da gruplararası (*Intergroup*)

kriteridir ve en iyisi önce *CD*'ye göre ve sonra rütbeye göre sıralama yapılmasıdır. Matematikte ifade edilirse önce *Rank* ve Sonra *CD*'ye göre kavramı doğru olacaktır. Ancak bu programlamada ters uygulanmaktadır. Önce *Rank* sonrada *CD*'ye göre sıralama yapılırsa işlemin sonucunda muhtemelen rütbeye göre yapılmış olan sıralamanın etkisi kalmayacaktır. Ama önce *CD*'ye göre sıralama yapıp sonra rütbeye göre yapılırsa elde edilen sıralama kesin rütbeye göre olacaktır. Eğer elde edilen sonuçta *CD*'den bir etki kalırsa dahada iyi, ama en azından *Rank*'a göre yapılan sıralamanin bozulmadığından emin olabiliriz. *Excel*'de bu konuyu kanıtlamak için bir örnek inceyelim, Tablo 6'daki gibi *Rank* ve *CD*'den bir tablo oluşturuyoruz ve hanelerini bir takım değerler ile dolduruyoruz. Önce *Excel*'de *Home → Sort → Custom Sort* seçeneği ile tabloyu Şekil 76'da gösterildiği gibi sıralama kriteri *Rank* (*Sort By : Rank ve Order : Smallest to Largest*) artan olarak sıralama yapıyoruz ve sonra yinede aynı işlemi Şekil 77'da gösterildiği gibi Sıralama kriteri *CD* ve busefer (*Order : Largest to Smallest*) azaltan olarak sıralama yapıyoruz.

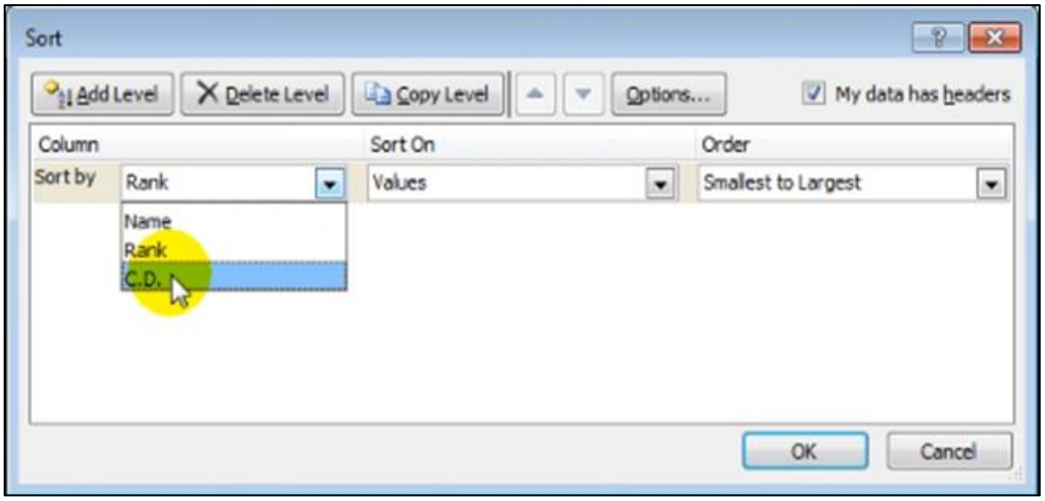
Tablo 6. *Excel*'de oluşturulmuş *Rank* ve *CD* değerleri

1	Name	Rank	C.D.
2	a	1	3
3	b	3	4
4	c	2	6
5	d	1	2
6	e	1	8
7	f	3	4
8	g	3	1
9	h	2	9
10	i	2	4
11	j	1	2



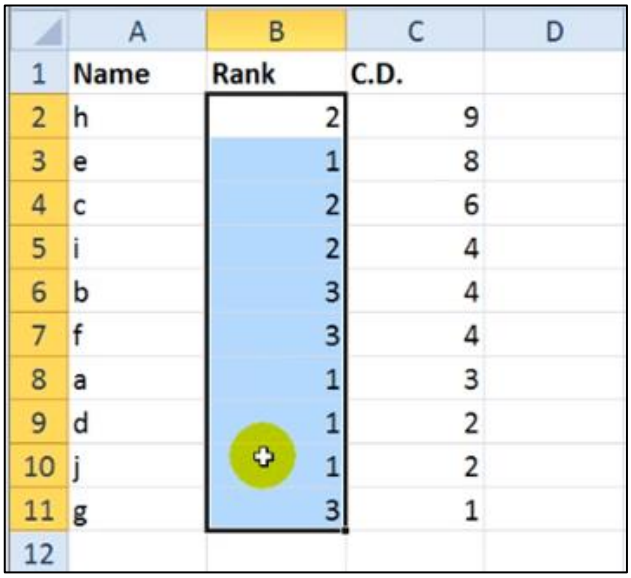
Şekil 76.*Excel*'de sıralama kriteri *Rank* ve *Order*: Artan (*Smallest to Largest*)

CD büyükten küçüğe sıralama yapılır çünkü *CD*'si büyük olanlar iyi sayılmakta ve iyiler listenin üst kısmında olmaları gerekmektedir. Hatırlayalım ki rütbesi düşük olanlar da iyi sayılmakta ve listenin üst kısmında olmaları gerekmektedir.



Şekil 77. *Excel*'de sıralama kriteri *CD* ve *Order*: Azaltan (*Largest to Smallest*)

Tablo 7. *Excel*'de rütbeye göre yapılmış sıralama



	A	B	C	D
1	Name	Rank	C.D.	
2	h	2	9	
3	e	1	8	
4	c	2	6	
5	i	2	4	
6	b	3	4	
7	f	3	4	
8	a	1	3	
9	d	1	2	
10	j	1	2	
11	g	3	1	
12				

Tablo 7'de gerçek şudur ki en son *CD*'ye göre sıralama yapıldı. Ama görüldüğü gibi *Rank* sırayla çıkmamıştır. 1 varken en üst listede 2 bulunmaması gerekirdi. Şimdi tekrar başa dönerek *Excel*'de anlatılan işlemi önce *CD'*ye göre ve sonrada rütbeye göre sıralama yapıyoruz :

Tablo 8. *Excel*'de sıralamanın sonucu

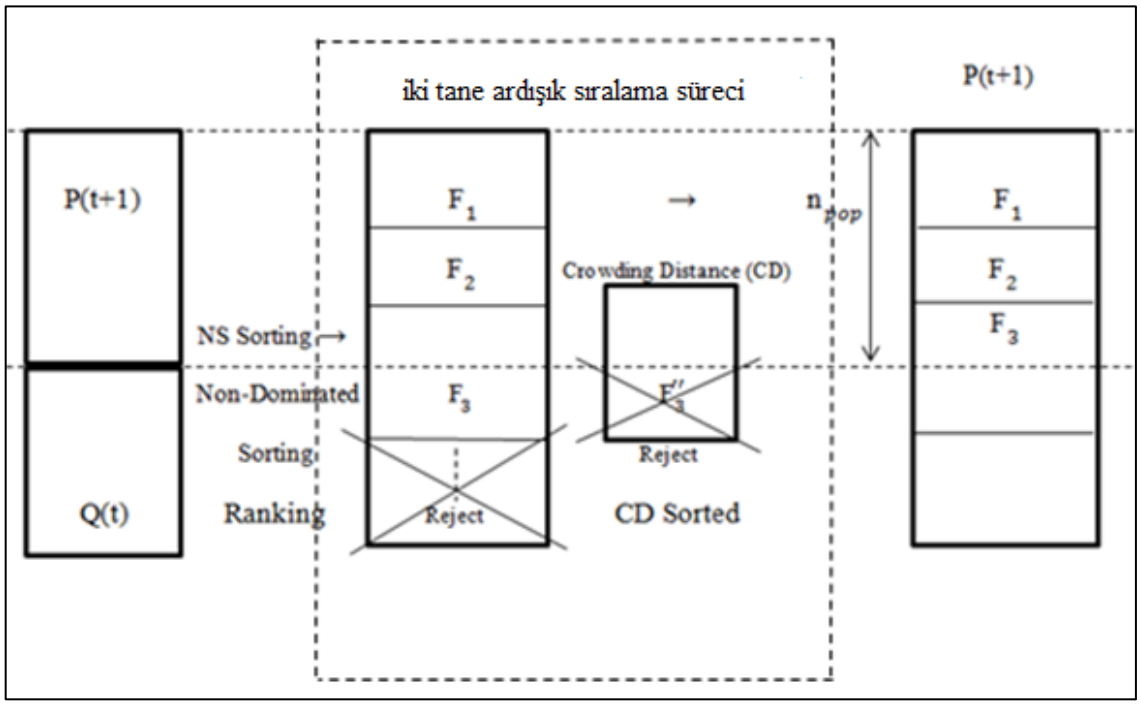
	A	B	C
1	Name	Rank	C.D.
2	e	1	8
3	a	1	3
4	d	1	2
5	j	1	2
6	h	2	9
7	c	2	6
8	i	2	4
9	b	3	4
10	f	3	4
11	g	3	1

turuncu alan

yeşil alan

mavi alan

Yukarıdaki Tablo 8 doğrudur yani *Rank* 1'ler *Front* 1 olmakta ve onlarla beraber karşılarında *CD*'ler de büyükten küçüğe doğru sıralanmışlardır. *Front* 1 turuncu alanla, *Front* 2 yeşil alanla ve *Front* 3 mavi alanla belirtilmiştir. *Front*'lardan her biri sanki *CD*'ye görede sıralanmışlardır. Birinci *Front*'ta *CD*'si yüksek olanlar listenin üst kısmındalar.



Şekil 78. *NSGA-II*'deki ardışık sıralaması

CD'ler sütununda 9 hepsinden büyüktür. Ama *Front* 2'de bulunduğundan dolayı *Rank* ya da *Front* 1'lerin altında yer almaktadır. Buda birinci kriterin *Rank* olduğunu gösteren somut bir işarettir. Aslında anlatılanlara göre 2 tane ardışık (*Sequential*) sıralama işlemini yapmış olduk. Böylece listeden üstlerden her nekadar popülasyon elemanı seçersek en iyilerini seçtiğimizden emin olabiliriz hatta *Rank*'ları eşit olsa bile üste gelenler *CD*'si yüksek olanlardır. Dolayısıyla Tablo 8'de listeyi her neresinden kesersek birşeyi kaybetmiş değiliz. Ozaman popülasyonu birleştirme (*Merge*) sonra ardışık sıralama ve son olarak kesme (*truncate*) işlemleri yapılacaktır. Yukarıdaki Şekil 78'i önceki nispeten karmaşık yapısını *Sort Population* ile binevi sadeleştirmiş olduk. *MATLAB*'da yapılan program koşturulduğunda çıkan rapora göre istenilen şey

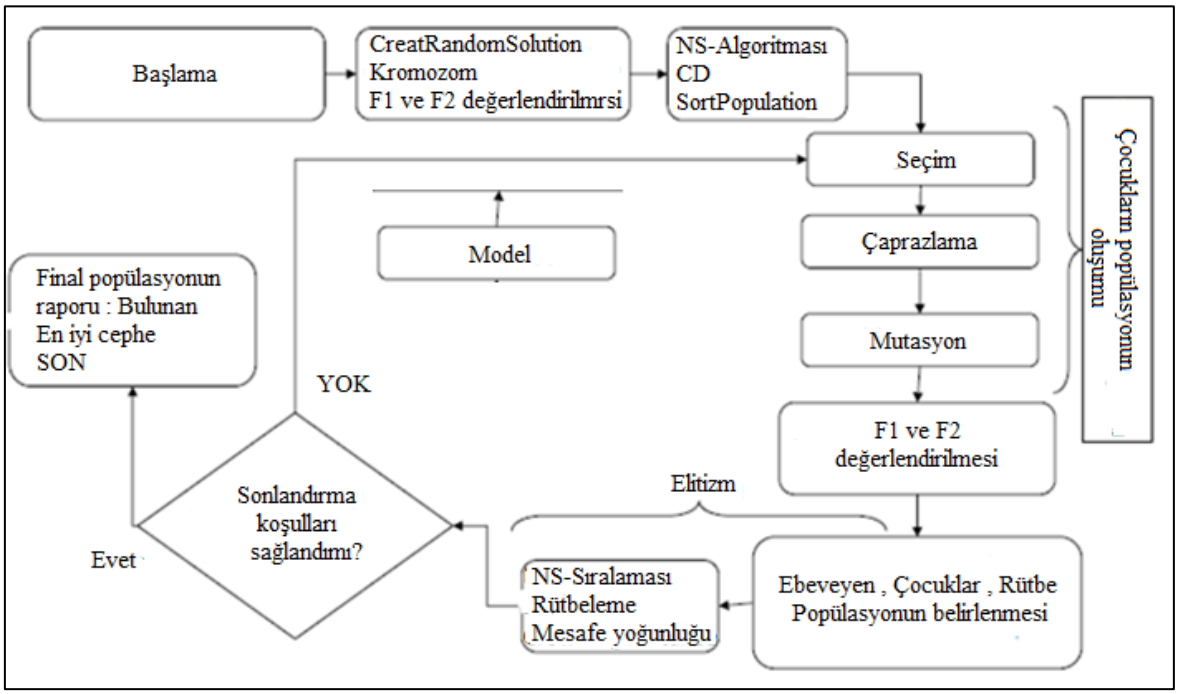
Tablo 9'da gösterildiği gibi gerçekleşmiştir.

Tablo 9. *NSGA-II* popülasyonunun sıralanmış yoğunluk mesafesi ve rütbeleri

>> CD=[pop. CrowdingDistance]’		7. 0000 Inf
>> R= [pop. Rank]' >> [R CD]		3. 0000 0. 6225
ans =		3. 0000 0. 5583
1. 0000 Inf		3. 0000 0. 5413
1. 0000 Inf		3. 0000 0. 4914
1. 0000 1. 0304		3. 0000 0. 4334
1. 0000 0. 5970		3. 0000 0. 3659
1. 0000 0. 3902		3. 0000 0. 3376
1. 0000 0. 3502	*Rank* ve *CD*	3. 0000 0. 3131
1. 0000 0. 2729		4. 0000 Inf
1. 0000 0. 1969		4. 0000 Inf
1. 0000 0. 1921		4. 0000 0. 8832
1. 0000 0. 1582		4. 0000 0. 8311
2. 0000 Inf		4. 0000 0. 7090
2. 0000 Inf		4. 0000 0. 6945
2. 0000 0. 9976		4. 0000 0. 4079
2. 0000 0. 3011		5. 0000 Inf
2. 0000 0. 2725		5. 0000 Inf
2. 0000 0. 2379		5. 0000 0. 9142
2. 0000 0. 2359		5. 0000 0. 7175
2. 0000 0. 1969		5. 0000 0. 5243
2. 0000 0. 1924		5. 0000 0. 4353
2. 0000 0. 1658		5. 0000 0. 3683
2. 0000 0. 1638		6. 0000 Inf
2. 0000 0. 1378		6. 0000 Inf

2.6. Problemin NSGA-II Algoritma ile Akış diyagramı

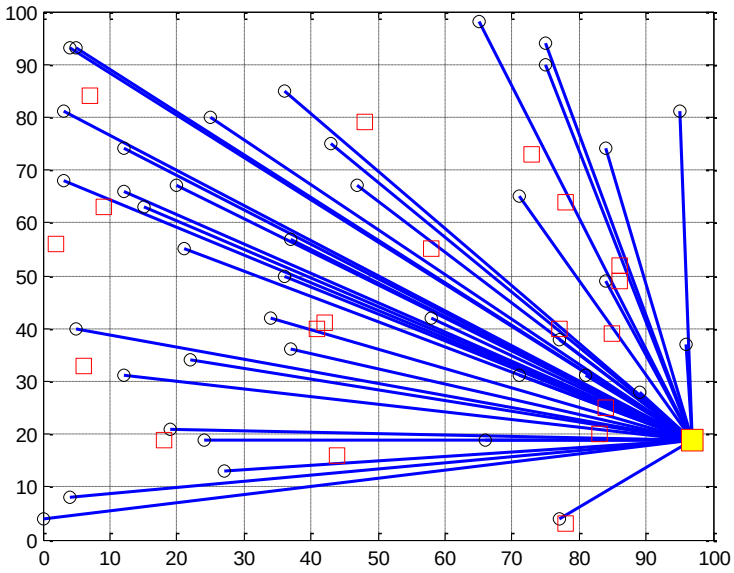
NSGA-II algoritmanın akış diyagramı aşağıda Şekil 79'da gösterilmiştir:



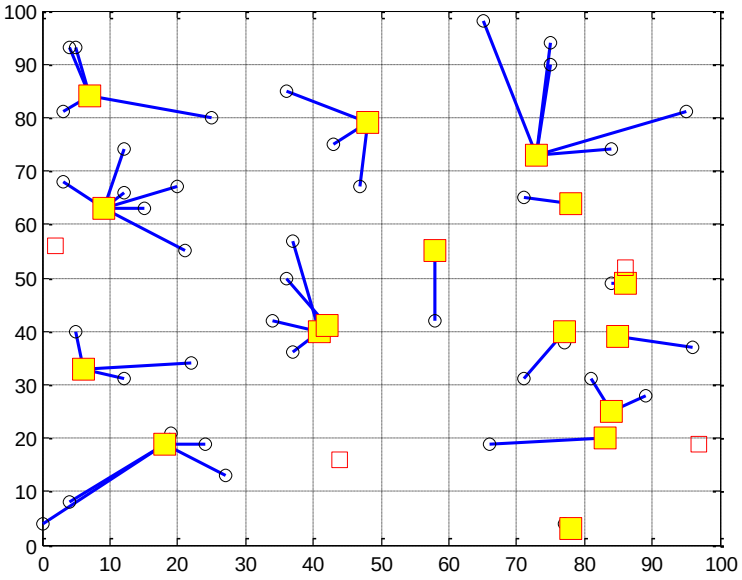
Şekil 79. NSGA-II algoritması ile problemin akış diyagramı

