GENETİK ALGORİTMA KULLANARAK RADYO YÖN BULMA SİSTEMLERİNİN KONUŞLANDIRILMASI

RADIO DIRECTION FINDING BASELINE SELECTION VIA GENETIC ALGORITHMS

NEBİ VURAN

Hacettepe Üniversitesi

Lisansüstü Eğitim – Öğretim ve Sınav Yönetmeliğinin

ELEKTRİK ve ELEKTRONİK Mühendisliği Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2007

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından **ELEKTRİK ve ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI 'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan	:	Prof. Dr. A. Salim Kayhan
Üye	:	Prof. Dr. Adnan Köksal
Üye	:	Doç. Dr. Engin Tuncer
Üye (Danışman)	:	Yrd. Doç. Dr. Yakup Özkazanç
Üye	:	Yrd. Doç. Dr. Semih Bingöl

ONAY

Bu tez ...../...../..... tarihinde Enstitü Yönetim Kurulunca belirlenen yukarıdaki jüri üyeleri tarafından kabul edilmiştir.

…./…./2007

Prof.Dr. Erdem Yazgan
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

GENETİK ALGORİTMA KULLANARAK RADYO YÖN BULMA SİSTEMLERİNİN KONUŞLANDIRILMASI

Nebi Vuran

ÖZ

Bu tez çalışmasında, radyo yön bulma teorisi, radyo yön bulma sistemleri ve bu sistemler yardımıyla elektromanyetik yayın yapan bir hedefin konumunu belirleme ve hedeflerin rasgele dağıldığı bir bölge için radyo yön bulma sistemlerinin en uygun yerleşim planı problemi incelenmiştir. Konum belirleme konusunun, en-küçük kareler yöntemini kullanarak matematiksel modeli çıkartılmış ve tezin ana konusu olan radyo yön bulma sistemlerini optimal olarak konuşlandırma problemi ayrıntılı olarak incelenerek genetik algoritma tabanlı bir yazılım geliştirilmiştir. MATLAB üzerinde çalışan bir ara-yüz olarak geliştirilen yazılım, genetik algoritmalar yardımı ile radyo yön bulma sistemlerinin verilen bir bölge içinde konum belirleme hatalarını en aza indirecek şekilde konuşlandırılmasını sağlamaktadır.

Anahtar Kelimeler: Radyo Yön Bulma, Radyo Yön Bulma Sistemleri, Konum Belirleme, Radyo Yön Bulma Sistemleri Konuşlandırma, Genetik Algoritmalar, Elektronik Harp, Elektronik Destek Sistemleri.

Danışman: Yrd. Doç. Dr. Yakup ÖZKAZANÇ, Hacettepe Üniversitesi, Elektrik ve Elektronik Mühendisliği Bölümü

RADIO DIRECTION FINDING BASELINE SELECTION VIA GENETIC ALGORITHMS

Nebi Vuran

ABSTRACT

In this thesis, radio direction finding theory, radio direction finders, geolocation of emitters, and RDF baseline selection problem for randomly deployed emitters have been investigated. In order to find optimal baseline selection for a given arbitrary emitter region the mathematical model of problem has been developed by using the least-squared estimation (LSE) algorithm and to simulate diffferent scenarios a software has been written which is based on genetic algorithms. The developed software run on MATLAB uses genetic algorithms as an optimization method and tries to minimize the errors of position fixing of RDF systems for a given regions that is determined by the user of program.

Keywords: Radio Direction Finding, Radio Direction Finders, Position Fixing, Fix Coverage, Genetic Algorithms, Electronic Warfare, Electronic Support Measures (ESM).

Supervisor: Asst. Prof. Yakup ÖZKAZANÇ, Hacettepe University, Department of Electrical and Electronics Engineering

TEŞEKKÜR

Tez çalışması süresince karşılaşılan zorlukların aşılmasında daima yol gösterici olan, değerli yardım ve katkılarını esirgemeyen danışmanım Sayın Yrd. Doç. Dr. Yakup Özkazanç'a

Sağladığı olanaklar ve gösterilen anlayış için HAVELSAN' a;

Sevgi ve desteklerini her zaman yanımda hissettiğim sevgili eşim Serpil'e, kızım Fatma İnci'ye ve aileme içtenlikle teşekkürlerimi sunarım.

İÇİNDEKİLER DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

SÖZLÜKÇE

Radio Direction Finding	:	Radyo Yön Bulma
Radio Direction Finder	:	Radyo Yön Bulucu
Electronic Support Measures	:	Elektronik Destek Sistemleri
Radar Warning Receiver	:	Radar Uyarı Alıcısı
Bearing Angle	:	Kerteriz Açısı
Line of Bearing	:	Kerteriz Hattı
Azimuth	:	Kuzey, ufuk açısı
Intermediate Frequency	:	Ara frekans
Time Difference of Arrival	:	Geliş Zamanı Farkı
Direction of Arrival	:	Geliş Yönü
Triangulation	:	Üçgenleme
Position Fixing	:	Konum Belirleme
Geolocation of Emitter	:	Yayın Yapan Hedefin Coğrafi Konumu
Geometric Dilution of Precision	:	Geometrik Konuşlandırmaya Bağlı Hassasiyet Kaybı
Orthogonality Principle	:	Dikgen İzdüşüm Prensibi
Circular Error Probable	:	Olası Dairesel Hata
Elliptical Error Probable	:	Olası Eliptik Hata
RDF Baseline	:	RYB Sistemlerinin Yerleşim Planı
Signal Intelligence	:	Sinyal İstihbaratı

Communication Intelligence	:	Haberleşme İstihbaratı
Electronic Intelligence	:	Elektronik İstihbarat

KISALTMALAR DİZİNİ

RYB : Radyo Yön Bulma

RYBS : Radyo Yön Bulma Sistemleri

ODH : Olası Dairesel Hata

OEH : Olası Eliptik Hata

OKH : Olası Küresel Hata

YP : Yerleşim Planı

GA : Genetik Algoritma

RDF : Radio Direction Finding

RDF : Radio Direction Finder

ESM : Electronic Support Measures

RWR : Radar Warning Receiver

LOB : Line of Bearing

IF : Intermediate Frequency

TDOA : Time Difference of Arrival

DOA : Direction of Arrival

GDOP: Geometric Dilution of Precision

CEP : Circular Error Probable

EEP : Elliptical Error Probable

SEP : Spherical Error Probable

SIGINT: Signal Intelligence

COMINT : Communication Intelligence

ELINT : Electronic Intelligence

1. GİRİŞ

Radyo Yön Bulma Sistemleri (RYBS), özellikle elektronik harp literatüründe kritik bir savaş bilgisi olarak değerlendirilen RF hedeflerin yönünü bulma ve konum belirleme amacıyla kullanılan sistemlerdir. Bu sistemler, tek başlarına da bir sistem olarak bulunabilirler veya Elektronik Destek Sistemleri (ESM – Electronic Support Measures), ya da Radar Uyarı Alıcısı (RWR -Radar Warning Receivers) gibi çok daha kapsamlı sistemlerin bir alt fonksiyonu olarak da gerçekleştirilebilirler.

Genel olarak literatürde, Radyo Yön Bulma (RYB) ve teknikleri ile ilgili birçok çalışma bulunmaktadır. Fakat konum belirleme ve belirli bir hedef bölgesinin kapsanması için kullanılan RYB sistemlerinin en uygun dağılımı, yerleşimi veya konuşlandırılması ile ilgili bir çalışmaya açık literatürde rastlanmamıştır. Bu tez çalışması ile bu boşluğun giderilmesine yönelik bir adım atılmaktadır.