Listenin herhangi bir yerinden kesilerek seçilirse en iyi cevaplar olduğundan emin olunabilir. Burda seçkincilik (*Elitism*) [42] mantığına uyumunu görülmektedir. Artık algoritmadan geriye pek birşey kalmamıştır. Şimdi algoritmanın ana döngüsüne girilmekte ve döngü'de çaprazlama, mutasyon, birleştirme ve sonra bunlara özel sıralama ve kesme süreçleri uygulanmakta olacaktır, Ancak her kesme işleminden sonra tekrar sıralama yapılması gerekmektedir, Çünkü *Rank* ve *CD* nispi özelliklerdir ve hala popülasyonun elemanları birbirlerini pozitif ya da negatif olarak etkileyebilirler. Aşağıdaki şekillerde elde edilen *pareto* cephesinden seçilen bazı cevaplara göre kümelenme yapısı gösterilmektedir. Elde edilen cevap sayısı 50'dir. Bu cevaplardan birincisi *Matlab* ortamında *Solution* = [*0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1*] => *CD = Inf* biçimindedir ve bu cevaba ilişkin dağılım Şekil 80'daki gibidir. Yine elde edilen cevaplardan ikincisi, yirmincisi ve kırkıncısına ilişkin dağılımlar sırasıyla Şekil 81, Şekil 82 ve Şekil 83'de gösterilmektedir. Bu cevapların *Matlab*'taki çıktıları da sırası ile *Solution* = [*1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1*

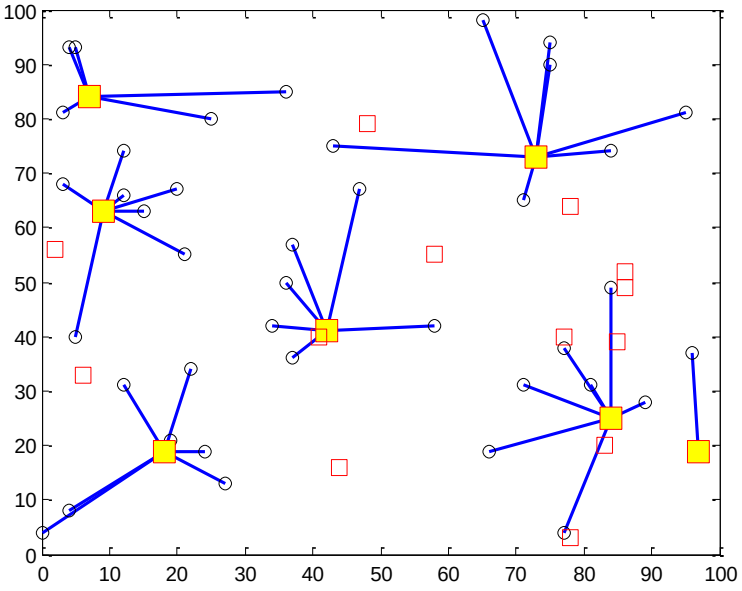
0] => *CD = Inf* ; *Solution* = [*1 1 1 1 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0*] => *CD = 0. 074502;* *Solution* = [*0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0*] => *CD = 0. 046899* biçimindedir.



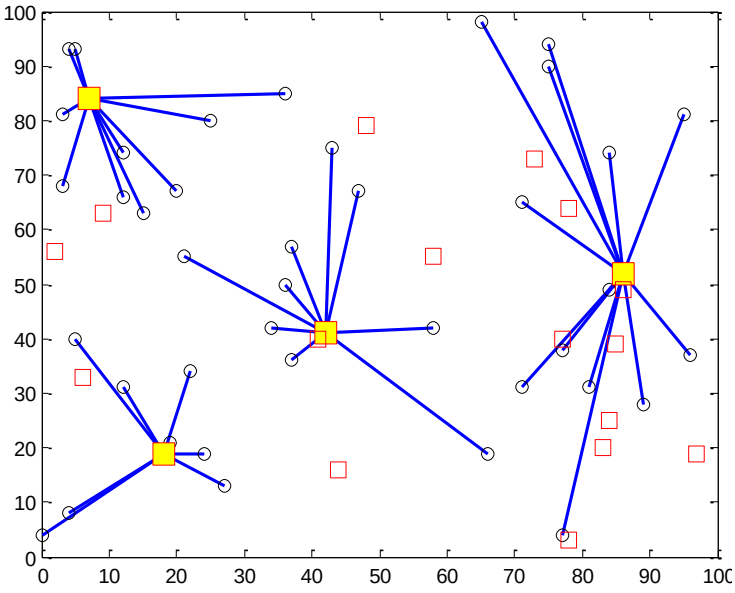
Şekil 80. Pareto-cephesindeki birinci cevap



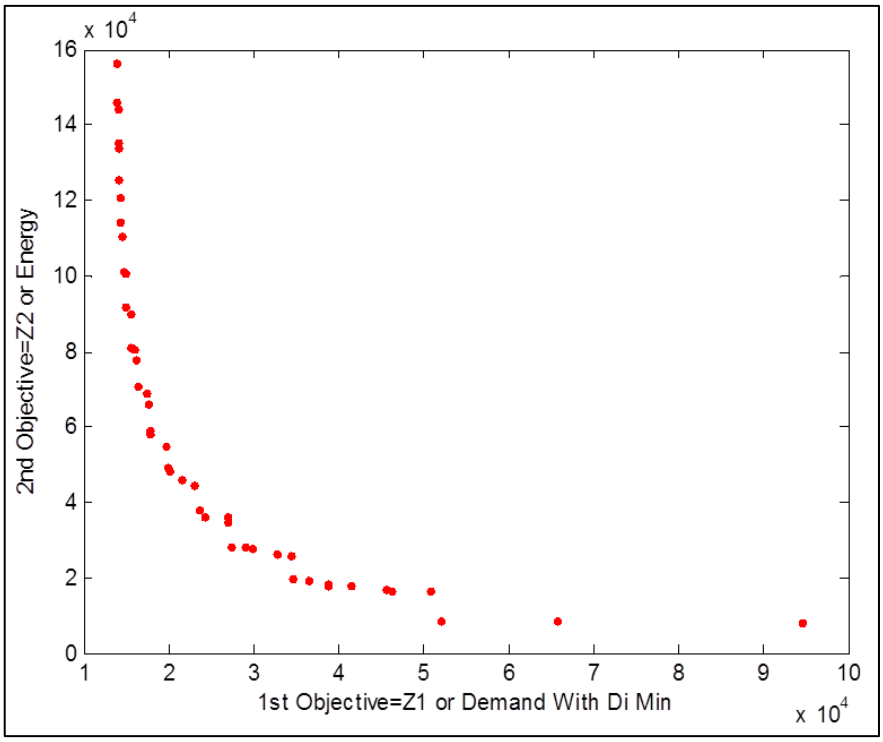
Şekil 81. Pareto-cephesindeki ikinci cevap



Şekil 82. Pareto-cephesindeki yirminci cevap



Şekil 83. Pareto-cephesindeki kırkıncı cevap

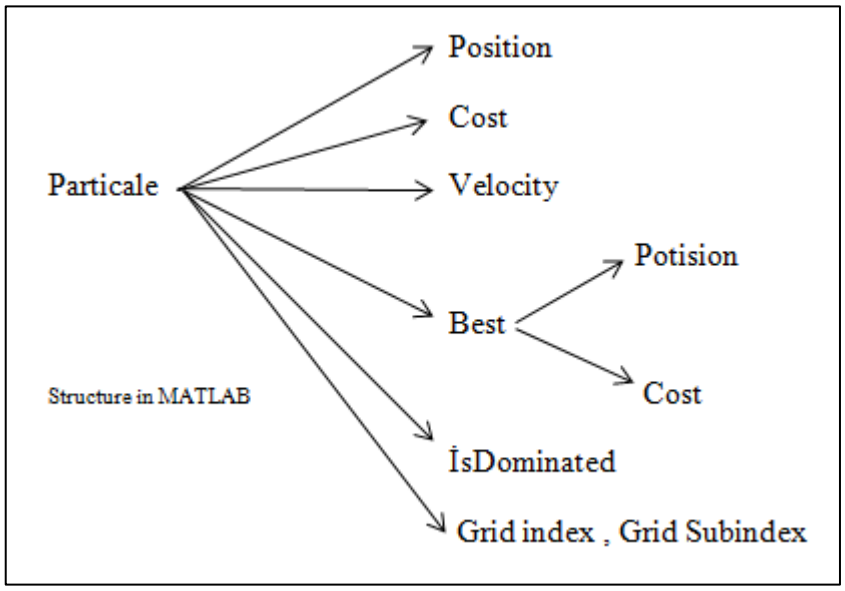


Şekil 84. Problemin *NSGA-II* ile çözülmüş paretosu

Sonuçta yukarıdaki Şekil 84'de gösterilen *Pareto* tanımlanan problemin mağlup olmayan en iyi çözümleri sayılmaktadır. Programa verilen d_i , $\mathrm{C}_{\mathrm{enerji}}$, düğümler ve küme başı düğümlerin sayısı ve hesaplanan mesafeler (D_{ij}) girdilerine göre *pareto* cephesi bulunmuştur. Rütbesi 1 (*Rank*) olan popülasyonlar, yani cephesi 1 (*front 1*) ya da (F_1) olan elemanlarının sayısı 50 tanedir. Şekilde de görüldüğü gibi 50 tane kırmızı nokta problemin çözümü ya da *pareto* cephesini göstermektedir. Bu 50 tane cevap *rank* açısından eşit durumdalar. Dolayısıyla aynı cephe içinde bulunmaktadırlar. *Dr. Carlos A. Coello Coello*'nun [42] kitabına göre bir cephenin içindeki çözümlerin rütbeleri eşit olduğundan dolayı yoğunluk mesafesine göre aralarında kıyaslama yapılarak sıralama (*Sorting*) ve kesme (*truncation*) işlemi gerçekleşmiştir. Ayrıca *CD*'si yüksek olan cevapların daha iyi bir cevap olmasına vurgulanmıştır. Dolayısıyla bu çalışmada cevapların raporlaması için ikinci kriter olarak yoğunluk mesafesi yani *C*D (*Crowding Distance*) [42] açısından listelenmiştir.

2.7. Tanımlanan Problemin *MOPSO* Algoritma ile Çözümü

Tanımlanan Problemin *MOPSO* Algoritma ile Çözümünde n_{Rep} *Pareto*'dan bir tahmin deposudur aynı zamanda *Repository*'nin eleman sayısıdır. Yani problemin ideal *pareto*'sunun yaklaşık cevapları sayılmakta ve belirlenmesi gerekmektedir. Bu sayı genelde 50 tutulur ve n_{Rep}sayısı nekadar yüksek olursa bir okadar da *Pareto*'nun eleman sayısı yüksek demektir, fakat n_{Rep}'in tekrarlanan sayılar olmaması gerekir. Bazı durumlarda problem ayrık bir problemdir ve 10 taneden'den de fazla cevabı yoktur. Bu durumda n_{Rep}=1000 bile tutulursa cevap yinede 10 tane olcaktır ve n_{Rep} dolsa bile hepsi tekrarlanan sayılar olacaktır. Başlatılma (*İnitialization*) aşamasında alogoritmanın çalışması için gereken bellek sağlanılır ve ilk popülasyon oluşturulur. Döngü boyunca birtakım *Array* ve vektörlerin doldurulması için şöyle popülasyonun elemanları yapılanmıştır. yani *MOPSO*'da klasik *PSO* algoritmasında olanların yanısıra başka hususlarda tanımlanmıştır [53].



Şekil 85. Her parçacığın yapısının bileşenleri

Her parçacığın Şekil 85'de gösterildiği gibi konum (*Position*), maliyet (*Cost*) ve hız (*Velocity*) bileşenleri vardır. *Position* ve *Cost* tüm optimizasyon algoritmalarında ortak olarak mevcuttur [53,42]. Her parçacıkta *Best* adlı bir bileşik özelliği vardır ve içinde en iyi pozisyonun deneyimi ve onunla ilgili *Cost* fonksiyonu vardır. Bir başka özellik daha

İsDominated bileşenidir. *İsDominated* yani parçacığın kimin aracılığıyla mağlup olup olmaması söz konusu değildir. Bu *NSGA-II* algoritması gibi Rütbeleme (*Ranking*) tabanlı değildir. Elemanlar 2 grup'tan oluşurlar yani mağlup olanlar ya da mağlup olmayanlar Grubu ve sonuçta bir *True* ya da *False* bayrağı (*Flag* : 0 yada 1) kullanılmaktadır. En sonunda bir ya da kaç eleman aracılığıyla mağlup (*Dominate*) olanlar için *İsDominated True* ve hiç bir şekilde mağlup olmayan elemanlar için *İsDominated False* olmaktadir. Buda *Non-Dominated* ya da mağlup olmayan elemanlarını oluşturmaktadır. *Grid index* parçacığın Şebekenin (*Grid*) neresinde olduğunu göstermektedir. *Grid index*'in iki çeşit ifadesi vardır. Eğer şebekelemenin haneleri sadece tek sayı ile gösterilirse *Gridindex* ve eğer her hane 2 sayılı yani satır ve sutün ile gösterilirse *Grid Subindex* olarak isimlenir. Dolayısıyla *Gridindex*'in yanında *Grid Subindex* adlı bir ek özelliği bulunmaktadır. Tanımlanan problem bir ayrık problemi olduğuna göre ayrık *PSO* (*Binary- PSO*) algoritmasını gözden geçirmekte olacağız.

2.8. İkili (*Binary*) PSO Algoritma

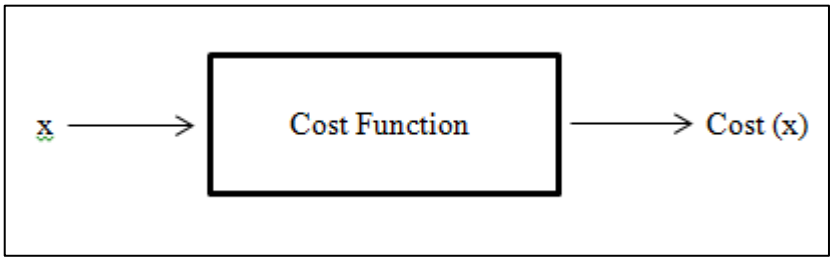
Gördüğümüz *PSO* optimizasyonu formülasyonuyla parçacıkların pozisyonunun konumuna göre bir sürekli parametresi şeklinde uzayda gösterilmiştir. Aşağıdaki ifade ile bu yöntem eski x'i yeni bir x'e ulaştırıyordu ve sürekli problemlere uygun bir algoritmadır.

$$x_i\ (t+1) = x_i(t) + V_i\ (t+1) \tag{114}$$

Ama *Hub Location Problemi* gibi bazı problemler şöyle sıfır ve bir şeklindeler :

$$x = [\ x_1\ x_2 \dots x_{20}\ ]_{1*20} \tag{115}$$

$$x^* = [\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1 \dots 0\ ]_{1*20} \tag{116}$$



Şekil 86. Ayrık x dizisinin değerlendirilmesi

B-PSO [55] algoritmasında sıfır ve bir değerleri problemi optimize etme amacıyla optimum olarak bulunması gerekmektedir. *PSO* algoritmasını önerenler bunu aşmak için bir çözüm daha ileri sürmüşlerdir [55]. Aşağıdaki parçacıkların hız vektörlerinin güncellenmesi aynen önceki gibidir.

$$V_i\ (t+1) = W\,V(t) + C_1 r_1 (p_i(t) - x_i(t)) + C_2 r_2 (G(t) - x_i(t)\ ) \tag{117}$$

Fakat burda pozisyon üzerinde durulmuştur. Yeni pozisyonu $x_i\ (t+1)$ bulmak için önceki tanımlamalar artık geçerli değildir.

$$x_i\ (t+1) \neq x_i(t) + V_i\ (t+1) \tag{118}$$

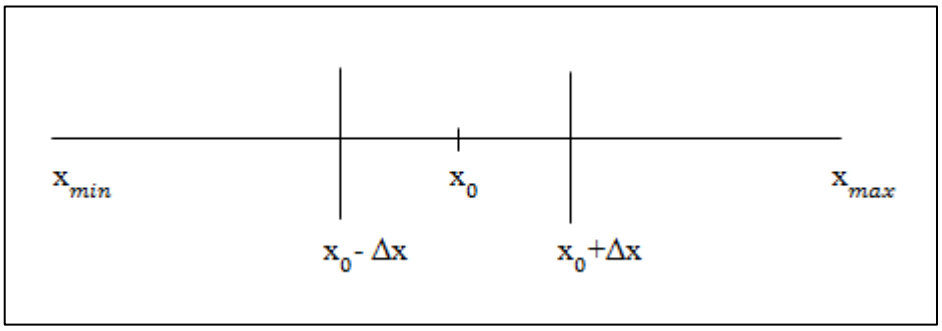
Binary PSO algoritmasında hız vektörü üzerinde herangi bir değişiklik yapılmamakta ve genel olarak hareket özünde sürekli uzayda gerçekleşmektedir fakat parametreler ayrık şekilde gösterilmektedirler.

$$x_i{}^{k^{\text{İterasyon}}} = \begin{cases} 1, \text{ rand} < \text{Sig}\,(V_i\,(t+1)) \\ 0, \text{ Aksi takdirde} \end{cases} \quad (119)$$

Rand ile bir rastgele sayı üretilir. Çıkan sayı denklem 120 gibi *Sigmoid Function*'un sonucundan küçük ise yeni pozisyon 1 ve aksi takdirde 0 olmaktadır.

$$Sig\,(V_i\,(t+1)) = \frac{1}{1+e^{-V_i(t+1)}} \quad (120)$$

Böylece yeni x'ler bulunmuş olur ve aslında normal sürekli *PSO* algoritması bir ayrık tanımlamasıyla örtülmüştür. Şimdi *MOPSO* algoritmasına geri dönüyoruz. Bahsi geçen makalede [53] fig. 5'de gösterildiği gibi bir mutasyon uygulandığını görüyoruz. Tabiki bu tezin çalışmasında problem ikili olduğuna göre ikili mutasyon kullanılmıştır.



Şekil 87. [53] makalesindeki mutasyon

Aşağıda her boyutta şimdiki değişken yani x_0, x_{min} ve x_{max} aralığında olmaktadır ve Δx maksimum mutasyon değeridir.

$$\Delta x = p_m\,(x_{max} - x_{min}) \quad (121)$$

Aşağıda ifade edilene ve Şekil 87‘de gösterilene göre itersyon arttıkca mutasyon oranın giderek düşmesine neden olmuştur.

$$x^{new} \sim u\,(x_0 - \Delta x, x_0 + \Delta x) \tag{122}$$

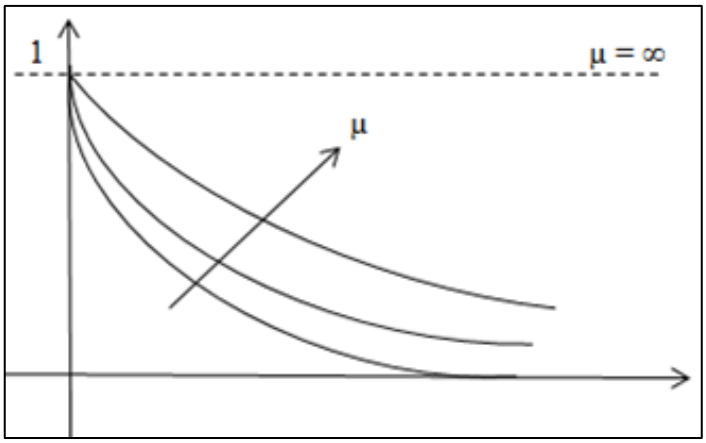
Fakat problem sürekli olmadığına göre üzerinde fazla durmadık ve daha kapsamlı bilgi için ilgili makaleye dönüş yapılabilir. Ayrıca bu çalışmada ayrık mutasyonu kullanıldı. *Binary* Havzasında mutasyon için aşağıdaki gibi sıfırlar bire ve birler sıfıra dönüşmektedir.

- $Eğer\ x_i = 0 \rightarrow x_i^{new} = 1$
- $Eğer\ x_i = 1 \rightarrow x_i^{new} = 0$

$0 \leq \pi_m \leq 1$: Mutasyon etkisinin oranıdır. Mutasyon etkisi altındaki eleman sayısı = π_m* n_{Var} , n_{Var} → tüm değişkenlerin sayısı mutasyon yoğunluğunun belirlenmesi gerekmektedir. π_m=1 olursa yani tüm elemanlar rastgele değişilir demektir ve π_m = 0 olduğu takdirde mutasyon yapılmıyor anlamına gelir. *Coello*’nun makalesinde [53] gördüğümüz gibi mutasyon etkisi altında olan elemanların sayısı giderek azalmaktaydı ve bunu şöyle bir matematik ifadesi ile yapmıştır.

$$p_m = \left(1 - \frac{İt}{Maxİt}\right)^{\frac{5}{\mu}}, \text{ μ: Mutasyon Oranı} \tag{123}$$

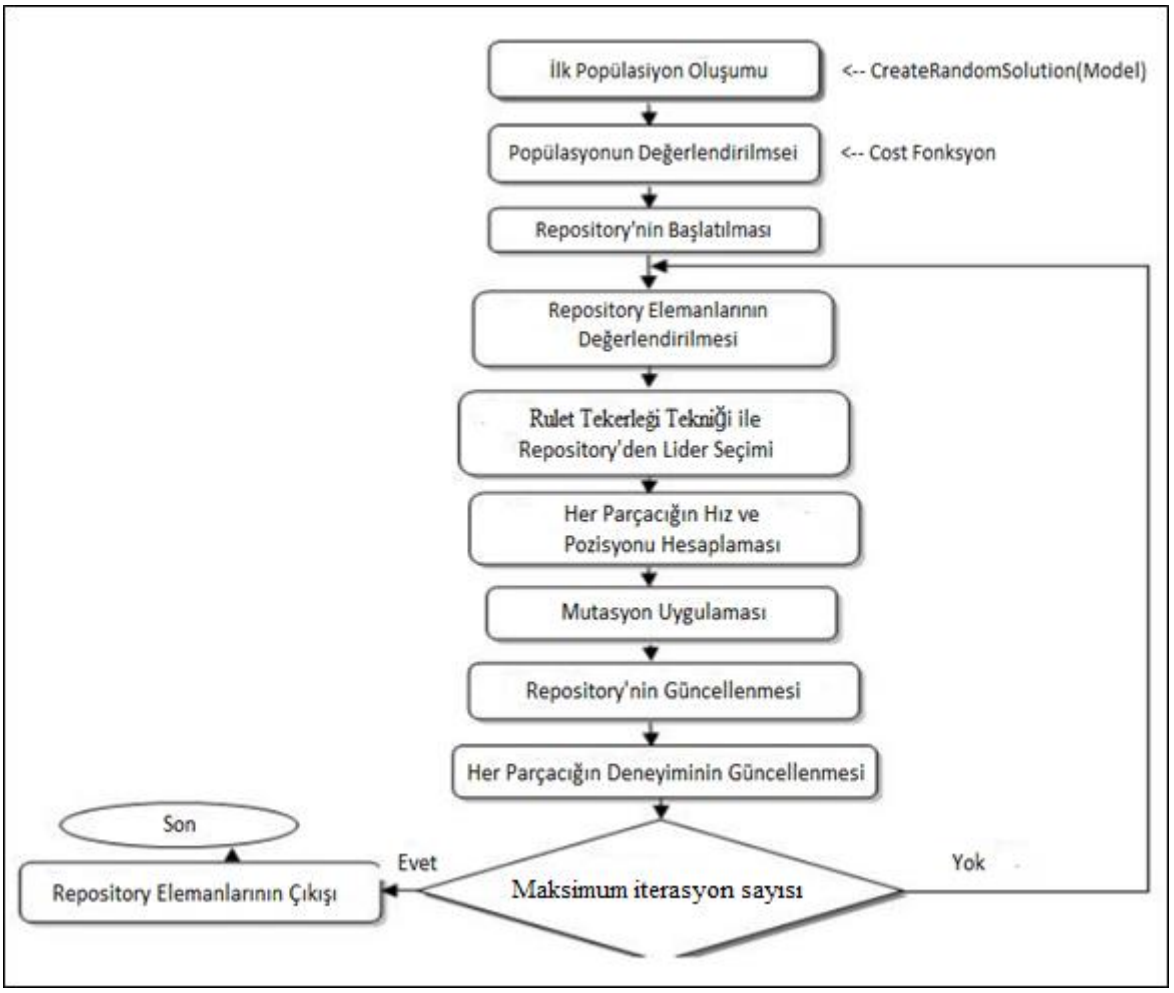
Yukarıdaki Matematik ifadesi şöyle, bir eksi şimdiye kadar geçmekte olan nesil (*İterasyon*) bölü tüm nesiller üssü beş bölü mutasyon oranı (μ) yani p_m Şekil 88’de grafiğin eğiminin denetleyicisi gibi kullanılmıştır. μ parametresinin değişimi ile mutasyon yoğunluğunun denetimi yapılabilmektedir.



Şekil 88. Mutasyon oranı (μ) ile denetim grafiği

2.9. Problemin MOPSO Algoritma ile Akış Diyagramı

Problemin *MOPSO* ile Akış diyagramı Şekil 89'da gösterilmiştir.



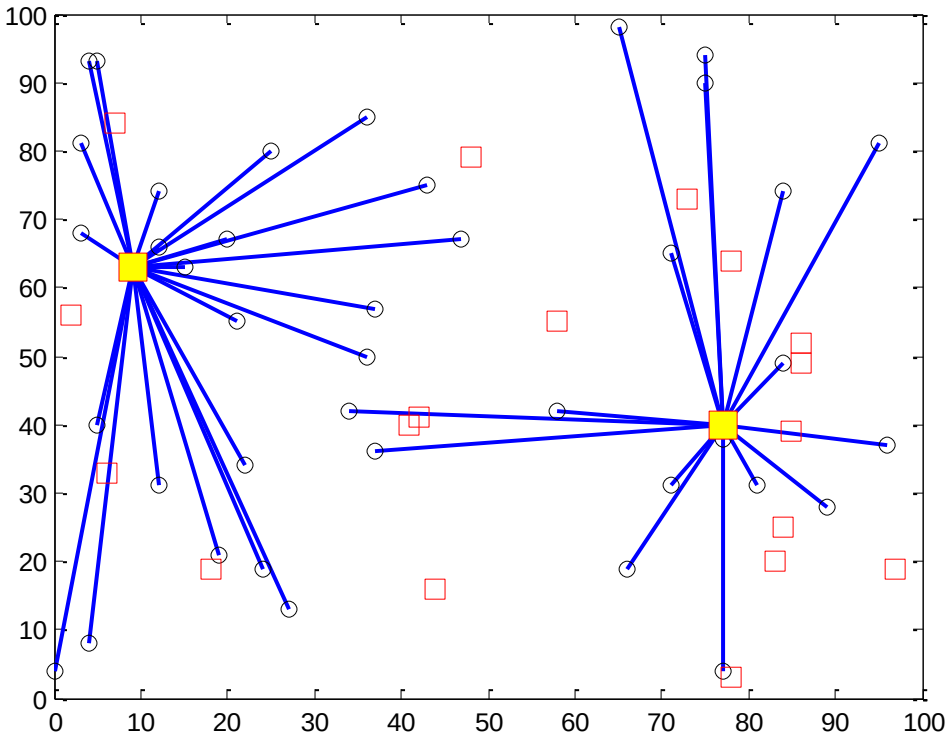
Şekil 89. Problemin MOPSO ile akış diyagramı

Burda modelden problemin bilgileri ve parametreleri çıkarılır. *CreatRandomSolution* fonksiyonu ile rastgele çözümler üretilir ve bunlardan ilk popülasyon oluşturulur. İlk popülasyon amaç fonksiyonu aracılığıyla değerlendirme sürecine girer. Sonra popülasyondan mağlup olmayanlar bulunup depoyu oluşturur. Sonrada deponun elemenları değerlendirmeye geçer ve rulet tekerleği ile depodan lider seçilir. Daha sonra parçacığın hız ve pozisyonu hesaplanır ve mutasyon uygulanır. Daha sonra depo güncellenir, ayrıca her parçacığın geçirdiği deneyimleri güncellenir. İterasyon maksimum sayıya ulaşılırsa deponun elemanları istenilen çözüm olarak raporlanır, yok ise tekrar depo başlatılır

3. BULGULAR

3.1. Problemin Ağırlıklı Toplam Genetik Algoritması ile Çözümü ve Bulguları

w_1=1, w_2=1 parametrelere göre program koşturulduğunda elde edilen sonucunun grafiği aşağıdaki Şekil 90'da gösterildiği gibi elde edilmiştir.



Şekil 90. Problemi W_1=1, W_2=1 ağırlık katsayıları ile çizimi

Sarı kareler küme başları ve diğerleri düğümleri göstermektedir. Oluşan kümeler (*cluster*) şekilde görülmektedir. Her düğüm en yakın baş kümesine bağlanmıştır. Aslında bu bir çeşit kümelemedir (*clustering*). Şekil 90'da görüldüğü gibi 2 tane küme başı bulunmaktadır. Bunun sebebi küme başı kuruluş maliyetinin yüksek olmasıdır. Aşağıda amaç fonksiyonu gösterilmektedir. Amaç fonksiyonundaki W_1 ve W_2 parametreleri sırası ile düğüm ve küme başı kuruluş maliyetlerini temsil etmektedir. Küme başı kuruluş maliyeti azaltılırsa küme başı sayısı artacaktır. Ağırlık katsayıları 0 ile 1 arasındadır.

$$Z_1 = \sum_{i=1}^{n} d_i . D_i^{min} \ : Objective\ 1 \quad (124)$$

$$Z_2 = \sum_j F_j . C_j \quad : Objective\ 2 \quad (125)$$

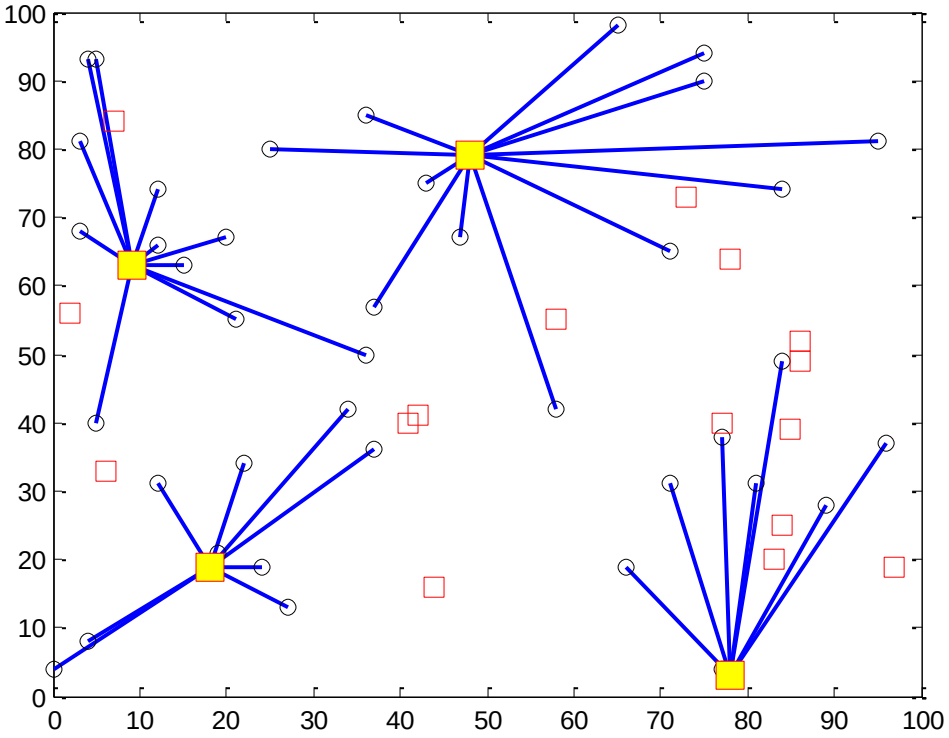
Objective 1'de düğümlerin isteklerinin yanısıra D_i^{min} mesafesi minimize edilmektedir. *Objective* 2'de düğümlerin kuruluş maliyeti ya da aktivasyon enerjisi minimize edilmektedir.

$$Z = w_1 * z_1 + w_2 * z_2 \quad (126)$$

Dekompozisyon *Weighted Sum* [55] yöntemi ile Problem doğrusal Z denklemi şeklinde program farklı çeşitli *w* katsayıları ile koşturulmuştur. Bu çalışmada tanımlanan problem zaten çok amaçlı (*Multi Objective*) problemi türündendir ve çözümü iki boyutlu alanda bir içbükey eğri (*Concave*) şeklindedir ve buna *Pareto* cephesi (*Pareto Front*) denilir. Fakat *Weighted Sum* yöntemi doğrusal yaklaşımından dolayı *pareto* cephesinin içbükey kısımlarını bulmakta yetersiz kalmaktadır. Ayrıca aşağıda görüleceği gibi *pareto'*nun her noktasını (çözümünü) bulmak için program farklı w katsayıları ile baştan koşturulmalıdır.

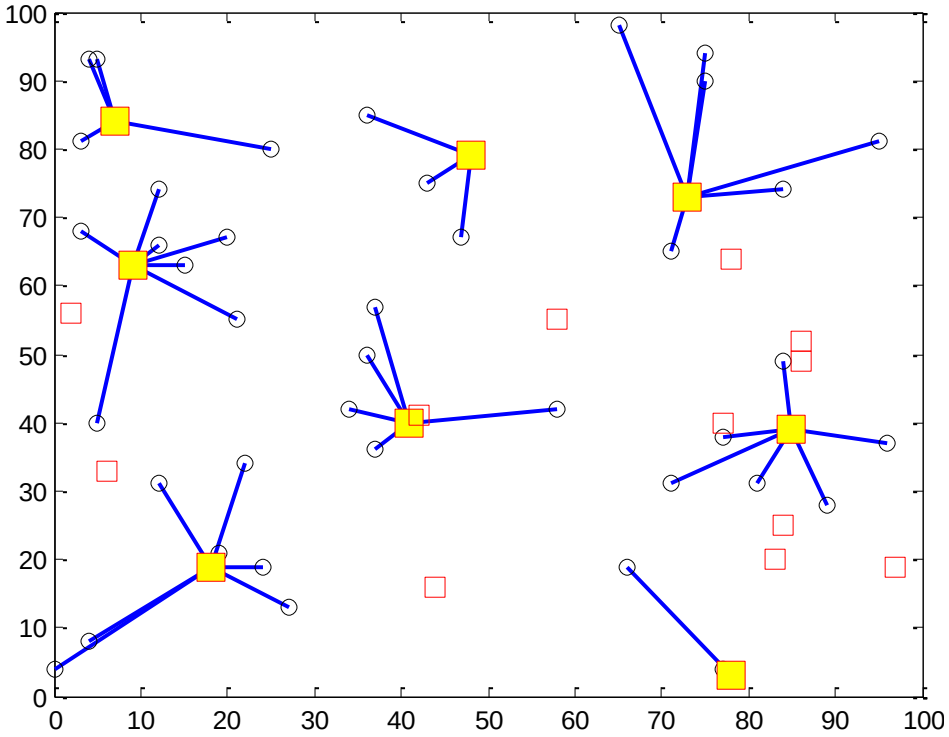
- w_1= Düğümlerin isteklerine verilen önemin ya da (*Objective 1*)'in katsayısı
- w_2= Küme başının Kuruluş Maliyeti ya da (*Objective 2*)'nin katsayısı

Küme başı kuruluş maliyeti ağırlığı yarıya düşerse ($w_1 = 1$ ve $w_2 = 0.5$) şekilde görüldüğü gibi küme başlarının sayısı dörde yani bir öncekinin iki katına çıkar. Dikkat edilirse küme başı sayısı arttıkca normal düğümlerin ilgili küme başlarına bağlanması için daha az mesafe kat etmesi gerekmektedir yani bu durumda düğümlerin isteklerine daha fazla önem verilmiştir anlamına gelir. Küme başının kuruluş maliyeti fazla önemli ise ozaman daha az küme başı aktivasyonu yapılacaktır ve bu durumda açık küme başı sayısı düşmüş olur ve düğümlere de fazla mesafe kat etmesi düşecektir.