Bu amaca yönelik olarak en uygun konuşlandırmayı veya yerleşim planını belirlemek için geliştirilen yazılımda optimizasyon algoritması olarak Genetik Algoritmalar (GA) kullanılmıştır. Genetik algoritmaların kullanılmasının nedeni problemin analitik olarak çözümünün zor olmasıdır. Geliştirilen yazılım ön-yüzünü kullanarak kullanıcı dilediği biçimde hedef bölgesini ve RYBS'lerin yerleştirileceği alanı iki boyutta çizebilmektedir. Hedef bölgesinin herhangi bir noktasında yayın yapan bir hedef olduğu varsayılmaktadır. Bundan dolayı hedef bölgesi küçük dikdörtgenlere bölünerek her bir dikdörtgenin orta noktasında bir hedef olduğu varsayılmıştır; kullanıcıya dilediği oranda dikdörtgenlere ayırma olanağı verilerek de hedef bölgesinin istenilen çözünürlükte kapsanabilmesi sağlanmaktadır. Kullanıcı belirlediği alanın içine genetik algoritma ile optimum yerleştirilecek olan dilediği kadar RYB sistemini kerteriz açısı ölçüm hassasiyet standart sapma değerleri ile ön-yüzde belirlenen alanlara girebilmektedir.

Genel olarak ise tez çalışması birinci bölümü giriş olmak üzere altı bölümden oluşmaktadır:

İkinci bölümde radyo yayınları kullanarak yapılan yön bulma işleminin temel kavramları ve radyo yön bulma sistemlerinin çalışma prensipleri anlatılmaktadır.

Üçüncü bölümde üçgenleme yöntemleri anlatılmaktadır. Ayrıca bu bölümde konum belirleme algoritmaları, istatiksel ve istatiksel olmayan yöntemler anlatılıp bu konuda ilk çalışma olan Stansfield'ın algoritması verildikten sonra tez çalışmasında kullanılan en küçük kareler yöntemi türetilmiştir. Bu bölümün son konusu olarak da RYB sistemleri için yerleşim planı seçiminin önemli hususları verilerek, verilen bir hedef bölgesini dinleyecek olan RYB sistemlerinin konuşlandırılması ile ilgili temel ilkeler anlatılmıştır.

Dördüncü bölümde ise programda kullanılan optimizasyon algoritması olan genetik algoritmaların temelleri verilip programın yapısı ve kullanım esasları anlatılmaktadır.

Beşinci bölümde ise belirlenen farklı senaryolar için program çalıştırılıp simülasyon sonuçları verilmiştir.

Son bölüm olan altıncı bölümde ise genel olarak sonuçlar tartışılıp sonraki çalışmalarda neler yapılabileceğine dair görüşler belirtilmektedir.

Simülasyonlarda kullanılan kaynak kodları eklerde verilmiştir.

2. RADYO YÖN BULMA ve RADYO YÖN BULMA SİSTEMLERİ

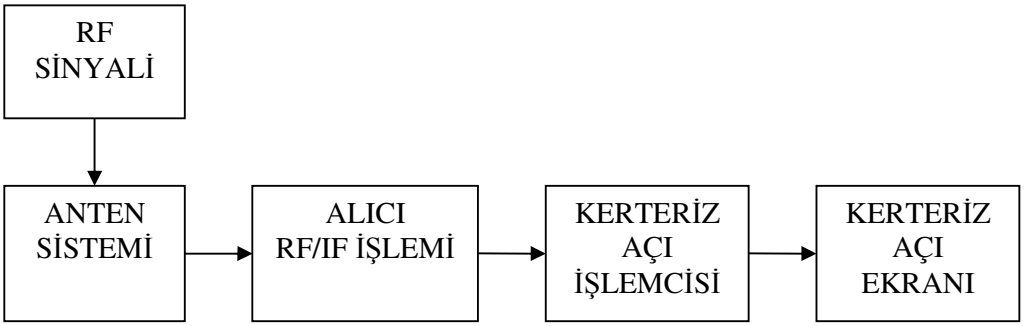
Bu bölümde [Herndon, 1991], [RF Products, 1998], [Güler, 2004] , [US Army Electronics Course, 1999], [Adamy, 2004] ve [Poisel, 2002] kaynaklarında verilen bilgiler esas alınarak, radyo yön bulma ve radyo yön bulma sistemleri ile ilgili genel bilgiler verilmiştir.

2.1 Radyo Yön Bulma Teknikleri ve Sistemleri İle İlgili Temel Kavramlar

Radyo Yön Bulma (RYB), yön bulma yöntemlerinden biri olup radyo yön bulma sistemlerini (RYBS) kullanarak belirli bir referans noktasına göre radyo kaynağının (radar, karıştırıcı) yönünü belirleme, radyo yayınını takip etme ve farklı yayınları birbirinden ayırt etme yöntemidir. Referans yönü genellikle stratejik sistemler için gerçek kuzey, taktik sistemler için manyetik kuzey seçilir.

RYBS ise RF enerjisi yaymayan bu yüzden de pasif sistem olarak kabul edilen üzerine düşen RF enerjisinin geliş yönünü belirlemede kullanılan sisteme verilen genel addır. Kronolojik olarak radyo yardımı ile yön bulma teknik ve sistemlerinin ortaya çıkışını radyonun Marconi tarafından icadına değin götürebiliriz. 20. Yüzyılın hemen başında kıta Avrupa'sında çıkan ve kısa sürede büyük bir coğrafyaya yayılan ilk dünya savaşında düşman sathında yayın yapan bir hedefin yerinin, yönünün kestirilmesi gibi amaçlar RYBS'lerini savunma sistemlerinden biri haline getirmiştir. Bu savaşın son yıllarına gelen bir tarihte, 1917 yılında vakumlu tüplerin geliştirilmesi RYBS teknolojilerine çok önemli katkı sağlamıştır. Daha sonra ortaya çıkan II. Dünya Savaşı ve Vietnam Savaşı, diğer elektronik harp sistem ve uygulamalarında olduğu gibi RYB sistemlerinin de önemini giderek artırmış ve gelişen teknoloji ile birlikte bu sistemler çok daha etkin ve sofistike cihazlar haline gelmişlerdir. Günümüzde yalnızca elektronik harp uygulamalarında değil birçok sivil uygulama alanlarında da radyo yön bulucular yaygın olarak kullanılmaktadır (Herndon, 1991).

Temel olarak bir RYB sistemi dört ana fonksiyonel bileşenden oluşmaktadır: Elektromanyetik dalganın alındığı anten, alıcı, kerteriz (yön) işlemcisi ve kerteriz açı monitörü.



Şekil 2.1 RYBS Temel Bileşenleri

Şekil 2.2 RYBS Kerteriz İşlemci ve Ekran Örneği

Bu bileşenlerden anten sistemi, RYBS'nin hassasiyet ve doğruluk derecesini baskın olarak belirleyen bileşen olduğu için tüm sistemin de ana elemanı olarak kabul edilmektedir. Anten sistemi yönselliği olan ve dönen tek bir anten (örneğin yagi anten) veya uzamsal olarak yayılmış sensör dizisinden (faz-dizili antenler) oluşabilir. Faz dizili antenler genel olarak SIGINT (signal intelligence) gibi çok daha özel amaçlar için kullanılmaktadır. Bu sensörlerin her birinin çıkış voltajı belirli bir faz/genlik karakteristiği üretir. Verici radyonun yönü hassas olarak anten çıkışlarının dizin sinyal işleme prensipleri ile analiz edilmesiyle belirlenebilir.

Sistemin ikinci bileşeni olan alıcı sistemi ise anten sisteminden aldığı sinyale standart sinyal işleme (ön-seçim, frekans çevrimi, IF filtreleme, demodülasyon) yöntemleri uygulayarak elde ettiği sinyali yön bilgisi işlemcisine iletir. Alıcı sistemin yapısı, RYBS'nin kullandığı tekniğe göre basit veya karmaşık olabilir. Örneğin tek-kanallı

genlik ölçümü kullanan RYB sisteminde alıcı, göreceli olarak antenin genlik cevabını ölçen RF voltmetre gibi düşünülebilecekken; tersine yön bulmada faz veya zaman farkını kullanan bir RYB sisteminin alıcı sistemi çok daha karmaşıktır.