Şekil 91. Problemi W_1=1 ve W_2= 0. 5 ağırlık katsayıları ile çizimi

Küme başı kuruluş maliyeti ağırlığı w_2=0. 1 değerine düşerse ve w_1=1 var sayılırsa Şekil 91'de görüldüğü gibi küme başlarının sayısı bir okadar da artıyor yani yine bir öncekinin iki katına çıkar.



Şekil 92. W_1=1 ve W_2= 0. 1 ağırlık katsayıları ile çizimi

Şekil 91'de görüldüğü gibi küme başlarının sayısı birokadar da artıyor yani yine bir öncekinin iki katına çıkmıştır. Problem ağırlıklı toplam (*weighted sum*) [55] yöntemi ile çözülmüştür. Bu sebeple amaç fonksiyonu iki parçalı (*2 objective*) ağırlıklı olarak hesaplanmıştır.

z_1=sum(d. *Dmin) ; *Objective* 1

z_2=sum(f. *Enerji) ; *Objective* 2

w_1=model. w_1;

w_2=model. w_2;

z=w_1*z_1+w_2*z_2 ; % ağırlıklı toplam

sol. A=A;

sol. Dmin = Dmin;

sol. z_1= z_1 ; % *Sol* isimli *struture* biçimi

sol. z_2= z_2 ;

sol. z = z ;

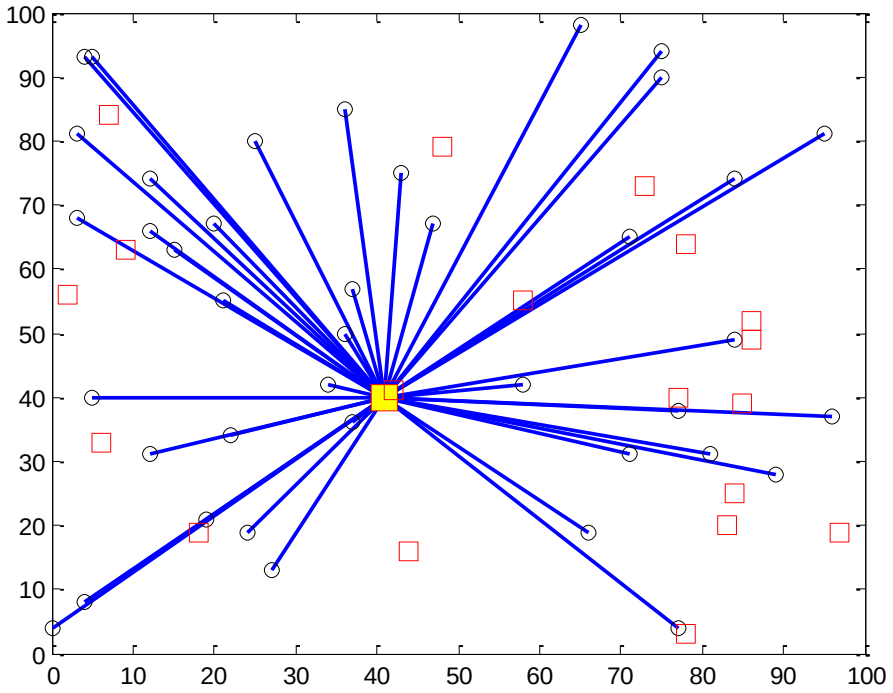
Bahsedildiği gibi yukarıda amaç fonksiyonu iki parçalı yani ayrı ayrı iki *Objective* fonksiyonlu programlanmıştır. Tüm senaryo bilgileri *Model. m* isimli bir alt programda

yazılmıştır. Veriler *MATLAB'*da *Structure* [59] yapısında yazılıp kaydedilmiştir ve tüm bilgiler *Model* fonksiyonu içinden çıkarıp (*Extract*) programlama boyunca kullanılmıştır. Örneğin : D_i^{min}, A, z_1, z_2, z bilgileri bir *sol* isimli *structure* biçiminde kaydedilmiştir ve *sol* içinden D_i^{min}, A, z_1, z_2, z verileri programlama boyunca çıkarıp kullanılabilmektedir. Program boyunca tüm değerlendirmelerde çaprazlama, mutasyon, değerlendirme (*crossover, mutation, evaluation*) iki amaç fonksiyonlu hesaplanıyor.

Tablo 10. Ağırlıklı toplam yönteminde (*Dominance*) durumunun raporu ve analizi

1) $\boldsymbol{W_1 = 0.5, W_2 = 1}$ →	A: [1x40 double] Dmin: [1x40 double] z1: 49865 z2: 9090 z: 3.4023e+004
2) $\boldsymbol{W_1 = 1, W_2 = 1}$ →	A: [1x40 double] Dmin: [1x40 double] z1: 24003 z2: 36269 z: 4.2138e+004

Ağırlıklı toplam yönteminin doğrusal denklemini $Z = w_1 * z_1 + w_2 * z_2$ hatırlatarak yukarıdaki Tablo 10'da 1) $W_1 = 0.5, W_2 = 1$ durumu için yani z_1 *objecive*'in ağırlığı 0. 5 ve z_2 *objecive'*in ağırlığı 1 olduğuna göre programdan z_1 amaç Fonksiyonun çıkışı 49865 olmuştur ve z_2 *objecive*'in çıkışı 9090 değerinde çıkmıştır. 2) $W_1 = 1, W_2 = 1$ durumu için de z_1 = 24003 ve z_2 = 36269 değerinde çıkmıştır. Yukarıdaki durumlarin birbirine göre üstünlükleri yoktur. Her durum diğerine göre bazı noktalarda üstündür ki bu duruma çok amaçlı (*multi objective*) optimizasyon alanında *dominance* problemi olarak adlandırılır. Eğer kuruluş maliyetleri belli bir seviyeden fazla yükselirse Şekil 93'te gösterildiği gibi bir toplumsal küme (*cluster*) meydana gelir yani tek bir küme başına bağlı olan düğümlerin senaryosu karşımıza çıkar.



Şekil 93. Bir toplumsal kümenin ortaya çıkması

3.2. Problemin NSGA-II Algoritma ile Çözümü ve Bulguları

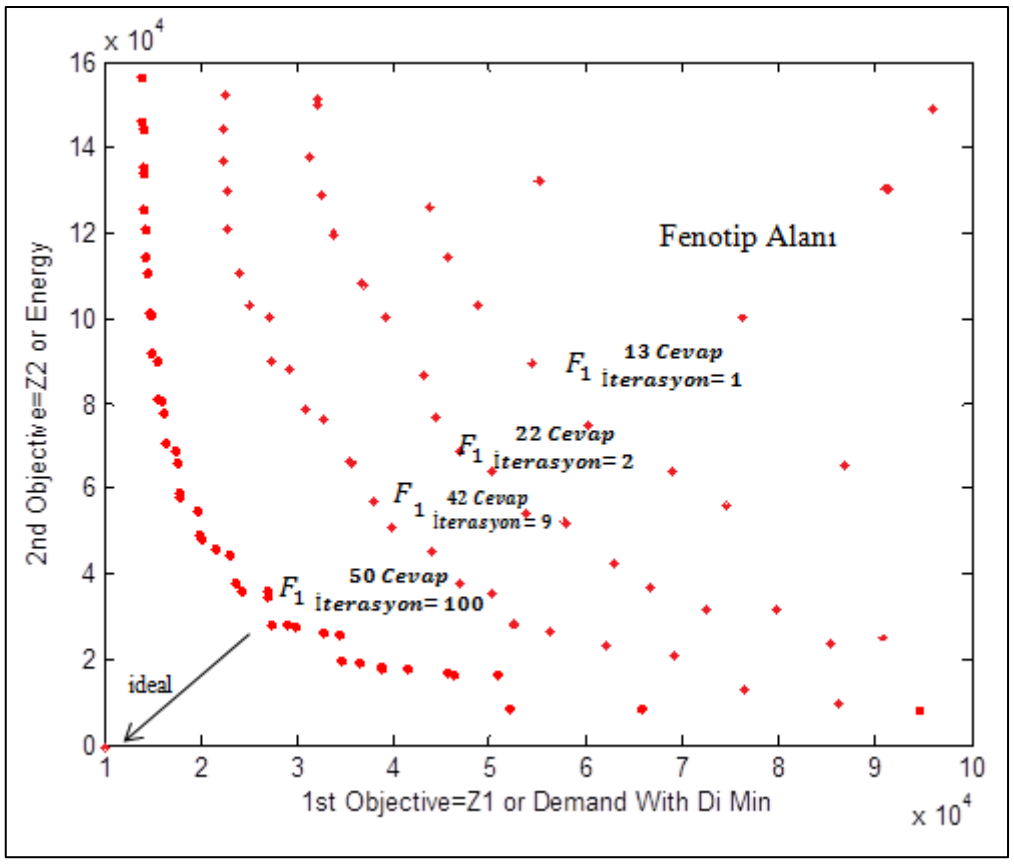
Problem *Multi Objective* olarak algoritmayı 100 İterasyon ve n_{pop} = 50 ayarlarıyla koşturulmuştur.

Tablo 11. Problemin NSGA-II ile çözümünün raporu

Iteration 1: Number of F1 Members = 13	Iteration 17: Number of F1 Members = 50
Iteration 2: Number of F1 Members = 22	Iteration 18: Number of F1 Members = 50
Iteration 3: Number of F1 Members = 32	Iteration 19: Number of F1 Members = 50
Iteration 4: Number of F1 Members = 32	Iteration 20: Number of F1 Members = 50
Iteration 5: Number of F1 Members = 37	Iteration 21: Number of F1 Members = 50
Iteration 6: Number of F1 Members = 36	Iteration 22: Number of F1 Members = 50
Iteration 7: Number of F1 Members = 42	Iteration 23: Number of F1 Members = 50
Iteration 8: Number of F1 Members = 50	Iteration 24: Number of F1 Members = 50
Iteration 9: Number of F1 Members = 42	Iteration 25: Number of F1 Members = 50
Iteration 10: Number of F1 Members = 50	Iteration 26: Number of F1 Members = 50
Iteration 11: Number of F1 Members = 50	Iteration 27: Number of F1 Members = 50
Iteration 12: Number of F1 Members = 50	Iteration 28: Number of F1 Members = 50
Iteration 13: Number of F1 Members = 50	Iteration 29: Number of F1 Members = 50
Iteration 14: Number of F1 Members = 50	Iteration 100: Number of F1 Members = 50
Iteration 15: Number of F1 Members = 50	Iteration 100: Number of F1 Members = 50
Iteration 16: Number of F1 Members = 50	Iteration 100: Number of F1 Members = 50

Yukarıda Tablo 11'de elde edilen rapora göre programın koşumunda birinci itrsasyonunda 13 mağlup olmayan cevap bulunmuştur ve bulunduğu iterasyonun cephe 1 ya da F_1'ni oluşturmuştur. Aynı şekilde iterasyon ikide 22 tane mağlup olmayan cevap bulunmuştur ve bulunduğu iterasyonun cephe 2 ya da F_2'yi oluşturmuştur. Yüzüncü ise yani son iterasyonda 50 tane mağlup olmayan cevap bulunmuştur ve problemin istenilen çözümü ya da *Pareto*'suna ulaşılmıştır.

Elde edilen rapora göre ve tabloda da gösterildiği gibi her iterasyonda bir F_1 ya da cephe bulunmaktadır. Yani her iterasyonda bir cevap kümesi keşfedilmektedir. Dolayısıyla Şekil 94'te gösterildiği gibi iterasyon arttıkca pareto cephesi ideal noktaya doğru ilerlemektedir. Tabloya göre yüzüncü iterasyonda algoritma son bulmuştur ve içinde 50 tane mağlup olmayan çözüm bulunmaktadır.



Şekil 94. Pareto front'un ideal çözüme doğru ilerleyişinin gösterimi

İterasyon 11'den sonra cephelerin elemanlarının sayısı hep 50 tane üzerinde sabit (*Stall iteration*) kalmıştır. Yani sonlandırma koşulu iterasyon 11'de sağlanmıştır ve sonraki

iterasyonlarda bir gelişme görülmemiştir. Fakat yinede çalışmalara göre algoritma genel olarak 100 iterasyonla koşturulmaktadır. Yüz iterasyonun sonunda yani 100'cü iterasyon, içinde 50 tane çözüm barındıran en iyi cephe keşfedilmiştir, (*Iteration 100: Number of F1 Members = 50*) ve şimdi başka bir tabloda bulunmuş bu 50 tane çözümün durumu incelenecektir. *Matlab*'da komut penceresinde F1=pop(F{1}) yazıp elde edilen cepheden çözümler şöyle raporlanır.

Tablo 12. Problemin NSGA-II ile çözülen cephenin 50 tane cevaplarının durumu

Number of F1 Members = 50, Element = 50, CD = 0. 028038 . Ranks Of F1 = 1 Solution = [0 0 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 1]
Number of F1 Members = 50, Element = 49, CD = 0. 036908, Ranks Of F1 = 1 Solution = [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
Number of F1 Members = 50, Element = 48, CD = 0. 038269 , Ranks Of F1 = 1 Solution = [0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0]
Number of F1 Members = 50, Element = 47, CD = 0. 038977, Ranks Of F1 = 1 Solution = [0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
Number of F1 Members = 50, Element = 46, CD = 0. 039256, Ranks Of F1 = 1 Solution = [0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
Number of F1 Members = 50, Element = 45, CD = 0. 043112, Ranks Of F1 = 1 Solution = [1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0]
Number of F1 Members = 50, Element = 44, CD = 0. 043443, Ranks Of F1 = 1 Solution = [0 0 1 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0]

Number of F1 Members = 50, Element = 35, CD = 0. 053904, Ranks Of F1 = 1 Solution = [0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0]
Number of F1 Members = 50, Element = 34, CD = 0. 057886, Ranks Of F1 = 1 Solution = [1 0 1 1 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0]
Number of F1 Members = 50, Element = 33, CD = 0. 061288, Ranks Of F1 = 1 Solution = [0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1]
Number of F1 Members = 50, Element = 32, CD = 0. 062723, Ranks Of F1 = 1 Solution = [0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
Number of F1 Members = 50, Element = 31, CD = 0. 066378, Ranks Of F1 = 1 Solution = [1 1 1 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0 0 0]
Number of F1 Members = 50, Element = 30, CD = 0. 06762, Ranks Of F1 = 1 Solution = [0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1]
Number of F1 Members = 50, Element = 29, CD = 0. 06778, Ranks Of F1 = 1 Solution = [0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
Number of F1 Members = 50, Element = 28, CD = 0. 068424 ,Ranks Of F1 = 1 Solution = [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1]
Number of F1 Members = 50, Element = 27, CD = 0. 068862, Ranks Of F1 = 1 Solution = [1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0]
Number of F1 Members = 50, Element = 26, CD = 0. 069079, Ranks Of F1 = 1 Solution = [1 1 1 1 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0]

Tablo 12'nin devamı

Number of F1 Members = 50, Element = 25, CD = 0. 069079, Ranks Of F1 = 1 Solution = [1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 0 0]
Number of F1 Members = 50, Element = 24, CD = 0. 069705, Ranks Of F1 = 1 Solution = [0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1]
Number of F1 Members = 50, Element = 23, CD = 0. 071288, Ranks Of F1 = 1 Solution = [1 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 1 1 0 0]
Number of F1 Members = 50, Element = 22, CD = 0. 071305, Ranks Of F1 = 1 Solution = [1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 1 0 0]
Number of F1 Members = 50, Element = 21, CD = 0. 071305, Ranks Of F1 = 1 Solution = [1 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 1 0 0 0]

Number of F1 Members = 50, Element = 12, CD = 0. 090772, Ranks Of F1 = 1 Solution = [1 0 1 1 0 0 1 0 1 0 0 0 1 0 0 1 1 0 0 0]
Number of F1 Members = 50, Element = 11, CD = 0. 094447, Ranks Of F1 = 1 Solution = [1 1 1 1 1 0 1 0 1 1 0 1 1 0 0 1 1 0 0 0]
Number of F1 Members = 50, Element = 10, CD = 0. 094959, Ranks Of F1 = 1 Solution = [1 1 1 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 0 0]
Number of F1 Members = 50, Element = 9, CD = 0. 10477, Ranks Of F1 = 1 Solution = [0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
Number of F1 Members = 50, Element = 8, CD = 0. 12795, Ranks Of F1 = 1 Solution = [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
Number of F1 Members = 50, Element = 7, CD = 0. 17341, Ranks Of F1 = 1 Solution = [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
Number of F1 Members = 50, Element = 6, CD = 0. 24081, Ranks Of F1 = 1 Solution = [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

Number of F1 Members =, 50 Element = 5, CD = 0. 35601, Ranks Of F1 = 1 Solution = [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
Number of F1 Members = 50 , Element = 4 , CD = Inf , Ranks Of F1 = 1 Solution = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
Number of F1 Members = 50 , Element = 3, CD = Inf , Ranks Of F1 = 1 Solution = [1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0 0]
Number of F1 Members = 50 , Element = 2 , CD = Inf , Ranks Of F1 = 1 Solution = [1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0 0]
Number of F1 Members = 50 , Element = 1 , CD = Inf , Ranks Of F1 = 1 Solution = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]

Raporlara göre Tablo 12'de her iterasyon sayısında mağlup olmayan *NS* algoritma (*Non-Dominated Sorting*), Yoğunluk mesafesi hesaplanması (*Calculate Crowding Distance*), Popülasyonun sıralaması (Sort *Population*) işlemleri yapılıyor ve sonra elde edilen Elitizm sıralamasından n_{pop} kadarı kesilir (*Truncate*) ve yinede aynı işlemlerle güncellenir. Bu iterasyonda kesilen n_{pop} boyutundaki popülasyondan *Rank*=1'ler istenilen *Front* olarak seçilmektedir. Yani F_1 o iterasyonun en iyi *Front*'u olarak saklanılır. Aynı işlemler iterasyon sayısına bağlı olarak tekrarlanır ve her işlem kendi iterasyonunda kendi

F_1'ini bulmaktadır. İterasyon sayısı arttıkca elde edilen *Front*'lar ideal *Pareto Front*'a yaklaşmaktadırlar. En son iterasyondaki bulunan F_1 algoritmanın bulduğu en iyi *Pareto* sayılmaktadır. Yukaridaki programın çıktısının raporuna göre son iterasyon it=100'de F_1'rin eleman sayısı yani nihayi *Pareto*'da mağlup olmayan (*Non-Dominated*) cevapların sayısı 50 tane olarak görülmektedir. F_1'in tüm elemanlarının *Rank*'ları (*Rank* = 1) olmaktadır. *Dr. Carlos A. Coello Coello*'nun [42] kitabına göre ve 2. 4. - *Sort Population* bölümün mantığına uyarak *pareto*'nun çözümlerini yoğunluk mesafesine göre listede üstten aşağıya doğru düşükten yükseğe raporlanmıştır. Rapora göre son dört (*Element* 1,2,3,4) elemanların *CD*'leri sonsuz olmuşlardır ve eşit değerlerle (*CD* = inf) çıkmışlardır. Bulardan birini seçmeye üçüncü bir kriterin olmasına gerek duyulmamaktadır. Burda çözüm üretmek istenirse Rulet tekerleği tekniğini kullanılabilir.