Açı veya yön bilgisi işlemcisi, alıcıdan aldığı sinyal parametrelerine göre sinyalin geliş-açısının (yönünü) nihai kestirimini yapmakla görevlidir. Yine bu işlemcinin de karmaşıklığı (complexity) sistemde kullanılan YB tekniğine bağlı olarak artar veya azalır.

Sistemin son bileşeni olan monitör ise kullanıcıya yön bilgisini göstermek amacıyla tasarlanan bir ekrandır.

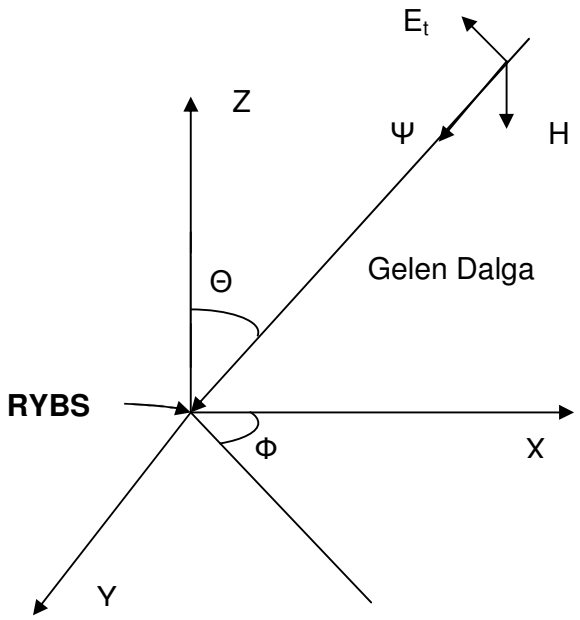
2.2 Radyo Yön Bulma İlke ve Yöntemleri

Şekil 2.3'te gösterilen uzamsal koordinat sisteminde RYBS'nin orijinde bulunduğu varsayılarak, geliş yön açısı yatayda ve yükseklikte iki bileşen olarak gösterilmektedir:

i-) gelen dalganın X-Y düzleminde X ekseni ile yaptığı açı, azimut (azimuth) açısı (Φ) veya kerteriz (bearing) açısı

ii-) gelen dalganın Z ekseni ve XY düzlemi ile yaptığı açı, yükseliş açısı (Θ) (elevation)

Genel olarak, coğrafi konumlandırmada X ekseni için kuzey, Y ekseni için doğu ve Z ekseni için yukarı yönler tercih edilmektedir.



Şekil 2.3 RYBS Örneği

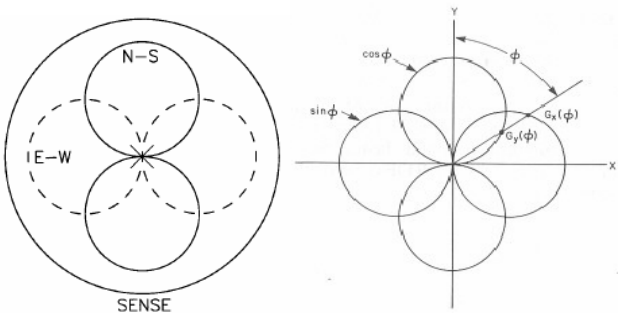
İdeal bir radyo yön bulma sistemi yatayda ve düşeyde olası herhangi bir yönden gelebilecek tüm elektromanyetik yayınları alabilmelidir; ufuk açısında (azimuth) 0^o - 360^o arası, yükselti açısında 0^o -90^o arası ölçüm yapabilmelidir.

Genel olarak radyo yardımı ile yön bulma teknik ve sistemlerinin geliştirilmesinde, gelen elektromanyetik yayının uzak bir noktadan doğrusal polarizasyona sahip düzlemsel bir dalga olarak RYBS üzerine düştüğü varsayımı yapılır. Temel olarak, bir elektromanyetik dalganın elektriksel alanı hem teğet, hem de merkez açılımlı (radyal) bileşenlere sahip olabilir. Bu bileşenlerden birinin sıfır olması dalganın doğrusal polarizasyona sahip olduğu anlamına gelmektedir. YB hesaplamalarında dalganın pozisyon vektörünün dik kaldığı kabul edilir, ki bu varsayım dalganın düzlemsel dalga olarak kabul edilmesine sebep olmaktadır. Düzlemsel dalga varsayımı da, elektriksel alanın radyal bileşenini sıfırlayarak, doğrusal polarizasyon elde edilmesini sağlar. Teğet elektriksel alan (E_t) ve teğet manyetik alan (H) uzayda hem birbirlerine dik hem de yayılma yönüne (Ψ) dik olarak hareket ederler (Herndon, 1991).

RYBS'ler temel olarak gelen sinyalin genlik, faz kayması, zaman gecikmesi, frekans kayması gibi parametrelerini kullanarak, yayının yataydaki geliş açısını, kerterizi üç farklı ölçüm yöntemi ile belirlemektedir. Bu yöntemler genlik cevabını, faz gecikmelerini ve zaman gecikmelerini kullanan yöntemlerdir.

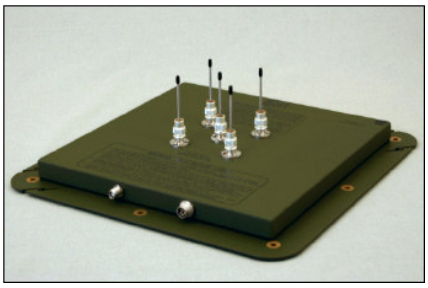
2.2.1 Genlik Cevabını Kullanarak Yön Tayini

Antenlerin genlik cevabını kullanarak veya genlik karşılaştırması yaparak yön bilgisini elde etme çalışmaları I. Dünya Savaşı sıralarında başlamış ve özellikle II. Dünya Savaşı sırasında yoğunluk kazanmıştır. Bu teknik tek bir halka (loop) anten ile ya da, yönselliği olan iki veya daha fazla anten kullanarak gerçekleştirilir ve Adcock/Watson-Watt, Wullenwebers YB tekniği olarak da bilinmektedir. Teorik olarak antenlerin her birinin birbirine göre göreceli genlik tepkileri, her bir geliş açısına göre tek bir şablon (pattern) oluşturduğu için yön bilgisi kolayca tayin edilebilir. Fakat pratikte her frekans değeri ve her kerteriz için anten üretmek mümkün olmadığından dolayı elde edilen yön bilgisi belirsizlik içerir. Dolayısıyla bu teknik çok hassasiyet istenmeyen uygulamalar için uygundur.



Şekil 2.4 Adcock Anten Genlik Tepkisi (Kazanç) Şablonu

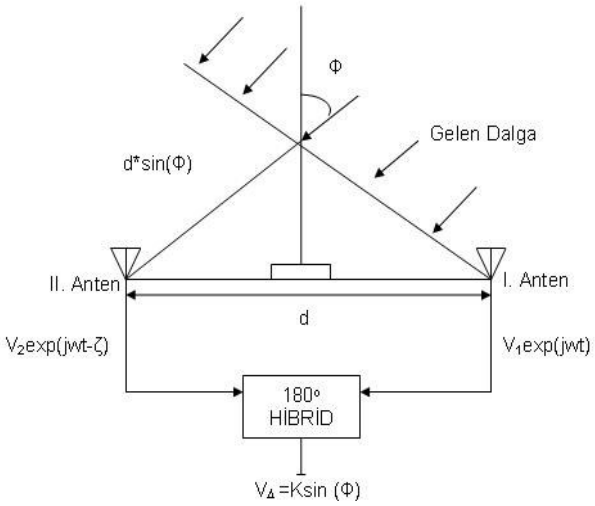
Şekil 2.4'te gösterilen genlik tepkisi şablonu, örnek olarak verilebilecek Sekil 2.5'teki Adcock anten sistemindeki her bir antenin çıkış gerilim değerlerinden elde edilebilir. Çıkış gerilim değerlerinin birbirleri ile orantılanması ile de geliş yönü açısı hesaplanır.



Şekil 2.5 Adcock Anten Örneği

2.2.2 Faz Gecikmelerini Kullanarak Yön Tayini

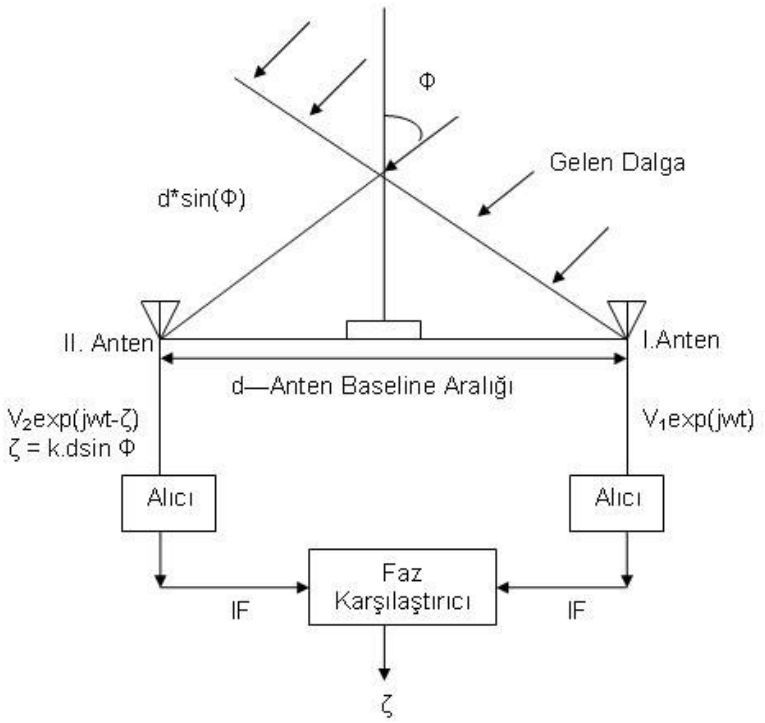
Faz gecikmeleri yön tayininde iki farklı şekilde kullanılmaktadır. Birinci yöntem yukarıda bahsedilen genlik cevabı yöntemine benzemektedir. Fakat burada genlik bilgisi (V_{Δ}), faz gecikmesinden (ζ) elde edilmektedir. Faz gecikmesi gelen dalganın antenlere farklı zamanlarda düşmesi sonucu oluşmaktadır ve antenler arası uzaklığa (baseline) , dalga boyuna ve geliş açısına bağlıdır.



Şekil 2.6 Faz Gecikmesi-Genlik YB Tekniği

Şekil 2.6'da gösterilen bir faz gecikmesini genliğe çevrim yaparak işlem yapan anten sistemi örneğidir. Bu yöntem Adcock YB tekniği olarak da bilinmektedir.

İkinci yöntem ise faz karşılaştırma yöntemidir, girişim-ölçme (inter-ferometre) yöntemi ve Doppler yöntemi olmak üzere ikiye ayrılmaktadır. Interferometre ölçümde yüksek çözünürlük isteniyorsa kullanılan pahalı bir yöntemdir, Şekil 2.7 üzerinden açıklanacak olursa: Gelen elektromanyetik dalga ilk önce I. Anten'e belirli bir geliş açısıyla varır, ikinci antene ise antenlerin aralarındaki uzaklıkla orantılı olarak d.sinΦ kadar daha yola kat ederek gecikmeli olarak düşer. Faz gecikmesini ζ ile gösterecek olursak $\zeta = k.d.\sin\Phi$ gecikmeyi formüle eder.



Şekil 2.7 Faz Karşılaştırması ile YB Tekniği

Dalga, II. Antene faz farkıyla geldiği için antenlerin çıkış gerilim değerleri birbirinden farklı olacaktır. Daha öncede belirtildiği gibi antenlerde oluşan göreceli faz farkı çıkış voltaj değerleri, olası her geliş yönü açısı için bir tanedir.

Faz karşılaştırıcı, her iki antenden gelen ve alıcı tarafından ara-frekans değerine indirgenmiş sinyalden faz farkını ölçer bu da kerteriz açısının, (Φ) elde edilmesini sağlar:

$\zeta = (2\pi d/\lambda)\text{Sin } \Phi$, $d <= \lambda /2$ (2.1)

d: antenler arası uzaklık, (λ/2 'ye eşit veya daha küçük olmalı belirsizlik olmaması için.)

λ :Gelen dalganın boyu, (m)

d ve λ'nın bilindiği varsayılıp ζ da ölçüldüğü için yukarıdaki formülden, Φ açısı

$\Phi = \arcsin(\zeta.\lambda / 2\pi d)$ (2.2)

ile hesaplanır.

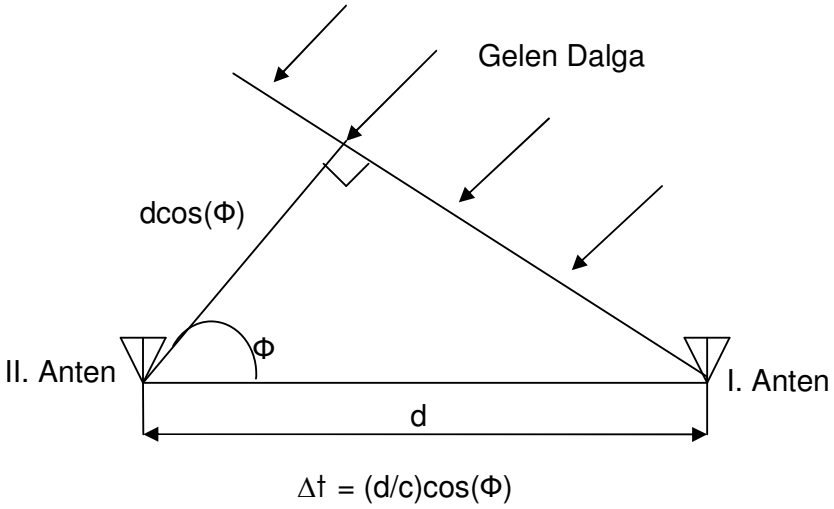
Diğer bir yöntem olan Doppler yöntemi faz karşılaştırma yöntemini kullanan en popüler tekniklerden biridir. Anten sisteminin döndürülmesi ile f_o frekansı ile gelen dalga, f_a (anten dönme frekansı) ile module edilir. Anten gelen dalgaya doğru veya ters istikamette farklı şekillerde hareket ettirilerek minimum ve maksimum Doppler frekansı elde edilir. Veya antenin kendi ekseni etrafında döndürülmesi ile şekli sinusoidal olan Doppler kayması elde edilebilir.

Pratikte mekanik olarak döndürme yerine çembersel dizili sensörlerin elektronik anahtarlama yapılması ile dönme etkisi verilmekte buna da Sözde-Doppler (Pseudo-Doppler) tekniği denmektedir.

2.2.3 Zaman Gecikmesini Kullanarak Yön Tayini

RYB sistemleri pasif cihazlar olduğu için yalnızca kendilerine gelen sinyalin geliş zamanını ölçebilirler, sinyalin hedeften çıkış zamanı doğal olarak bilinmemektedir. Bu yöntemde, birbirlerinden belirli bir mesafe ile ayrılmış antenlere gelen dalga antenlere farklı zamanlarda varacağından dolayı bir önceki bölümde anlatılan faz farkını kullanma yerine doğrudan zaman farkını kullanma tercih edilmektedir. Bu yöntem "Time Difference of Arrival" (TDOA), varış zamanı farkı yöntemi, olarak da adlandırılmakta ve bazı uygulamalarda çok kesin yön bilgisi verebilmektedir. Bu yöntemin matematiksel modelinde kerteriz çizgileri hiperboller oluşturduğu ve bu hiperbollerin kesişim noktaları radyo kaynağının konumunu verdiği için bu yöntemle çalışan sisteme hiperbolik yerbulum sistemleri de denmektedir.