3.3. Problemin MOPSO Algoritma ile Çözümü ve Bulguları

Şimdi n_{pop}=50 tane parçacığı dediğimiz özellikleri ile oluşturuyoruz.

```
empty_particle. Position=[ ];
empty_particle. Velocity=[ ];
empty_particle. Cost=[ ];
empty_particle. Best. Position=[ ];
empty_particle. Best. Cost=[ ];
empty_particle. IsDominated=[ ];
empty_particle. GridIndex=[ ];
empty_particle. GridSubIndex=[ ];
```

Yukarıda bir parçacık (*empty_particle*) *Structur* [59] aracılığıyla oluşturulmuştur ve 50 tane olması için *MATLAB*'da *Repmat* (*RepeatMatrix*) komutu şöyle kullanılmıştır:

```
pop=repmat(empty_particle,nPop,1);
```

Böylece ilk popülasyon (n_{pop}=50) yani 50 tane içeriksiz parçacık oluşmuştur. Yani doldurulmaya hazır boş matris oluşmuştur. *Position* özelliğini doldurmak için

oluşturduğumuz *Model*'den CreateRandomSolution(model) fonksiyonun aracılığıyla rastgele ikili (*Binary*) 1*20 boyutunda çözümler dizisi üretilir.

pop(i).Position = CreateRandomSolution(model);

Böylece *Position* dizileri döngü içerisinde rastgele ikili sayılarla doldurulur ve *Velocity* dizileri algoritmanın başlangıcında sıfır olarak alınmıştır.

pop(i).Velocity= 0 ; or pop(i). Velocity = zeros(VarSize);

Burda sıfır skalar olarak *MATLAB*'da tanımlanabilir, ama en iyisi pozisyon dizisinin boyutunda yani *VarSize* boyutunda sıfır olması daha mantıklıdır. Daha sonra maliyet hesaplanır. Parçacıklar önceden deneyim geçirdikleri pozisyonları olmadığına göre *Personal Best*'leri aynı başta üretilen değerler, pozisyonlar için kullanılır. Bu aşamada klasik *PSO* algoritması olsaydı doğrudan *Global Best* söz konusu olurdu. ancak *MOPSO* böyle değildir ve deponun belirlenmesi gerekmektedir. Bunun için popülasyonun tüm elemanları incelenir ve mağlup olmayanlar depoyu oluşturular. Mağlup etmek ve mağlup olmak nispi ifadeler olduğuna göre önce tüm popülasyon elemanlarının durumu belirlenmesi gerekir. Dolayısıyla *For* döngüsünden önce bu işlem yapılmaktadır ve sonra popülasyon elemanlarından mümkün olan bir çifti seçilir ve mağlup olanın *İsDominated* bayrağı *True* yapılır. Başlangıçta tüm elemanlar mağlup olmayan var sayılır. Olursa mağlup olan bayrağı çalıştırılır ve olmasada işin sonuna kadar *Flag* pasif kalacaktır. Bunu gerçekleştirmek için bir fonksiyon yazılmıştır.

pop=DetermineDomination(pop);

Bu fonksiyonun içinde *Dominance* mantığı kullanılmıştır :

b=all(x<=y) && any(x<y); Evrensel Nicelik ve Varoluşsal Nicelik

Program buraya kadarını koşturulursa :

```
>> pop(1) ans =
  Position: [1 0 1 1 1 1 1 0 1 0 1 0 0 0 0 1 1 0 1 0]
  Velocity: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
  Cost: [2x1 double]
  Best: [1x1 struct]
  IsDominated: [ ]
  GridIndex: [ ]
  GridSubIndex: [ ]
>> pop(2) ans =
  Position: [0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 0 1 0 1 0]
  Velocity: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
  Cost: [2x1 double]
  Best: [1x1 struct]
  IsDominated: [ ]
   GridIndex: [ ]
   GridSubIndex: [ ]
```

Yukarıda görüldüğü gibi pop(1) ile popülasyonun birinci elemanı ve pop(2) ile popülasyonun ikinci elemanı çıkarılıp raporlanmıştır. Tüm popülasyon elemanları için *IsDominated*'lerini şöyle çıkarıyoruz:

Tablo 13. Problemin *MOPSO* çözümünde *İsDominated*'lerinin raporlanması

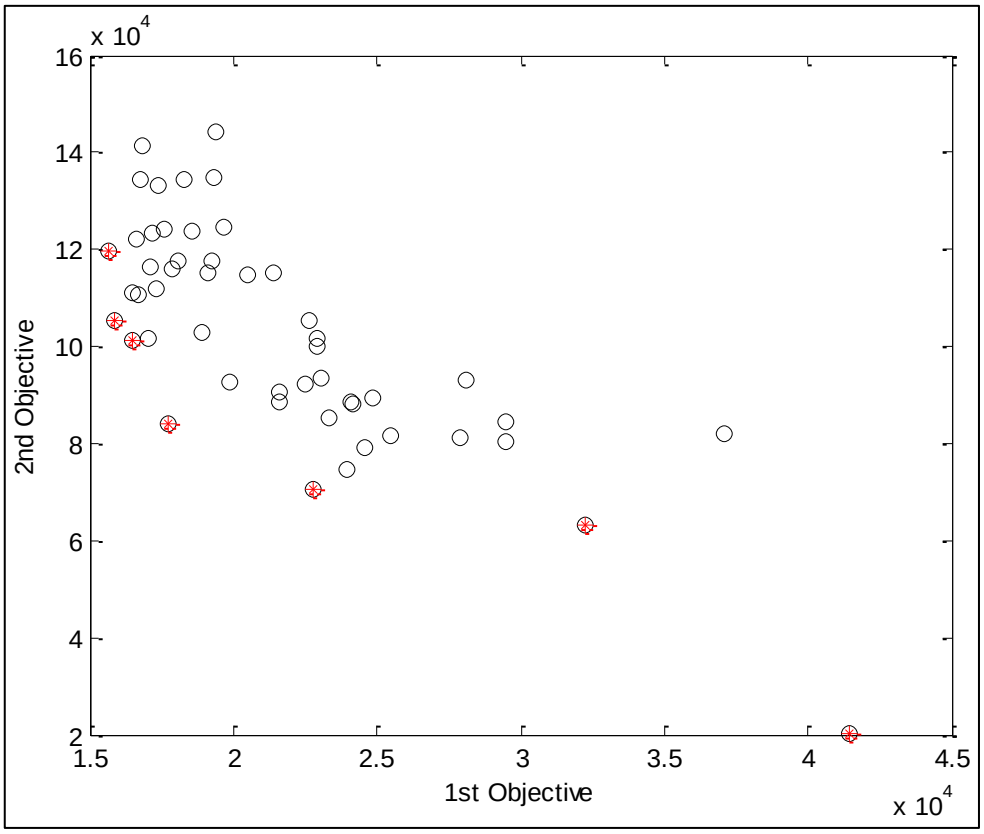
```
>> IsDominated=[pop. IsDominated]

IsDominated =
 Columns 1 through 10
  1  1  1  1  0  1  1  1  0  0
 Columns 11 through 20
  1  1  0  1  1  1  1  1  1  1
 Columns 21 through 30
  1  1  1  1  1  1  0  0  1  1
 Columns 31 through 40
  1  1  1  1  1  1  1  1  0  1
 Columns 41 through 50
  1  1  1  0  1  1  0  1  1  1
```

Tablo 13’te gördüğümüz gibi bunlar ya *True* ya da *False* olmaktadırlar ve çok az sayıda *False* bulunmaktadır. Bu doğal bir durumdur çünkü popülasyon ilk başta hepsi mağlup olmayan durumda olamazlar. Şimdi mağlup olmayanlar yani koyulaştırdığımız sıfırları bunların aralarından bulmak istiyoruz :

```
>> pop(~[pop. IsDominated]), 9x1 struct array with fields:
 Position, Velocity
 Cost , Best
  IsDominated, GridInde, GridSubIndex
```

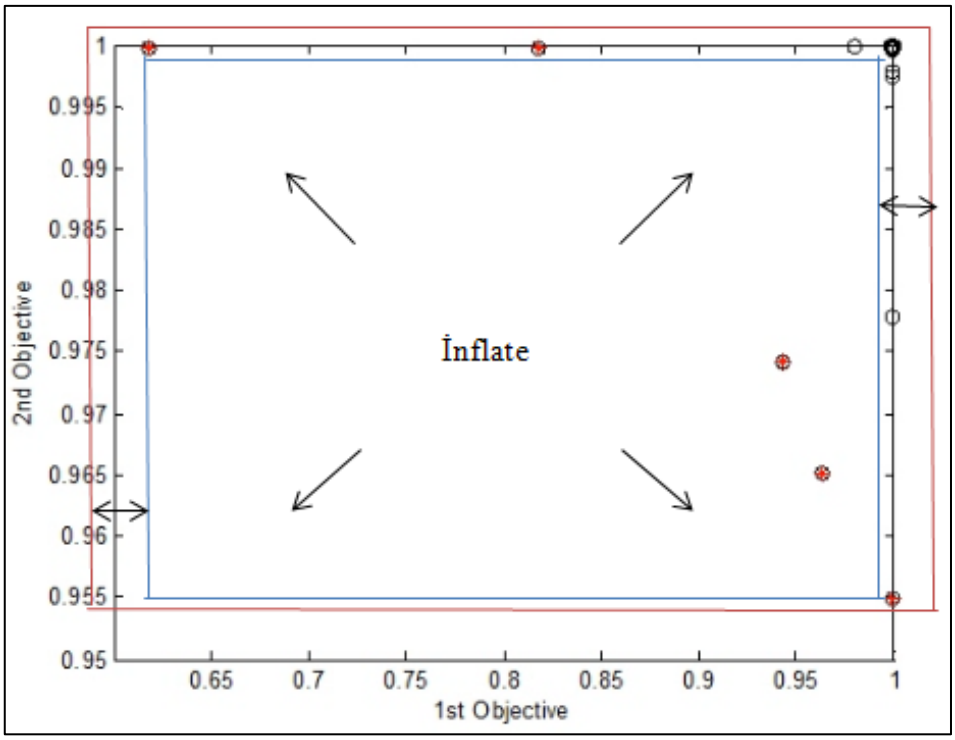
Tablo 13’te görüldüğü gibi 9 tane mağlup olmayan elemanlar bulunmuştur. Dolayısıyla bulunan mağlup olmayanlar depo içine aktarılacaklardır. Tabiki program her koştuğunda *rep* sayısı farklı çıkabilir. Fakat kaç tane olursa olsun mağlup olmayan cevaplar olmaktadırlar. Bunların grafiğide şöyle çizilebilir :

```
pop_costs = [pop. Cost];
plot(pop_costs(1,:),pop_costs(2,:),'ko');
hold on;
rep_costs = [rep. Cost];
plot(rep_costs(1,:),rep_costs(2,:),'r*');
xlabel('1st Objective');
ylabel('2nd Objective'); % grid on;
hold off;
```



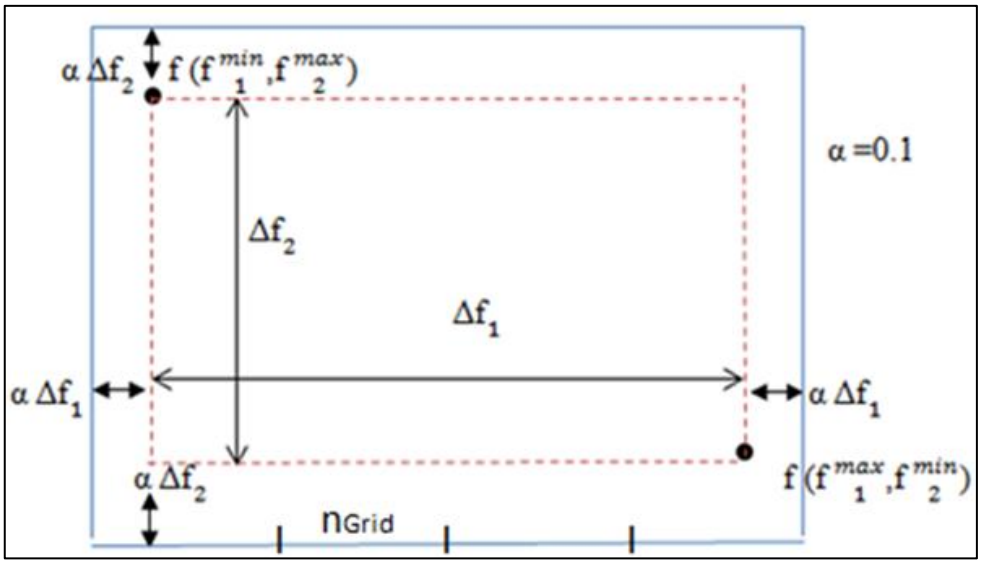
Şekil 95. Programın her koşumu ile depo elemanlarının grafiği

Popülasyon elemanlarından alınan *Cost*'lar *Objective* sayısı kadar satır ve popülasyon sayısı kadar sütunu vardır. Şekil 95'deki grafiğe göre kırmızı çizili noktalar mağlup olmayanlardır. Mağlup olmayan popülasyonu ayırdıktan sonra keşfedilen amaç fonksiyonun alanının şebekelemesine sıra gelmiştir. deponun elemanlarına göre şebekeleme yapılır, Fakat şebekelemeyi biraz şişirerek (*İnflate*) [53] yapılmasına çaba gösterilmiştir.



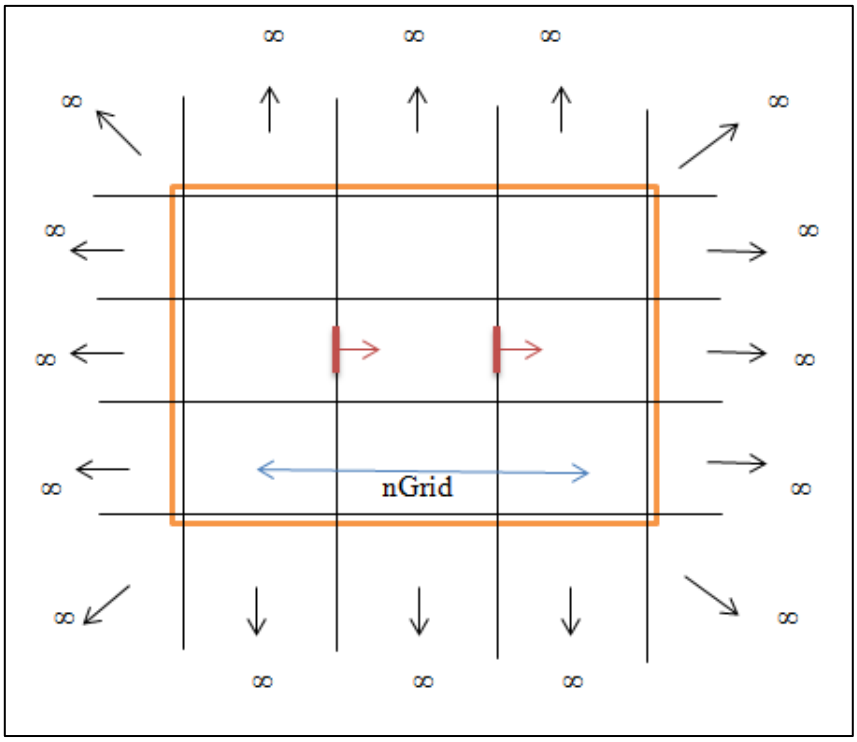
Şekil 96. Şebekelemenin şişirme işlemi

Şekil 96'da gösterildiği gibi tüm popülasyonun elemanları en iç karede, deponun elemanları içinde yer almışlardır. Noktalar tam sınır bölgelere düşmemesi için ve gelecekte yeni keşfedilenlere yer açılması için bu her yönden biraz şişirilir.



Şekil 97. Δf katsayıları ile şişirme uygulaması

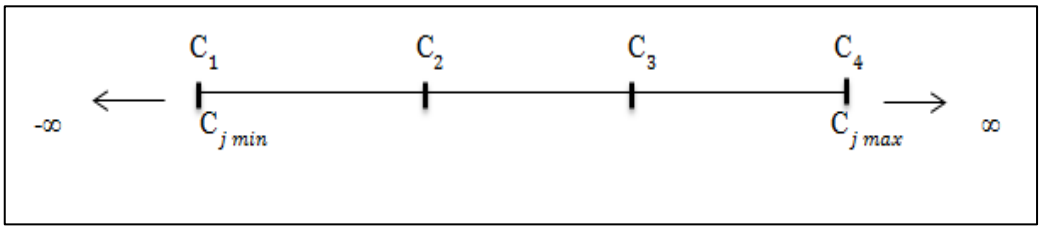
Amaç fonksiyonun genelde iki bitiş noktasının katsayılarda etkisi olmaktadır. Aslında burda biri f_1 açısından minimum, f_2 açısından maksimumdur ve diğeri de f_1 açısından maksimum ve f_2 açısından minimumdur yani noktalı çizilen dikdörtgen alanı gibi depo elemanlarını içeren bir düzlemdeyiz. *İnflate* yapmak için Δf_1'in üst sınırına $\alpha\ \Delta f_1$ kadar eklenir ve birazda $\alpha\ \Delta f_1$ gibi alt sınırından eksilir. α şişirme katsayısıdır (*İnflation Coefficient*) [53] ve genelde α = 0. 1 tutulmaktadır. Aynen böyle düşey boyutunda yani Δf_2'nin üst sınırına $\alpha\ \Delta f_2$ kadar eklenir ve alt sınırından da $\alpha\ \Delta f_2$ miktarında eksilir. Böylece bir şişirtilmiş dik dörtgen elde edilmiş olur.