Zaman farkı aynı zamanda faz farkını da ifade ettiği için bu yöntem de radyo interferometre olarak bilinmektedir. Eğer gelen dalga sürekli dalga değil de darbe (pulse) sinyali ise bu yöntemi kullanarak yön bilgisini bulmak her bir alıcı sistemde yüksek hassasiyette çalışan zaman ölçerler varolduğu için oldukça kolaydır.



Şekil 2.8 Zaman Gecikmesi ile YB Tekniği

Şekil 2.8 bir RYB sistemindeki iki antene farklı zamanlarda gelen dalgayı ve oluşan zaman farkının denklemini göstermektedir. Buna göre;

$$\Delta t = (d/c)\cos(\Phi) \qquad (2.3)$$

d: antenler arası uzaklık, (m)

c :Gelen dalganın hızı, (m/s)

d ve c'nın bilindiği varsayılıp zaman farkı, Δt de ölçüldüğü için yukarıdaki formülden, Φ açısı

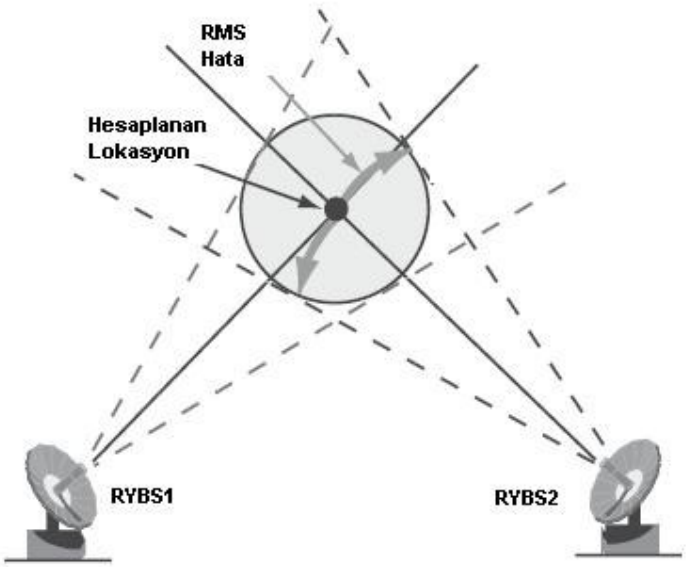
$$\Phi = \arccos(c\Delta t / d) \qquad (2.4)$$

ile hesaplanır.

2.3 Hedef Konumlandırılması

Hedef konumlandırılması literatürde radyo kaynak konumlandırma (RF source localization), pozisyon sabitleme (position fixing), pasif konumlandırma (passive

localization) veya pasif yön bulma (passive direction finding), yayıncı coğrafi-konum bulma (emitter geolocalization), hedef- kaynak konuşlandırma (target-source localization) gibi farklı adlar ile de anılmaktadır. Önceki bölümlerde anlatılan tekniklerle, sadece sinyalin geliş yönü, kerteriz açısı dolayısıyla, kerteriz hattı (KH) bulunabilmektedir; yayın yapan hedef de bu hat üzerinde veya uzağında-yakınında herhangi bir yerde olabilir. Yani anlatılan yöntemler ile hedefin konumu tam olarak belirlenememektedir. Radyo kaynağının konum bilgisini elde etmek için en az iki RYB istasyonuna veya hareketli bir RYB sistemine ihtiyaç vardır. Şekil 2.9'da betimlendiği gibi iki sistemden aynı anda kerteriz hatları çizildiğinde konum kestirimi doğruluk derecesi sınırlı da olsa yapılabilir. Bu yönteme Üçgenleme (Triangulation) adı verilmektedir. Doğruluk derecesini sınırlandıran etkenler RYB sistemlerinde pratikte her zaman varolan sistematik veya rasgele nitelikteki ölçüm hatalarıdır. Bu hataların tanımlamaları ve ayrıntıları daha sonraki bölümlerde anlatılacaktır. İki RYBS kullanarak oluşturulan konfigürasyona radyo yön bulma literatüründe kesme (CUT) adı da verilmektedir.



Şekil 2.9 İki RYBS Kullanarak Konum Kestirimi (Adamy, 2004)

Üç ya da daha fazla RYBS sistemi kullanarak konum belirleme tekniğine ise sabitleme (FIX) adı verilmektedir.

3. ÜÇGENLEME VE KONUM BELİRLEME ALGORİTMALARI

Bu bölümde [Herndon, 1991], [Poisel, 2002], [Stansfield, 1947], [Torrieri, 1984], [Foy, 1976], [Paradowski, 1995] ve [Poisel, 2005] kaynaklarında verilen bilgiler esas alınarak, üçgenleme, hedef konumlandırılması ve konum belirleme ile ilgili bilgiler verilmektedir.

Pozisyonları belirli RYB sistemlerinden hedefe doğru birden fazla kerteriz hattı oluşturup farklı RYBS'lerin kerteriz hatlarını kesiştirerek ve de bu kesişim noktalarını kullanarak, hedefin konumunu kestirme yöntemine üçgenleme (triangulation) adı verilmektedir. Üçgenleme ile ilgili ilk bilimsel etüt 1947 yılında Stansfield tarafından yayınlanmıştır (Stansfield, 1947). Üçgenleme COMINT (Communication Intelligence) ve ELINT (Electronic Intelligence) uygulamalarında sıkça kullanılagelmiştir.

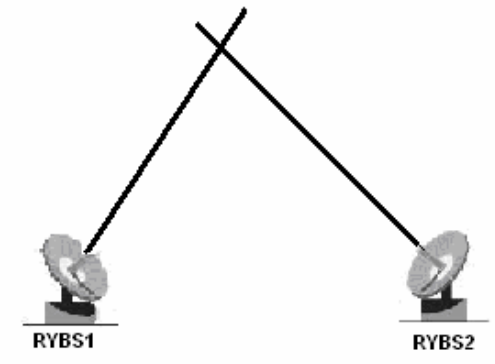
Yöntem hareket eden bir platform üzerinde tek bir RYBS ile veya iki ya da daha fazla RYBS'ye sahip sabit platformlar ile uygulanabilir. Teknik, yayının tipine karşı da oldukça esnektir, hem sürekli dalga ile hem de darbeli sinyaller ile çalışabilir.

Herhangi bir hatanın, gürültünün veya girişimin olmaması durumunda iki yada daha fazla istasyondan hedefe doğru yönelen kerteriz çizgileri tek bir noktada kesişir. Gürültü, RYBS'lerin rasgele ölçüm hataları, elektromanyetik dalganın farklı yerlerden gelen yansımalarının da antenlere düşmesi, antenlerin karşılıklı kuplajı, sistematik hatalar (bias) gibi etkenler kerteriz çizgilerinin farklı farklı yerlerde kesişmesine neden olurlar. Bu faktörler konum bilgisinin doğruluk derecesini sınırlayan öğelerin başlıcalarıdır. Üçgenleme yöntemi salt rasgele ölçüm hatalarını efektif olarak azaltabilirken; sistematik ölçüm hatalarına karşı çok duyarlıdır. Sistematik hata ancak, cihazın kalibre edilmesi ile giderilebilir.