Şekil 98. nGrid ve alpha (şişirme oranı) ile şebekeleme

Tüm şebekeleme haneleri kaç tane elemanı olacağını belli olması için *nGrid* bölümleme yapılır ve bu alanın dışındakilerde dahil göz önünde bulundurulur. Yukarıdaki Şekil 98'de nGrid=3 *İnflate* alanıdır ve tüm cevaplar bu alanın içinde bulunmaktadırlar. *İnflate* alanın dışındakiler özel haneler sayılır ve sonsuza kadar ∞ sürebilmekteler. Ayrıca her cevap özel haneden sonralarda da bulunursa yinede bu hanenin elemanı sayılmaktadır. Yukarıdaki şekilde alanda bölümleme sayısı *nGrid* olursa her boyutta tüm şebekelemenin sayısı nGrid + 2 olur. Burda iki boyut söz konusu olduğuna göre (nGrid + 2 * nGrid + 2) yani 25 hane olur ve bunlardan 9 tanesi sonlu ve gerisi sonsuz hanelerdir. Her hane her boyutta üst ve alt sınırlarla belirlenir, Dolayısıyla iki boyutlu uzayda her hane 4 sayı ile belirlenir yani üst, alt, yatay ve düşey sınırlarıdır. Bunlardan bazıları alt sınırı ∞ ve bazıları üst sınırı ∞ olmaktadır. Dolayısıyla her hane dört parametre ile belirlenebilir. Şekilde okla gösterildiği gibi her hanenin kenarı yanındaki haneden sayılmakta yani alt sınırdan kapalı ve üst sınırdan açık olmaktadırlar. Programda nGrid ve alpha (*İnflation Rate*) parametrelerini şebekeleme fonksiyonunu yazmak için tanımlandı. Bunun yanında her parçacığın şebekelemedeki pozisyonunu belirlemek için bir başka fonksiyon da yazılması gerekmektedir. Sıradan parçacıkların pozisyonu önemi olmamakta ve yalnız deponun içindekileri önemlidir. Amaç fonksiyonu sayısına göre şebekeleme yapılır ve problemin iki

tane olduğuna göre iki boyutlu uyzayın şebekelemesi yapılmakta ve her hane için iki tane alt ve iki tane üst sınırları tanımlanır. Eğer üç boyut söz konusu olursa üçer tane alt ve üst sınırları tanımlanacaktır. İki boyut olunca küp ve üzeri ise *HyperCube* şebekelemesi söz konusu olacaktır. Aslında her boyut için bağımsız olarak şöyle birşey vardır:



Şekil 99. Her boyut için *grid* uygulaması

$C_{j\,min}$, $C_{j\,max}$ aralığında dört sayı aritmetik ilerleme (*Arithmetic Progression*) aracılığıyla buluyoruz ve bu *MATLAB*'da *linspace* komutuyla gerçekleşmektedir. Sağ ve solda. -∞ ve ∞ olduğuna göre alt sınır (*LowerBand*) [52,42] esas alınırsa birinci hanenin alt sınırı -∞ olur:

$$LB : -\infty, \boxed{C_1, C_2, C_3, C_4}$$

Belirtilmiş sayıya göre bölümleme yapılır ve *UpperBand* [53,42] (UB) esas alınırsa yani birinci hanenin üst sınırı C_1 olur ve şöyle bölümleme yapılır:

$$UB : \boxed{C_1, C_2, C_3, C_4}, \infty$$

Aslında bölümlemede bunlar yanlız kesilme noktaları olmaktadırlar ve LB'lerin başına -∞ ve UB'lerin sonuna ∞ eklenilmesini görmekteyiz. Bu işlemler her boyut için yapılması gerekmektedir. Şimdi programlanan şebekeleme fonksiyonunu çağırarak programı koşturuyoruz:

```
Grid=CreateGrid(rep,nGrid,alpha);
```

Elde edilen deponun elemanları, nGrid = 5 ve alpha = 0. 1 tutarak program çıktısı şöyle raporlanır:

Grid =

2x1 struct array with fields:

LB

UB

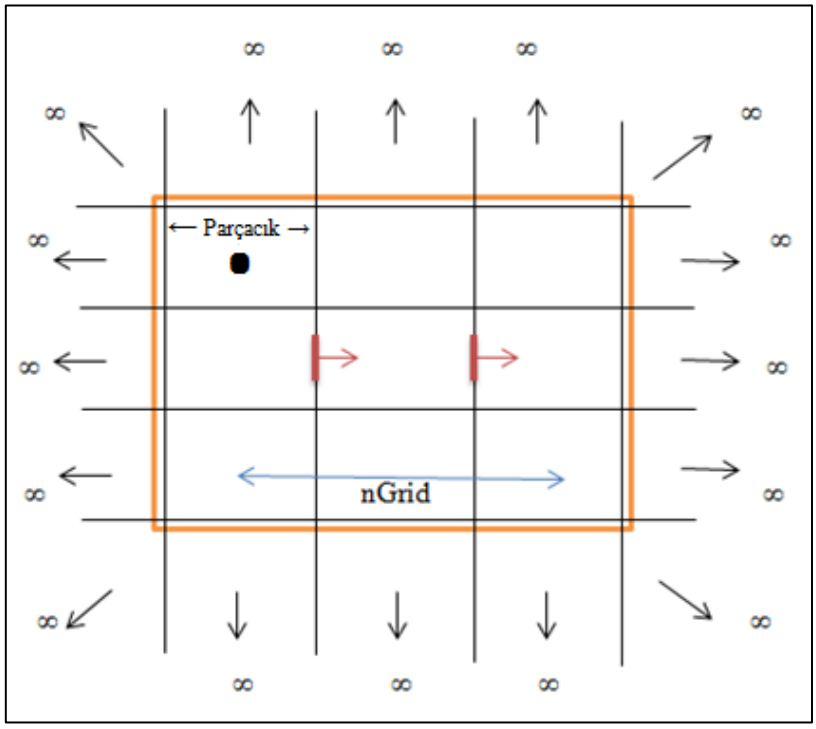
Grid adında bir dizi oluştuğunu görmekteyiz. Her amaç fonksiyonu bir boyut sayılır ve problemin iki amaç fonksiyonu olduğuna göre iki boyutta şebekeleme yapıldığını da görmekteyiz. Tablo 14'te gösterildiği gibi *Grid* iki *Struct* dizisinden oluşur ve her *Struct* dizisi bir boyuta aittir, Ayrıca her *Struct* dizisi içinde ilgili boyutuna ayit olan *LB* ve *UB* dizilerini'de saklamaktadır. Birinci boyut için şöyle :

Tablo 14. Şebekelemenin alt sınırı (LB) ve üst sınırı (UB) ile raporlaması

```
>> Grid(1,1)

ans =
LB: [1x7 double]
UB: [1x7 double]

ans = LB : -Inf 1. 2073 1. 8162 2. 4250 3. 0338 3. 6426 4. 2515
ans = UB : 1. 2073 1. 8162 2. 4250 3. 0338 3. 6426 4. 2515  Inf
İkinci Boyut İçin :

>> Grid(2,1)
ans =
LB: [1x7 double]
UB: [1x7 double]

ans = LB : -Inf  0. 0846 0. 3807 0. 6769 0. 9731 1. 2692 1. 5654
UB : 0. 0846 0. 3807 0. 6769 0. 9731 1. 2692 1. 5654  Inf
```

Böylece şebekeleme yapılmış oldu. Şimdi her parçacığın şebekelemedeki pozisyonunu ve her boyutta hangi grupta yer aldığını belirlemek gerekir. Bunun için popülasyondan bir elemanını programlanan ilgili fonksiyona girdi olarak verilir. Kaç çeşit bölümleme işlemi yapılabilmektedir her hane böyle (1,1), (1,2), (2,3) … örnekler ile ifade edilir. Bunu yaptıktan sonra her haneyi bir sayı ile gösterebilen durumu oluşturacağız. Her parçacığın hangi hanede olduğunu belirlemek için parçacığın maliyeti her hanenin alt sınırından büyük ya da eşit ve üst sınırından küçük olması gerekir ve bu durum parçacığın bu haneye ait olduğunu göstermektedir. Şekil 100'de gösterildiği gibi parçacık yatay boyutta hanenin alt sınırından büyük ve üst sınırından küçüktür ve düşey boyut açısından da alt sınırından büyük ve üst sınırından küçük olmaktadır. Söylendiği gibi eğer nokta

hanenin kenar sınır çizgiler üstüne düşerse ozaman her hanenin kenarı yanındaki haneden sayılmakta olacaktır.



Şekil 100.Şebekelemede her hanenin kenarı yanındaki haneden sayılmaktadır

Şimdi program *MATLAB*'da buraya kadarı koşturulur:

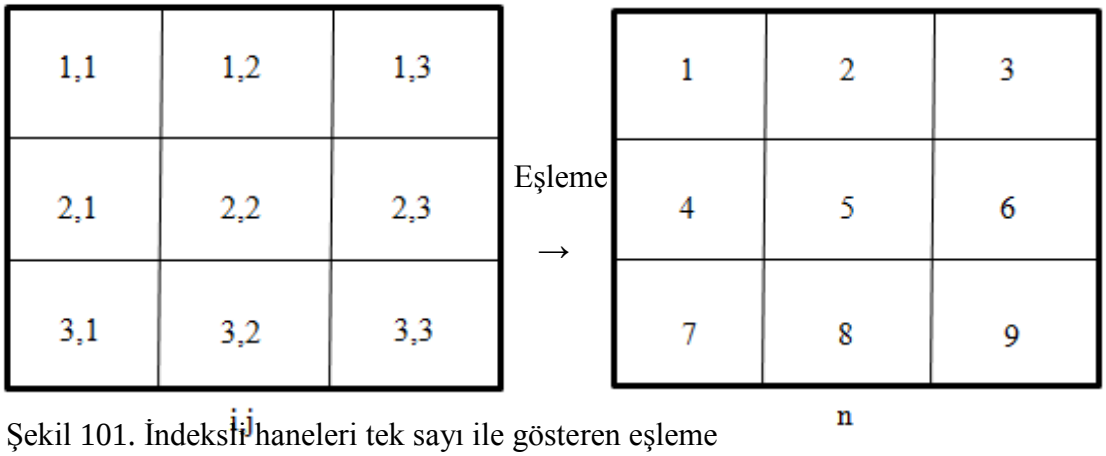
```
>> rep(1), ans =
  Position: [1 0 1 1 1 1 1 0 1 0 1 0 0 0 0 1 1 0 1 0]
  Velocity: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
   Cost: [2x1 double]
   Best: [1x1 struct]
   IsDominated: 0
   GridIndex: [ ]
   GridSubIndex: [2 6]
>> rep(2), ans =
  Position: [0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0]
  Velocity: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
  Cost: [2x1 double]
  Best: [1x1 struct]
```

IsDominated: 0

GridIndex: [ ]

GridSubIndex: [4 3]

Yukarıda rep(1)'e baktığımızda satır 2 ve sütun 6, [4 3] hanesinde ve rep(2)'de [4 3] hanesinde olduğunu görmekteyiz. Asıl yapılan şebekeleme 5*5 boyutundadır ve genel olarak yani şişirilmiş hali ile 7*7 boyutunda olmaktadır. Dolayısıyla toplam 49 hane bulunmaktadır ve bunları 1 ve 49 aralığında sayılarla gösterilebilir. Bu eşleme (*Mapping*) işlemini nasıl yapılacağına örnek getirerek incelemekte olacağız.



Şekil 101. İndeksli haneleri tek sayı ile gösteren eşleme

Yukarıdaki Şekil 101'de eşleme (*Mapping*) yapılır yani indeksli haneleri tek sayı ile gösteren biçime dönüşümü aralarında kurulan bir ilişkiden geçmektedir.

$$n = 3 * (i - 1) + j \tag{127}$$

Aslında bu işlem bundanda daha basit yapılabilir çünkü üç boyutta yapılması var sayılırsa ozaman sayılar daha karmaşık olur. Malesef *MATLAB*'da bu işlem için uygun fonksiyon bulunmamaktadır ve mevcut *Sub2ind, ind2Sub* komutların özel çağrılarıyla kullanması yinede genelde karmaşık bir durumu yaratmış olur. Bunun için bir algoritma geliştirilmesi gerekir. Algoritma şöyle aşağıda gösterildiği gibi birinci indeks i'den bir birim eksilir sonraki indekse gidince bu indeksin alabileceği mükün değerleri sayısına yani 3'e çarpılır ve sonra yine sonraki indeksin değeri ile toplanır.

i

$i - 1$

$3 * (i - 1)$

$3 * (i - 1) + j$

Bir tane daha olsaydı şöyle olurdu:

$$3 * (3 * (i - 1) + j - 1) + k \quad (128)$$

Burda özyineli ilişki (*Recursive relation*) kurulduğunu görmekteyiz yani eğer 3 bileşen olsaydı şöyle yazılır:

$$3 * (3 * (i - 1) + j - 1) + k \quad (129)$$

Matematiksel bakışını arttırmak için bunu şöylede yazabiliriz:

$$9 * (i - 1) + 3 * (j - 1) + k \quad (130)$$

$$3^{2} * (i - 1) + 3^{1} * (j - 1) + 3^{0} * (k - 1) + 1 \quad (131)$$

+i	i
-1	i-1
*3	3*(i-1)
+j	3*(i-1) + j
-1	3*(i-1) + j – 1
*3	9*(i-1) + 3*(j-1)
+k	9*(i-1) + 3*(j-1) + k

İfadenin sonunda hep +1 vardır. Belirtilmiş 3, *grid* sayısı olarak sayılmaktadır. Burda asıl *Grid* sayısı 5 ve şişirmiş hali ile 7 olmaktadır.

Özyineli yazdığımızda bir takım operatörlerin tekrarlandığını görmekteyiz. Bu sürecin programını yazarak şöyle:

$$i \rightarrow i_1, j \rightarrow i_2, k \rightarrow i_3 \tag{132}$$

İndeksleri bahsi geçen boyutlarda i . j . k yerine i_1, i_2, i_3 olarak kullanıyoruz. Bu eşlemenin sonucu bir sayıdır çünkü hesaplamalar hepsi sayı tabanlı yapılacaktır. Bu işlemin programlama kısmı kaçınılmazdır ve *MATLAB*'da buna göre hazır fonksiyon ya da komut bulunmamaktadır. Şimdi programı buraya kadar koşturuyoruz :

```
>>  rep(1), ans = Position: [1 0 1 1 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1]
    Velocity: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
    Cost: [2x1 double]
    Best: [1x1 struct]
    IsDominated: 0, GridIndex: 12, GridSubIndex: [2 5]
```

Yukarıda gördüğümüz gibi rep(1) satırı iki sütunu beş [2 5] olmaktadır. Aslında şebekeleme 7*7 boyutunda şöyle hesaplanmiştır.

$$7 * (2 - 1) + 5 = 12 \tag{133}$$

Eğer objective sayısı artarsa program kendiliğinden ayarlanacaktır. Burda *Objective* sayısı 2 tane olduğuna göre el ile kolay hesaplama yapmış olduk. Ama programın genelleştirilmesi (*generalization*) kabiliyetine sahip olması gerekmektedir. Böylece bunuda yapmış olduk ve algoritmanın en önemli kısmı gerçekleşti ve geriye pek fazla karmaşık birşey kalmamaktadır. Şimdi programın ana döngüsüne (*Main Loop*) kısmına girilmektedir. *Leader* seçimi bahsettiğimiz gibi özetleme yaparsak ilk başta tüm *Gridindex*'ler bulundu sonra işgal edilmiş (*Occupied Cell*) haneler tespit edildi. Sonra işgal edilmiş hanelerin popülasyonu hesaplandı daha sonra her işgal edilmiş hanenin seçilmesi ihtimali hesaplandı, daha sonra Rulet Tekerleği Tekiniği ile işgal edilmiş hanelerden bir tanesi seçildi daha sonra seçilen hanenin elemanları çıkarıldı ve bir tanesi seçildi sonrada depodan buna denk gelen elemanını *Leader* olarak seçildi.

Leader seçimi leader=SelectLeader(rep,beta) isimli fonksiyonu *Beta* (*Selection Pressure*) ve *rep* (*Repository*) parametreleri ile çalışır yani *rep*'ten *Beta* miktarında seçim baskısıyla lider seçimi yapılmaktadır. Artık *Global Best* yerine *MOPSO* algoritmasında *Leader* kullanılır ve programın geriye kalanı klasik *PSO* algoritmasına benzemektedir. *MOPSO* programın koşumunda programı *MaxIt*=100 ve nPop=100 değerleri ile koşturulur. Rapora göre *Non-Dominated* cevapları yani deponun eleman sayısı 27 tane mağlup olmayan cevaplar olduğu görülebilmektedir. *Hub Location* probleminin 27 tane cevabı (Opsiyonu) vardır. Aslında bir *Pareto Front* çok amaçlı probleminin cevapları olduğunu da görüş olduk. Burda deneyimlere baktığımızda *MOPSO* oldukca *NSGA-II* 'ye göre hızlıdır yani *NSGA-II* algoritma programda *Maxİt*=100 ve nPop=100 değerleri ile koşturulursa, aşikar nekadar yavaş olduğunu görmek mümkündür.

Tablo 15. Problemin *MOPSO* ile çözülen cephenin cevaplarının durumu

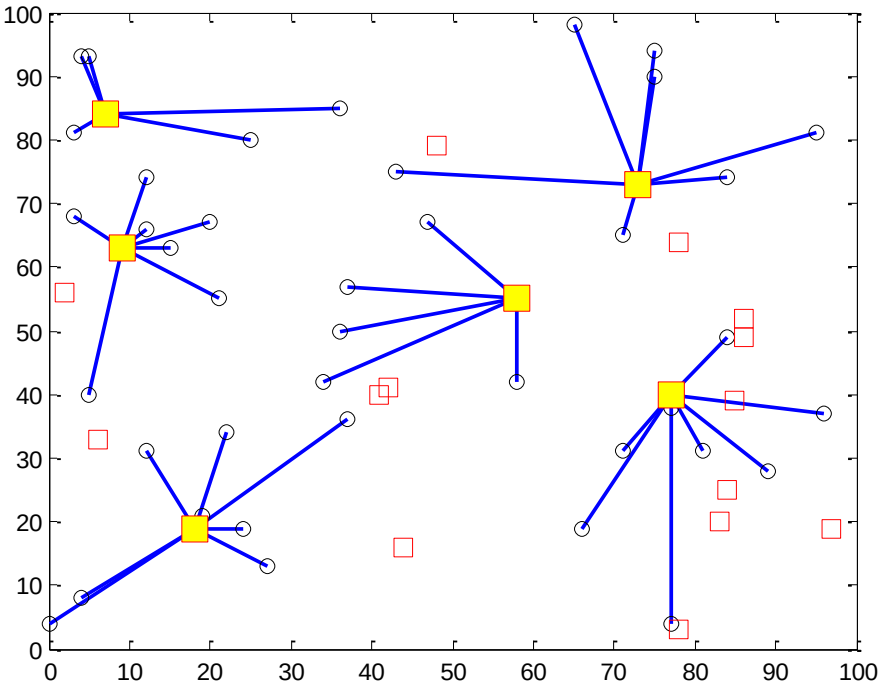
Number of Repository Members = 27, Element = 1 Solution = [0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0]
Number of Repository Members = 27, Element = 2 Solution = [0 0 1 1 0 1 1 0 1 0 1 0 0 0 0 1 0 1 0 0]
Number of Repository Members = 27, Element = 3 Solution = [1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1]

Number of Repository Members = 27, Element = 4 Solution = [1 0 1 1 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1]
Number of Repository Members = 27, Element = 5 Solution = [1 1 1 1 0 0 1 1 0 1 1 1 0 1 1 1 1 0 1 0]
Number of Repository Members = 27, Element = 6 Solution = [0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1]
Number of Repository Members = 27, Element = 7 Solution = [1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0]
Number of Repository Members = 27, Element = 8 Solution = [0 0 1 1 0 0 1 0 1 1 1 1 0 0 0 0 1 0 0 0]
Number of Repository Members = 27, Element = 9 Solution = [0 1 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0]
Number of Repository Members = 27, Element = 10 Solution = [1 0 1 1 0 0 1 0 1 1 0 0 1 0 0 1 0 0 0 1]
Number of Repository Members = 27, Element = 11 Solution = [1 1 1 1 0 1 1 0 0 1 0 0 1 0 1 1 1 0 0 0]
Number of Repository Members = 27, Element = 12 Solution = [0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0]
Number of Repository Members = 27, Element = 13 Solution = [0 0 1 0 0 0 1 0 1 0 1 1 1 0 0 0 1 0 0 0]
Number of Repository Members = 27, Element = 14 Solution = [0 0 1 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0]

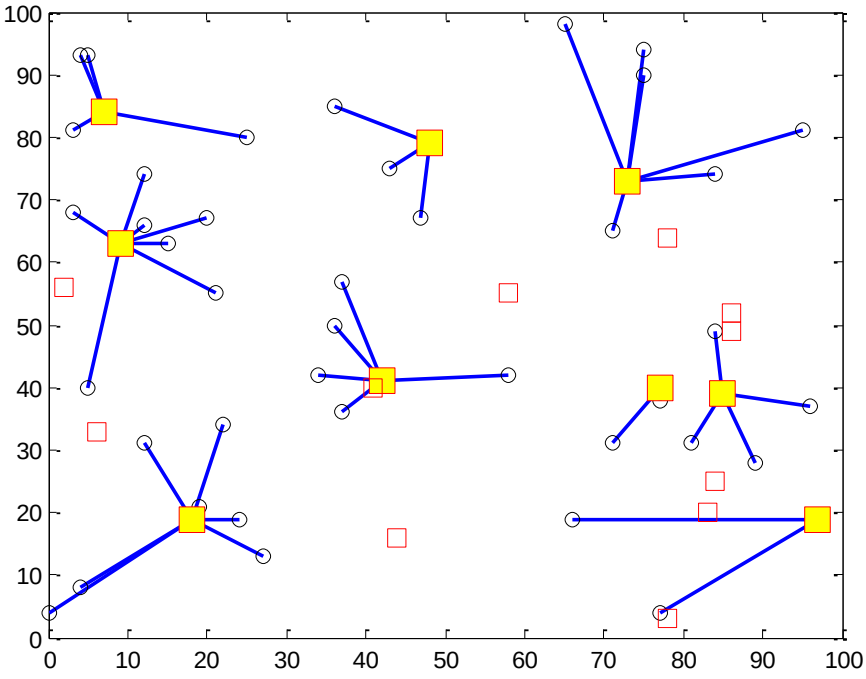
Number of Repository Members = 27, Element = 15 Solution = [0 1 1 1 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 1]
Number of Repository Members = 27, Element = 16 Solution = [0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1]
Number of Repository Members = 27, Element = 17 Solution = [0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
Number of Repository Members = 27, Element = 18 Solution = [0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
Number of Repository Members = 27, Element = 19 Solution = [1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1]
Number of Repository Members = 27, Element = 20 Solution = [1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 0]
Number of Repository Members = 27, Element = 21 Solution = [1 0 1 0 1 0 1 0 1 0 0 0 1 0 0 0 0 1 0 1]
Number of Repository Members = 27, Element = 22 Solution = [0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0]
Number of Repository Members = 27, Element = 23 Solution = [0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0]
Number of Repository Members = 27, Element = 24 Solution = [0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1]
Number of Repository Members = 27, Element = 25 Solution = [0 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1]

Number of Repository Members = 27, Element = 26 Solution = [1 0 1 0 0 1 1 0 1 0 1 0 0 1 0 1 1 0 1 0]
Number of Repository Members = 27, Element = 27 Solution = [1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0 0 1]

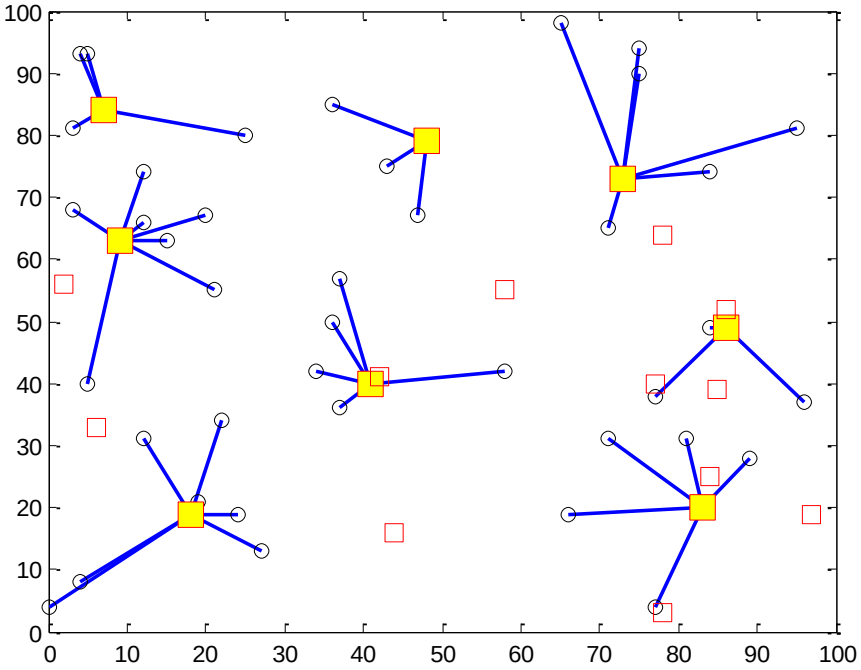
Aşağıdaki şekillerde elde edilen *pareto* cephesinden seçilen bazı cevaplara göre kümelenme yapısı gösterilmektedir. Elde edilen cevap sayısı 27'dir. Bu cevaplardan birincisi *Matlab* ortamında *Solution = [0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0]* biçimindedir ve bu cevaba ilişkin dağılım Şekil 102'deki gibidir. Yine elde edilen cevaplardan ikincisi, onuncusu, onbeşincisi, Yirmincisi, ve yirmiyedincisine ilişkin dağılımlar sırasıyla Şekil 103, Şekil 104, Şekil 105, Şekil 106 ve Şekil 107'de gösterilmektedir. Bu cevapların *Matlab*'taki çıktıları da sırası ile *Solution = [0 0 1 1 0 1 1 0 1 0 1 0 0 0 0 1 0 1 0 1]*; *Solution = [1 0 1 1 0 0 1 0 1 1 0 0 1 0 0 1 0 0 0 1]; Solution = [0 1 1 1 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 1]* ; *Solution = [0 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1] ; Solution = [1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0 0 1]* biçimindedir.



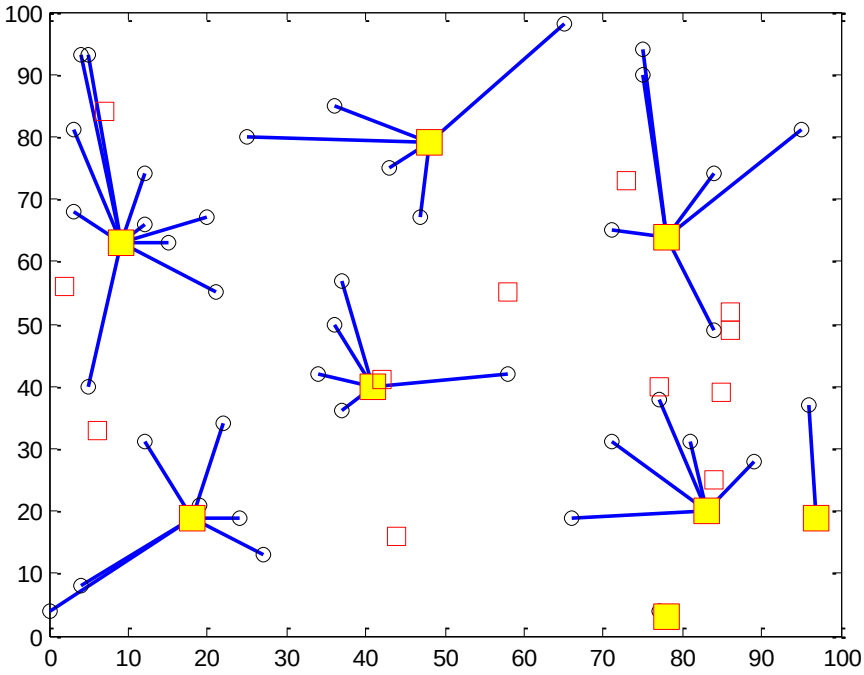
Şekil 102. Pareto cephesindeki birinci cevap



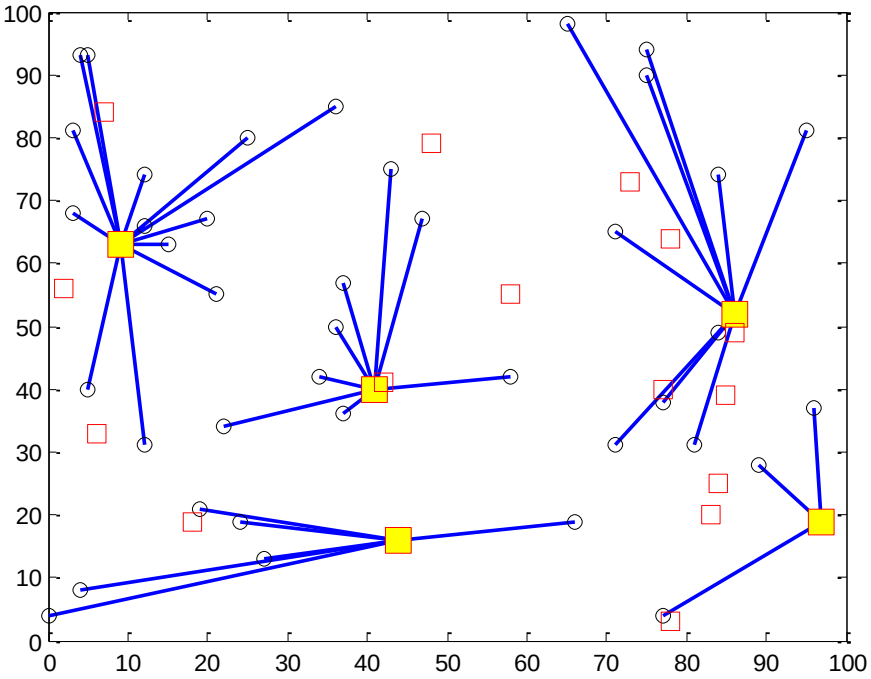
Şekil 103. Pareto cephesindeki ikinci cevap



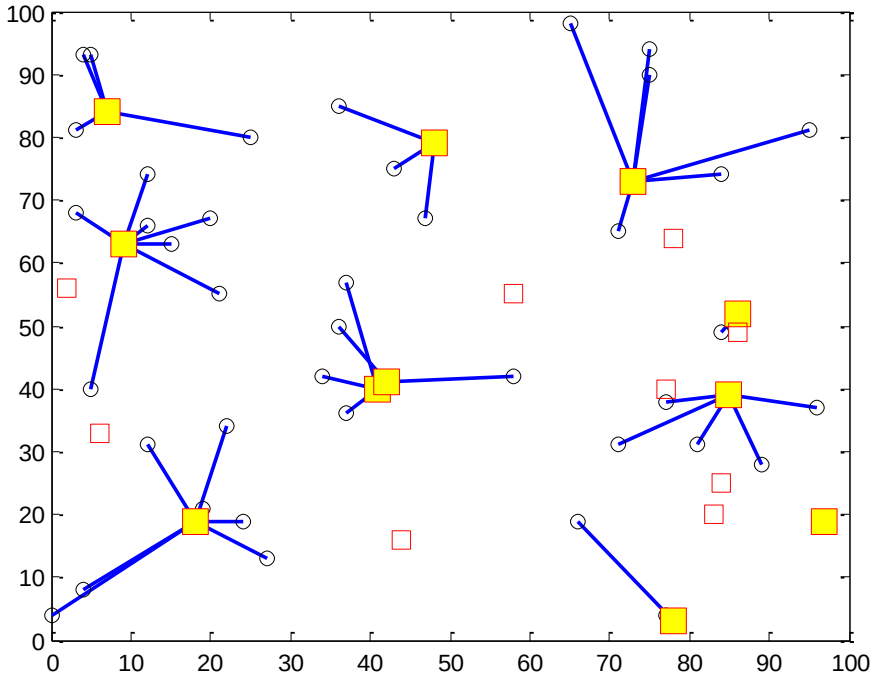
Şekil 104. Pareto cephesindeki onuncu cevap



Şekil 105. Pareto cephesindeki onbeşinci cevap



Şekil 106. Pareto cephesindeki yirminci cevap



Şekil 107. Pareto cephesindeki yirmiyedinci cevap

4. TARTIŞMA

MOPSO algoritması *NSGA-II* algoritmasına göre daha hızlı performansa sahiptir ve genelde sürekli problemler için daha uygun görülerek kullanılmaktadır, Ama kablosuz algılayıcı ağları kapsamında çevresel zeka (*Ambient intelligence*) ve esnek programlama (*Soft computing*) konuların açısından kullanılan algoritmanın işlem yükü, karmaşıklığının az olması ve hızlı olmasını gerektirmektedir. Algılayıcı düğümlerin kısıtlı işlemcilerine *MOPSO* gibi hızlı ve basit bir algoritma sonuçlara dayanarak uygun görülmektedir. *MOPSO* algoritmasının hesaplama işlemleri açısından *NSGA-II* algoritmasından daha hızlı çalışmaktadır. *NSGA-II*'ye Göre *MOPSO* sürekli problemlerinde daha hızlı hemde yüksek doğrulukla çalışmaktadır. *NSGA-II* hesaplama karmaşıklığı şöyle hesaplanmaktadır.

$NSGA - II\ Time\ Order\ O\ (M * N^2)$, M : *Objective* Sayısı, N : Popülasyon sayısı

Yani eğer popülasyon sayısı 10 kat olursa programın koşma zamanı 100 kat olacaktır. *NSGA-I* algoritmasına göre *NSGA-II* oldukca gelişmiştir. Genelde algoritmanın kendisinin hesaplama karmaşıklığı söz konusudur ve amaç fonksiyonunun karmaşıklığı bununla ilgisi olmamaktadır. Aslında *NSGA-II* algoritmanın her iterasyonunda $M*N^2$ defa amaç fonksiyonunu çağırmaktadır. Problemin çözümü için *NSGA-II* algoritmayı 50 popülasyon sayısı ve 100 iterasyon ayarlarıyla koşturulmuştur ve elde edilen sonuçlarda harcanan koşma zamanı (*Elapsed time*) 74,647890 saniye olarak raporlanmıştır. *MOPSO* Algoritmasını problem için aynı koşullarla koşturuldu ve koşma zamanı 23,341028 saniye olarak elde edilmiştir. Tekrar problem *NSGA-II* algoritmasını 200 iterasyon ve 100 popülasyon sayısıyla koşturuldu ve koşma zamanı 432,21012 saniye olarak elde edilmiştir. *NSGA-II* algoritmanın popülasyon sayısının artışıyla hız performansı oldukca düşmektedir. Ayrıca işlem yükü ve hesaplama karmaşıklığı da bunlara bağlı artmaktadır. *MOPSO* algoritmasını aynı koşullarla koşturulmuştur ve 61,854769 saniye olarak elde edilmiştir. Sonuçlara göre *MOPSO* nerdeyse yedi kat daha hızlı bir performans sağlamıştır. Metasezgisel algoritmalar rastgele doğalarından kaynaklı bazen her koştuklarında biraz farklı cevap üretebilmektedirler. Ama bu çalışmanın amacı tanımlanan problemin mağlup olmayan cevapları ya da *Pareto*'sunu bulmaktır. Çevresel zeka (*Ambient intelligence*) bir çeşit yapay zeka (*Artificial intelligence*) sayılmaktadır ve basit işlemciler üzerinde uygulanmaktadır. Dolayısıyla bunun için algoritma esnek programlama açısından basit ve etkin kullanılmalıdır.