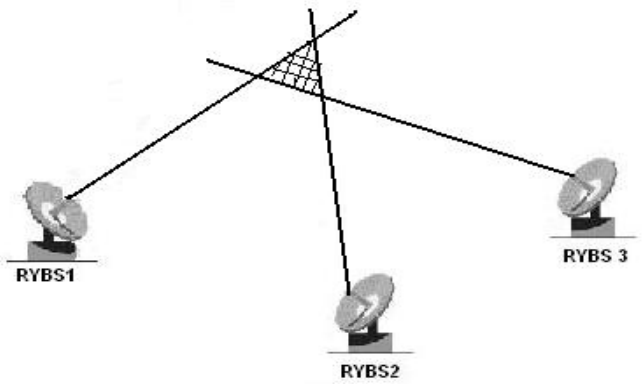
Bu bölümde ilk olarak üçgenleme yönteminin temel kavramları, daha sonrada başlıca konum belirleme algoritmaları kısaca anlatılıp bu çalışmada esas olarak kullanılan en küçük kareler kestirim yöntemi detaylı olarak verilecektir.

3.1 Üçgenleme Temel Kavramları

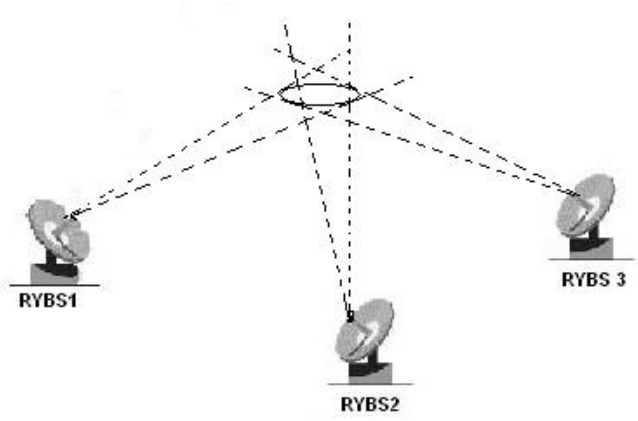
Temel üçgenleme formları aşağıda Şekil 3.1, Şekil 3.2, Şekil 3.3, farklı hata durumları için gösterilmiştir.



Şekil 3.1 Üçgenleme Geometrisi – Herhangi Bir Hatanın Olmadığı Durum için Temsili Gösterim



Şekil 3.2 Üçgenleme Geometrisi – Sistematik Hatanın Olduğu Durum için Temsili Gösterim



Şekil 3.3 Üçgenleme Geometrisi – Rasgele Hatanın Olduğu Durum için Temsili Gösterim

Yukarıda betimlenen geometriler yalnızca yatayda geliş açısı ölçümü için iki boyutlu kartezyen koordinat sistemi baz alınarak ve RYBS'lerin yerlerinin bilindiği varsayılarak oluşturulmuştur. İlk şekil, Şekil 3.1, ideal, yani herhangi bir hatanın olmadığı kerteriz hatlarının kesişim noktasının radyo kaynağının yerini tam olarak gösterdiği durum betimlenmiştir. İkinci şekilde ise ölçümde yalnızca sistematik hataların olduğu varsayılarak, tek bir kesişim noktasının değil bir hata bölgesinin tanımlandığı durum gösterilmiştir. Üçüncü durum ise, bu tez çalışmasının ana konusu olan yalnızca rasgele hataların olduğu varsayımının yapıldığı ve hata bölgesinin de eliptik olarak tanımlandığı durumdur.

Son senaryoda gösterilen yalnızca rastsal hataların olduğu durumda, radyo kaynağının konum bilgisinin doğruluk derecesi üç farklı doğruluk derecesine bağlıdır:

i-) RYB sisteminin konum bilgisinin doğruluk derecesi,

ii-) Açısal referansın (manyetik veya gerçek kuzey) doğruluk derecesi,

iii-) Ölçülen kerteriz açısının doğruluk derecesi,

Yukarıdaki doğruluk derecelerinden ilki GPS varolduğundan bu yana artık sorun olmaktan çıkmıştır. İkincisinde ise magnetometre veya ataletsel seyrüsefer sistemi ile doğruluk derecesi yüksek ölçümleme yapılabilmektedir. En önemli doğruluk derecesi

üçüncü maddede belirtilen kerteriz hattının ölçüm doğruluğudur. Bu ölçümün doğruluk derecesi konum bilgisinin doğruluk derecesini belirlemektedir.

Kerteriz Açısının Ölçümünde Karşılaşılan Hatalar Hatalar iki ana başlık altında toplanabilir: Sistematik hatalar veya ekipman hataları ile operasyon sırasında karşılaşılan ortam faktörlerine bağlı rasgele hatalar. Sistematik hata, herhangi bir fiziksel gerçekliği ölçen diğer herhangi bir sistem veya cihaz gibi kullanılan RYB cihazının kendisinde varolan, örneğin antenlerin referans kerteriz açısına yanlış hizalanması, veya gelen dalga ile ilgili varsayımın pratikte geçersiz olmasından dolayı ortaya çıkabilir. RYBS'ye gelen elektromanyetik dalga ile ilgili temel yaklaşım onun düzlemsel bir dalga olarak kabul edilmesidir. Bu varsayımın temelinde de kaynak ile RYB sisteminin arasında giden dalganın ovalliğini hissettirmeyecek oranda yeterli mesafe olması gerekliliği bulunmaktadır. Rasgele hatalar olarak tanımlanan ve RYB sisteminin ölçümlerini belirli bir sapmada yapmasına neden olan hatalar ise yukarıda da anlatıldığı gibi çoklu yol etkisi, ölçüm yapılan bölgenin coğrafi koşulları, günün hangi saatinde veya yılın hangi mevsiminde ölçüm yapıldığı gibi operasyon sırasında karşılaşılan temel olarak ortam faktörleri ile ilişkili olan hatalardır.

Sistematik hatalar sonucu oluşan hata üçgeninde radyo kaynağının yeri aşağıda Şekil 3.4'te gösterildiği gibi istatiksel olmayan, geometriye dayalı yöntemler ile kestirilebilir. Çünkü birden fazla ölçüm yapıp ortalama almak sistematik hataları ortadan kaldırmamaktadır.

Şekil 3.4 Hata Üçgeninde Geometri Bilgisi Kullanarak Konum Belirleme

Geometri kullanarak yapılan konum belirleme oldukça basit olup yetersizlikleri fazladır. Bu yetersizliklere, bu yöntemlerde kerteriz hatlarının veya açılarının kalitesi,

standart sapması, sayıları gibi bilgilerin kullanılmaksızın yalnızca oluşan üçgensel bölgede işlem yapılması örnek olarak verilebilir (Herndon, 1991).

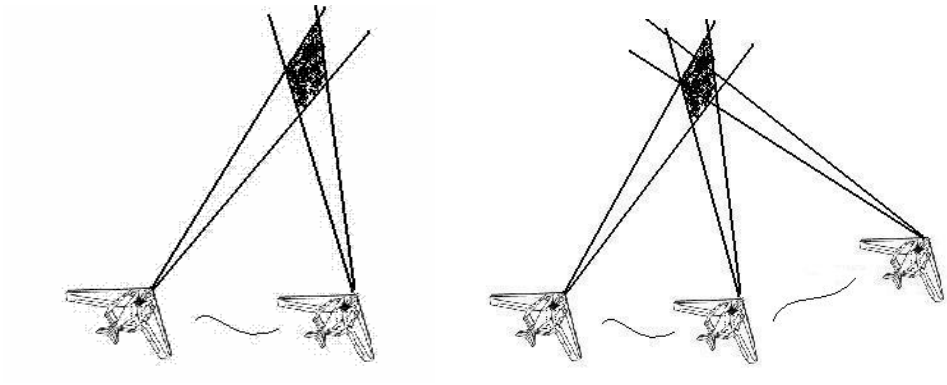
İstatiksel olmayan pozisyon kestirim yöntemleri içerdiği yetersizliklerden dolayı istatiksel kestirim tekniklerinin konum belirleme problemlerine uygulanmasına neden olmuştur. İstatiksel konum belirleme algoritmaları her bir RYB sistemi için birden fazla ölçüm alarak çoklu üçgenleme yapmayı esas almaktadır. Çoklu ölçüm ve üçgenleme senaryoda kullanılan her bir RYB sisteminin ölçüm standart sapmasına bağlı olarak belli bir alanda farklı farklı noktalarda hedef bulunmasına yol açar. Ortaya çıkan çoklu dağılım istatiksel yöntemler ile incelenerek pozisyon kestirimi yapılmaktadır. Bu konuda öncü olarak kabul edilen Stansfield'ın çalışmalarını takip eden araştırmacılar, bu çalışmaların türevleri üzerinde çalışmışlardır. Örneğin Torrieri en büyük olabilirlik kestirimi (maximum likelihood estimation) veya en küçük kareler yöntemini, Butterly Bayesian yaklaşımını; Foy, Taylor serilerini baz alarak en az kareler yöntemini geliştirmiş, Springarn genişletilmiş Kalman filtre tekniğini pozisyon belirlemede kullanmıştır (Herndon, 1991).

Üçgenleme yöntemlerinde kullanılan istatistiksel kestirim yöntemleri tipik olarak, birbirlerinden bağımsız, sıfır ortalamalı normal dağılıma sahip rasgele hataların oluştuğu, sistematik hataların ise oluşmadığı varsayımlarına dayanmaktadır.

Rastsal kerteriz hataları literatürde standart sapma veya RMS (root-mean-square) değerleri verilerek tanımlanmaktadır. RYB cihazlarının hassasiyet ölçüsü olarak da bu değerler pozisyon kestirimi yapabilmesi için kullanıcıya verilir. Hatalı açısal ölçümler, kerteriz çizgilerinin tek bir noktada değil rasgele farklı farklı koordinatlarda kesişmesine, dağılmasına neden olur. Bu dağılım veya pozisyonlardaki rasgelelik kerteriz açısı ölçümlerinde olduğu gibi matematiksel modelde ilave sıfır ortalamalı beyaz Gauss gürültü olarak ifade edilmektedir. İki boyutta ölçüm yapıldığı için, yani yalnızca kuzey açısı ölçüldüğü için, Gauss gürültü iki değişkenli (x,y) olur. Çok değişkenli Gauss hatanın olasılık yoğunluk fonksiyonunun bir düzlem üzerine projeksiyonu veya konturları eliptiktir, oluşan elipsin parametreleri de pozisyon kestiriminde metrik olarak kullanılmaktadır ve literatürde bu metriğe Olası Dairesel Hata, (Circular Error Probable, CEP) veya Olası Eliptik Hata (Eliptical Error Probable, EEP) adı verilmektedir. Kestirimin doğruluk derecesi bu metrikler ile belirlenmektedir.

Hata elipsi veya hata dairesinin büyüklüğü, yani metriğin değeriyle pozisyon kestiriminin doğruluk derecesi birbirleri ile ters orantılıdır.

Elipsin büyüklüğünü her bir RYB sisteminin ölçüm hatasının standart sapması belirlediği kadar, RYB sistemlerinin ve hedeflerin uzayda göreceli geometrileri de belirlemektedir. Aynı ölçüm varyanslarına sahip RYB sistemleri farklı geometrik dizilimler için farklı elips büyüklükleri oluşturur. Bu geometrik dizilimin yer bulma hatalarına etkisine Geometrik Konuşlandırmaya Bağlı Hassasiyet Kaybı (Geometric Dilution of Precision, GDOP) denmektedir. RYB sistemlerinin sayısının artırılması da aşağıdaki Şekil 3.5'te de görüldüğü gibi elipsin büyüklüğünü yani diğer bir anlamda hata metriğinin değerini küçültüp kestirimin doğruluk derecesinin artmasına neden olmaktadır.



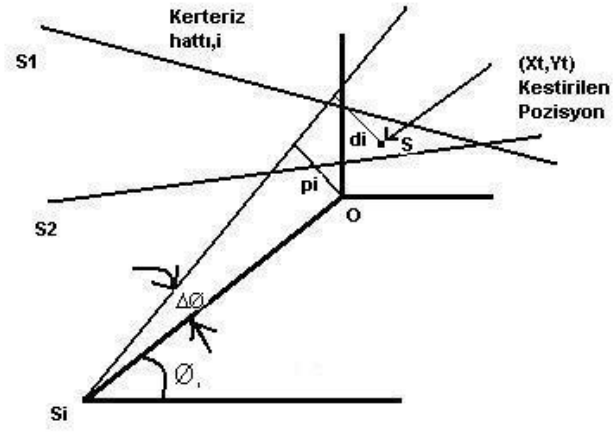
Şekil 3.5 RYB Sistemlerinin Sayısının Artırılması İle Belirsizlik Bölgesinin Küçülmesi

CEP ve GDOP metrikleri ile ilgili matematiksel ifadeler en küçük kareler kestirim yönteminin anlatıldığı bir sonraki bölümde verilecektir.

3.2 Konum Kestirim Algoritmaları

Stansfield'ın bu alanda yayınlanan ilk makalesinden günümüze değin farklı kestirim algoritmaları konum belirleme çalışmalarında kullanılagelmiştir. Bunların birçoğu en küçük kareler kestirim yöntemi ve türevlerine dayanmaktadır. Farklı yöntemler aynı sayıda ve aynı hassasiyetle çalışan RYBS'ler için farklı derecelerde sonuçlar sağlamaktadır. Fakat bazı yöntemler kestirime yanlılık (bias) ekleyebilmekte bazı yöntemlerin de karmaşıklık derecesi çok yüksek olabilmektedir. Bu bölümde ilk olarak Stansfield'ın klasik yön bulma sabitleme algoritması kısaca anlatılıp daha sonra da tezde kullanılan en küçük kareler kestirim yöntemi ayrıntılı olarak işlenmektedir.

3.2.1 RYB ile Konum Belirlemenin İstatiksel Teorisi – Stansfield'ın Kestirim Algoritması



Şekil 3.6 Stansfield'in Yöntemi İçin Kullanılan Örnek Bir Geometri

Stansfield'ın konum belirleme algoritması temel olarak en büyük olabilirlik kestirim (maximum likelihood estimation) yöntemine dayanmaktadır.

Stansfield da modelini geliştirmeye kerteriz açılarının ölçümlerindeki hataları sıfır ortalamalı Gauss gürültüsü $\mathcal{N}(0,\sigma^2 \mathrm{I})$ kabul ederek başlamıştır. Bu durumda hedef bölgesinde kesişen kerteriz hatlarının ortak olasılık yoğunluk fonksiyonları da iki

boyutta çok-değişkenli Gauss olasılık yoğunluk fonksiyonu olur. En büyük olabilirlik kestirim yöntemi ortak Gauss olasılık yoğunluk fonksiyonunun üstel kısmını maksimize eder. Üstel kısmın maksimize edilmesi aşağıdaki denklemlerde de daha açık ifade edildiği gibi üstel kısımdaki negatiflikten dolayı toplam hata olasılığını minimize edilmesi anlamına gelmektedir.

$$f(p_1, p_2,..., p_N) = \frac{1}{(2\pi)^{N/2}\prod_{i=1}^{N}\sigma_{p_i}} \exp\left\{\frac{-1}{2}\sum_{i=1}^{N}\frac{{p_i}^2}{{\sigma_{p_i}}^2}\right\} \tag{3.1}$$

Denklemdeki N kerteriz hattı sayısını göstermektedir.

Şekil 3.6 da verilen tanımlamalarda hedef $S=(x_T,y_T)$ noktasında varsayılmakta, d_i kerteriz hatlarının hedefin konumunu ıskalama uzaklığı (miss-distance) olarak gösterilmektedir.

$$d_i = p_i + x_T \sin(\Phi_i) + y_T \cos(\Phi_i) \tag{3.2}$$

$$\sin(\Delta\Phi_i) = \frac{p_i}{D_i} \tag{3.3}$$

D_i gerçek pozisyon ile i.inci istasyon arasındaki mesafeyi göstermektedir.