5. SONUÇLAR

Tanımlanan problem zaten bir çok amaçlı problemin türünden olduğuna göre ilk başta dekompozisyon tabanlı tek amaçlı bir algoritma ile çözümlemeye çalışılmıştır. Ama önceki bölümlerde görüldüğü gibi bu yöntem doğrusal bir şekilde problemin fenotip alanını süpürerek keşfetmiştir. Dolayısıyla *pareto* cephesinin iç bükey kısımları çözüm dışında kalmıştır. Ayrıca fenotip alanı iki boyutlu bir düzlem olduğuna göre sıralama işlemi (*Sorting*) eski kriterlere göre yapılabilmektedir. Diğer taraftan *pareto* cephesinin her çözümünü bulmak için algoritma tekrar baştan koşturulmalıdır. Bu problemleri aşmak için tanımlanan problemi doğrudan çözmek amacıyla çok amaçlı *NSGA-II* ve *MOPSO* algoritmaları kullanılmıştır. Çözülen problemde hem düğümlerin mesafeleri ile istek sayıları minimize edilmiştir hem de küme başlarının kuruluş maliyeti optimum yapılarak kümelenmiştir. Sonuç olarak düğümlerin işlemcileri basit ve sınırlı olduğuna göre kullanılan algoritmalar hız, işlem yükü ve hesaplama karmaşıklığı açısından birbiriyle kıyaslanarak hızlı ve basit olanı problemin çözümü için bu tez kapsamında önerilmiştir.

6. ÖNERİLER

Bu çalışmada düğümlerin ve küme başlarının koordinatları biliniyor ve sabittir. Bunun mantığını biraz değiştirerek farklı senaryolar da incelenebilmektedir. Benzetim yapılabileceği senaryolardan biri bu çalışmayı dinamik koordinatlar ile yapılmasıdır. Yani düğümlerin koordinatları artık sabit değildir ve seçilen senaryoya bağlı sürekli ya da belli aralıklarla değişebilir ve bunu gerçekleştirmek için koordinatlar seçilen senaryonun karakteristiğine bağlı ya da rastgele üretilebilir. Yapılabilecek başka bir senaryo küme başlarının aktivasiyon enerjisi senaryonun karakteristiğine bağlı değişebilmesidir. Eğer bir düğüm bozulursa ya da devreden çıkarsa bu durumda diğer düğümler nasıl yerleri seçilir ve kümelenir problemi söz konuları olmak üzere incelenebilmektedir. Bu çalışmada küme başlarına kapasite sınırı (*Uncapacitated Single Allocation Hub Location Problem SAHLP*) [2] koyulmamıştır. Dolayısıyla küme başlarında kapasite sınırı (*Capacitated*) tanımlayarak problem incelenebilmektedir. Yine bir başka senaryo üç boyutlu senaryosu olabilir. Yani çukurlar ve cıkıntıları göz önünde bulundurarak bu tezin senaryosu incelenebilir. Örneğin: *Networked Infomechanical Systems* (*NIMS*) [61].

7. KAYNAKLAR

1. Özger, A. ve Oktal, H., Havayolu Kargo Taşımacılığında Kapasite Sınırı Olmayan Çok Atamalı P-ana Dağıtım Üssü Medyan Problemine Tamsayılı Model Yaklaşımı, havacılık ve uzay teknolojileri dergisi, ocak 2009, Bildiriler Kitabı : 47-60

2. Naeem, M., Using Genetic Algorithms for the Single Allocation Hub Location Problem, degree of Master of Science Faculty of Computer Science, Catherines Ontario, 2009.

3. Yamany, ve H. Carelloy, G., Solving the Hub Location Problem with Integer Links, 24, 10129.

4. Seungjun, L., Seong, K., A hub location-allocation problem in IMT-2000 Networks.

5. Çetiner, S., An iterative hub location and routing problem for postal delivery systems, degree of master of science in the department of industrial engineering, 2003.

6. Raghuvanshi, A. Tripathi, S. Tiwari, Kishor. M, Optimal Number of Clusters in Wireless Sensor Networks: An FCM Approach, National Institute of Technology Allahabad , 211004.

7. Andre S. And Ruela Raquel S., Andre L., Frederico ,G., Guimaraes Evolutionary design of wireless sensor networks based on complex networks.

8. Huang, A., Network Location Problems with Multiple Types of Facilities, degree of Doctor of Philosophy, Joseph L. Rotman School of Management University of Torontoc, 2005.

9. Bing Huang, R., Network Location Problems with Multiple Types of Facilities, Doctor of Philosophy, Joseph L. Rotman School of Management, Torontoc, 2005.

10. Abdollahi Demneh, S., A., Saeedeh, K., Ghandehari M., Location-allocation model for loss minimization in large-scale Emergency situation, interdisciplinary journal of contemporary research in business, 954.

11. Tahir Emre, K., Kablosuz Algılayıcı Ağlar ve Uygulamaları, http://webs. cs. berkeley. e du/nest-index. html, 1998.

12. Durukan Odabaşi, Ş., Halim ZAİM, A., Kablosuz Algılayıcı Ağlar ve Güvenlik Problemleri.

13. http://www.politekno.com/kablosuz-sensor-ag-sistemleri, PoliSense, Çevresel Gözlem Sistemi, Poli*Sense* - Akıllı Tarım Sistemi, Poli*Sense* - Endüstriyel Otomasyon Sistemi,

14. Derviş, K., Selçuk, Ö., Kablosuz Algılayıcı Ağlarında Yönlendirme Teknikleri, Akademik Bilişim'07 - IX. Akademik Bilişim Konferansı Bildirileri, 2007, Kütahya, Bildiriler Kitabı : 409.

15. Pekmezci, M., Kablosuz Algılayıcı Ağlarında Lokalizasyon Problemi Kablosuz Algılayıcı Ağları, Kasım 2010.

16. Pal,A., Localization Algorithms in Wireless Sensor Networks Current Approaches and Future Challenges, Network Protocols and Algorithms, 1943-3581 2010.

17. Xiao, Q., Range-free and range-based localization of wireless sensor networks Wireless sensor networks, Detectors Location, Publisher: The Hong Kong Polytechnic University, 2011.

18. Raghuvanshi, A., Tripathi, R., Kishor, N., Motilal Nehru Optimal Number of Clusters in Wireless Sensor Networks: An FCM Approach, IEEE, National Institute of Technology Allahabad , 211004.

19. Derviş, K., Selçuk, Ö., Kablosuz Algılayıcı Ağlarında Yönlendirme Teknikleri, Akademik Bilişim'07 - IX. Akademik Bilişim Konferansı Bildirileri, 2007, Kütahya, Bildiriler Kitabı : 409.

20. Dianati,M., Insop,S., and Mark,T., An Introduction to Genetic Algorithms and Evolution Strategies, Canada.

21. http://cs. felk. cvut. cz/~xobitko/ga/, Introduction to genetic algorithms with Java applets.

22. Kumar, A., Encoding Schemes in Genetic algorithm, International Journal of Advanced Research in, IT and Engineering, 2278-6244.

23. Bhaskar, H., Mihaylova, L. and Maskell, S., Human Body Parts Tracking Using Pictorial Structures and a Genetic Algorithm.

24. Ali, M., Pant, M., and Abraham, A., Simplex Differential Evolution, Network for Innovation and Research Excellence, 2259, 6, 5, 2009.

25. www. geatbx. com support@geatbx. com, GEATbx, Examples of Objective Functions by: Hartmut Pohlheim GEATbx version, December 2006.

26. Benuskova, L., Optimisation (Beyond classical search) ,COSC343: Lecture 15 , Reading AIMA 2nd ed. Ch 4. 3-4. 4 or AIMA 3rd ed. Ch. 4. 1 plus the box on p. 130.

27. Turrini, S., Optimization in Permutation Spaces, November 1996.

28. Kumar, A., Network Design Using Genetic Algorithm 63 CHAPTER 4 Genetic Algorithm, Registration No: 3893.

29. Whitley, D., A Genetic Algorithm Tutorial, Computer Science Department Colorado State University Fort Collins.

30. Randy, L., Haupt, Ellen, S., Practical Genetic Algorithms, John Wiley & Sons, 2004.

31. Kumar, R., Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms, International Journal of Machine Learning and Computing, 2012.

32. Gen, M,, Cheng ,R. , Genetic Algorithms and Engineering Optimization Engineering Design and Automation, 1999.

33. Randy, L., Haupt, Ellen, S., Practical Genetic Algorithms, John Wiley & Sons, 2004.

34. Schwefel, H., Wolpert, D., and W. G., Evolution and Optimum Seeking, IEEE Trans. Evol. Comput, 1:67–82. Wiley, 1997.

35. http://www. geatbx. com/docu/tutindex-04. html#P473_46938, Tutorial 5 Overview of GEA Toolbox Structure

36. Van Hoai, T., Discrete optimization, Faculty of Computer Science & Engineering HCMC University of Technology, 2011-2012.

37. Abdel-Rahman Hedar A., Studies on Metaheuristics for Continuous Global Optimization Problems, degree of Doctor, 2000.

38. Gupta, R., K., Genetic algorithms-an overview Dept. of Mechanical Engineering, 2006.

39. Meruane, V., Heylen, W., An hybrid real genetic algorithm to detect structural damage using modal properties, Mechanical Systems and Signal, YMSSP 2703, 2010.

40. Qu, B. and Suganthan, P. N., Constrained Multi-Objective Optimization Algorithm with Ensemble of Constraint Handling Methods, Singapore.

41. Marler, R., T., Arora, J., Survey of multi-objective optimization methods for engineering, Struct Multidisc Optim, 26, 369–395, 2004.

42. Coello Coello, C. A., B. Lamont, G. and A., Van Veldhuizen, D., Second Edition Evolutionary Algorithms for Solving Multi-Objective Problems, Genetic and Evolutionary Computation Series, Springer, 2007.

43. http://cw. felk. cvut. cz/doku. php/courses/a4m33bia/start, Kubalik, J. , Evolutionary Algorithms: Multi-Objective Optimization.

44. Jiang, S., Cai, Z., Zhang, J., Soon Ong ,Y., Multiobjective Optimization by Decomposition with Pareto-adaptive Weight Vectors, China.

45. Konak, A., David W., Alice E., Multi-Objective Optimization Using Genetic Algorithms: A Tutorial.

46. Zhang, J. Q. , Xu, F. and Xian-Wen, F. , School Decomposition of Multi-Objective Evolutionary Algorithm based on Estimation of Distribution, Applied Mathematics & Information Sciences An International Journal, 2013.

47. Qingfu,Z. , and Li, H. , MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition, ieee transactions on evolutionary computation, 11, 6, 2007.

48. James, P. I., Romero, C., Goal Programming, Encyclopedia of Information Systems, Volume 2, Elsevier Science (USA). All rights reserved, 2003.

49. Banerjee. D., Ranaghat ,Y., Goal programming approach to chance constrained multi-objective linear fractional programming problem based on taylor's series approximation, International Journal of Computers & Technology , 2, 2, April 2012.

50. Fonseca, C., and Flemingz, P., Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization Dept. Automatic Control and Systems Eng. University of Sheeld Sheeld S1 4DU, U. K.

51. Coello Coello, C. , A Short Tutorial on Evolutionary Multiobjective Optimization, Instituto Polit´ecnico Nacional No. 2508 Col. San Pedro Zacatenco M´exico, D. F., ccoello@cs. cinvestav. mx, 2001.

52. Stringer, H. S. Wu, A., Variable-Length Genetic Algorithms and an Analysis of Changes in Chromosome Length Absent Selection Pressure, For Submission to Evolutionary Computation, School of Computer Science, FL 32816.

53. Coello Coello,C. , Handling Multiple Objectives WithParticle Swarm Optimization, ieee transactions on evolutionary computation, 8, 3, 2004.

54. Montes de Oca, A., Particle Swarm Optimization Introduction, IRIDIA-CoDE, Universit´e Libre de Bruxelles (U. L. B.) May 7, 2007

55. Engelbrecht ,A. , Computational Intelligence An Introduction Second Edition University of Pretoria South Africa, John Wiley & Sons Ltd, 2007.

56. El-Bendary, N., Mostafa M, Fouad, A., Ramadan, R., Soumya Banerjee, and Aboul Ella Hassanien Smart Environmental Monitoring Using Wireless Sensor Networks Cairo University.

57. Fernández ,L., M., Blasco, J., Eduardo, H. ,Wireless Sensor Networks in Ambient Intelligence Montón Technologies for Health and Well-being Instituto ITACA Universidad Politécnica de Valencia.

58. Wireless Sensor Networks for Precision Agriculture, Methods and Experiences Novel Sensor Technologies for Plant Phenotyping September , 2012.

59. MATLAB: Structures and Cell Arrays Kipp Martin, University of Chicago Booth School of Business, February 16, 2012.

60. , http://www. lania. mx/~ccoello/, Coello Coello, Carlos A. These pages were created and are maintained by. Any contributions are welcome.

61. W. J. Kaiser, G. J. Pottie, M. Srivastava, G. S. Sukhatme, J. Villasenor, D. Estrin, Networked Infomechanical Systems (NIMS) for Ambient Intelligence

62. Hussain . S . Abdul W. Matin Energy Efficient Hierarchical Cluster-Based Routing for Wireless Sensor Networks, Nova Scotia, CanadaTechnical Report, TR-2005-011.

8. EKLER

EK 1. Birinci Ekler Listesi

```
function model=CreateModel()


% Sensor Data
d=[34 18 47 23 8 48 35 18 13 31 11 16 22 10 26 48 30 49 46 38
36 25 7 8 49 7 22 12 41 24 29 6 17 14 45 14 32 28 29 13];

xc=[12 12 12 15 77 3 21 0 3 5 47 95 81 5 19 96 36 58 37 43 34
25 75 65 89 22 71 36 4 84 75 24 20 66 27 84 71 37 4 77];

yc=[31 66 74 63 38 68 55 4 81 93 67 81 31 40 21 37 85 42 57
75 42 80 90 98 28 34 31 50 8 49 94 19 67 19 13 74 65 36 93
4];
N=numel(d); % N=40 tane node

% Cluster head Data
c=[11372 9128 8557 9841 9689 11206 9536 10839 11410 9878
10097 10921 8626 11934 10348 11505 10278 10314 9962 8060];
c=randi([8000 12000],1,M);

xs=[85 78 18 9 86 86 7 44 73 77 41 6 42 58 78 48 84 83 2 97];
ys=[39 3 19 63 52 49 84 16 73 40 40 33 41 55 64 79 25 20 56
19];
M=numel(xs); % M=20 tane Cluster head
 D=zeros(N,M);
 for i=1:N
  for j=1:M
  D(i,j)=norm([xc(i)-xs(j) yc(i)-ys(j)],1); % Norm
  end
 end


 model. N=N;
 model. M=M;
 model. d=d;
 model. xc=xc;
 model. yc=yc; --> Structure Yapısı
 model. c=c;
 model. xs=xs;
 model. ys=ys;
 model. D=D;

end
```

Ek 2. İkinci Ekler Listesi

```
function [z sol]=MyCost(f,model)

 global NFE;
 if isempty(NFE)
  NFE=0;
 end

 NFE=NFE+1;

 if all(f==0) % Logical indexing
  z=inf; % hic aktif Kume basi olmazsa Cost Sonsuz olur!
  sol=[];
  return;
 end

 N=model. N;
 D=model. D;
 d=model. d;
 c=model. c;

 M=model. M;

 Dmin=zeros(1,N);
 A=zeros(1,N);
 for i=1:N

[Dmin(i), A(i)]=min(D(i,:). /f); % A(i) assignment:hangi node
hangi Kume basina baglidir!

%%%%%%%% WSN Heinzelman et al Radyo ileyişim modeli

 distance=Dmin(i);

 end

NumberOfCluster=sum(f); % Kromozomlarin toplami kumelerin
sayisini belirler

disp(['Number Of Cluster ' num2str(NumberOfCluster) ]);

el=0. 0013*10^-12; % Energy consumed by the amplifier to
transmit at a longer distance
es=10*10^-12;  % Energy consumed by the amplifier to transmit
at a shorter distance
Ee=50*10^-9; % Energy consumed in the electronics circuit to
transmit or receive the signal
Ebf=5*10^-9; % Energy consumed for beam forming
```

Ek 2'nin devamı

CreateRandomSolution. m

```
function f=CreateRandomSolution(model)

 % M=20 tane rastgele küme başı üretiliyor
 M=model. M;

 f=randi([0 1],1,M);

end
```

CalcCrowdingDistance. m
Dominates. m
NonDominatedSorting. m
SortPopulation. m
PlotSolution. m
PlotCosts. m
MyCost. m

```
 z1=sum(d. *Dmin); % Objective Function 1
 z2=sum(f. *Energy); % Objective Function 2

 z=[z1 z2]';
```

Mutate. m
Crossover. m

ÖZGEÇMİŞ

Vahid (Vahit) FARYAD AGHJEH KAND, 1983 yılında Tebriz'de doğdu. Liseyi Khrazmi Lisesi'nde okudu. 2005 yılında Seraj Üniversitesi Elektronik Teknisiyenlilği ve 2010 yılında İCT (*İnformation Communication Technology*) Bilgi ve İletişim Teknolojisi Mühendisliği Bölümü'nü bitirdikten sonra, 2010 yılında Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim dalında yüksek lisans yapmaya başladı. Anadili Azerbaycan Türkçesi olmakla beraber, Türkçe, İngilizce ve Farsça dillerini çok iyi derecede bilmektedir.