$\Delta\Phi_i$ küçük bir açı olduğu için şöyle bir yaklaşım yapılabilir:

$$p_i \approx \Phi_i D_i , \qquad \sigma_{pi} \approx D_i\sigma_\Phi \tag{3.4}$$

d_i 'ların ortak olasılık fonksiyonu yukarıdaki ifadeye benzer biçimde yazılabilir:

$$\begin{aligned} f(d_1, d_2,..., d_N) &= \frac{1}{(2\pi)^{N/2}\prod_{i=1}^{N}\sigma_{d_i}} \exp\left\{\frac{-1}{2}\sum_{i=1}^{N}\frac{{d_i}^2}{{\sigma_{d_i}}^2}\right\} \\ &= \frac{1}{(2\pi)^{N/2}\prod_{i=1}^{N}\sigma_{d_i}} \exp\left\{\frac{-1}{2}\sum_{i=1}^{N}\frac{(p_i + X_T\sin(\phi_i) - Y_T\cos(\phi_i))^2}{{\sigma_{d_i}}^2}\right\} \end{aligned} \tag{3.5}$$

x_T ve y_T değerleri üstel ifadeyi maksimum yapan en yüksek olasılığa sahip hedef koordinat noktalarıdır. Yukarıdaki ifadeden hedefin kestirim koordinat noktaları aşağıdaki gibi çıkar:

$$x_T = \frac{1}{ab - c^2} \sum_{i=1}^{N} p_i \frac{c\cos(\phi_i) - b\sin(\phi_i)}{\sigma_{di}^2}$$
$$y_T = \frac{1}{ab - c^2} \sum_{i=1}^{N} p_i \frac{a\cos(\phi_i) - c\sin(\phi_i)}{\sigma_{di}^2} \quad (3.6)$$

Eşitlikteki değişkenlerin tanımlamaları ise:

$$\tan(\phi_i) = \frac{y' - y_i}{x - x_i}$$
$$\cos(\phi_i) = \frac{d_i}{y' - y_T} \quad (3.7)$$

$$y' = (x_T - x_i).\tan(\Phi_i) + y_i \quad (3.8)$$

$$d_i = [(x_T - x_i).\tan(\Phi_i) + y_i - y_t] \cos(\Phi_i) \quad (3.9)$$

$$d_i = a_i.\ x_T + b_i.\ y_T - c_i \quad (3.10)$$

$$a = \sum_{i=1}^{N} \frac{\sin^2(\phi_i)}{\sigma_{di}^2 Di^2} \quad (3.11)$$

$$b = \sum_{i=1}^{N} \frac{\cos^2(\phi_i)}{\sigma_{di}^2 Di^2} \quad (3.12)$$

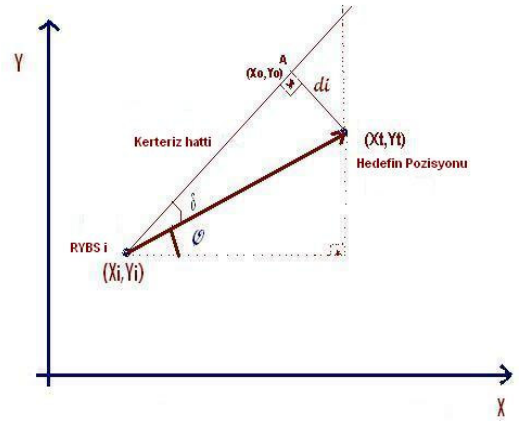
$$c = \sum_{i=1}^{N} \frac{\sin(\phi_i)\cos(\phi_i)}{\sigma_{di}^2 Di^2} \quad (3.13)$$

olarak verilir.

3.2.2 En Küçük Kareler Konum Belirleme Algoritması

Şekil 3.7'de genel olarak algoritmanın geliştirilmesinde kullanılan geometrik ilişkiler kartezyen koordinat sisteminde gösterilmektedir. Bu gösterime göre iki boyutlu uzayda RYBS'ler (x_i,y_i) noktalarında hedef ise (x_T,y_T) koordinatlarında yer almaktadır.

Betimlemede herhangi bir RYBS ile hedef arasındaki kerteriz açısı Ø, ölçüm hatasından kaynaklanan açı sapması ise δ olarak gösterilmektedir.



Şekil 3.7 En-Küçük Kareler Konum Kestirimi Algoritmasında Kullanılan Geometrik İlişkiler

Kestirim optimizasyon algoritmalarının amacı pozisyonun kestirilen değeri ile gerçek değeri arasındaki farkı ya da hatayı azaltan kestiriciyi bulmaya dayanmaktadır. Bu yöntem de ölçülen kerteriz hatlarının ıskalama uzaklıklarının (miss-distance) karesini minimize etmeye dayanmaktadır. Kerteriz açılarının ölçümlerindeki gürültü veya hatanın dağılımı Gauss ve sıfır ortalamalı, $\mathcal{N}(0,\sigma^2 I)$, olarak kabul edilirse doğrusal kestirim modeli için en büyük olabilirlik kestirimi ile en küçük kareler kestirimi eşlenik olur ve her ikisi de minimum varyanslı yanlı olmayan (unbiased) kestirici sınıfına girer (Torrieri, 1984).

Algoritmanın matematiksel modeli şu şekildedir:

$$\tan(\phi_i + \delta_i) = \frac{y - y_i}{x - x_i} = \frac{y_o - y_i}{x_o - x_i} \cong \tan(\phi_i) \qquad (3.14)$$

Eşitlik 3.14'te δ küçük bir açıyı gösterdiği için böyle bir yaklaşım yapılmaktadır.

$$d_i = a_i X_t + b_i Y_t - c_i \qquad (3.15)$$

d_i hedefin koordinatlarından A noktasına indirilen dikmenin denklemidir.

d_i=|A(x,y)-TargetPos| = $\frac{|aX_t + bY_t + c|}{\sqrt{a^2 + b^2}} = \frac{|\tan(\phi_t)X_t - 1.Y_t + (\tan(\phi_t)X_i - Y_i)|}{\sqrt{\tan^2(\phi_t) + (-1)^2}}$ (3.16)

Yukarıdaki eşitlik analitik geometri eşitliği olup bir noktanın bir doğruya olan uzaklığını, d_i'ın uzunluğunu göstermektedir. Eşitliğin paydasında verilen

$1 + \tan^2(\phi_t) = 1 + \frac{\sin^2(\phi_t)}{\cos^2(\phi_t)} = \frac{\cos^2(\phi_t) + \sin^2(\phi_t)}{\cos^2(\phi_t)} = \frac{1}{\cos^2(\phi_t)}$ olarak açılırsa

$$d_i = |\frac{\sin(\phi_t)}{\cos(\phi_t)}X_t - Y_t + \frac{\sin(\phi_t)}{\cos(\phi_t)}X_i - Y_i|.\cos(\phi_t) \quad (3.17)$$

$$d_i = \sin(\phi_t)X_t - \cos(\phi_t)Y_t + \sin(\phi_t)X_i - \cos(\phi_t)Y_i \quad (3.18)$$

$$a_i = \sin(\phi_t) \quad (3.19)$$

$$b_i = -\cos(\phi_t) \quad (3.20)$$

$c_i = x_i . \sin(\phi_t) - y_i . \cos(\phi_t)$ (3.21)

olduğu görülür.

Bu eşitlikleri matris formatında yazacak olursak;

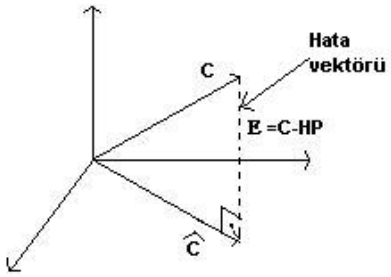
D=HP-C (3.22)

$$D = \begin{bmatrix} d_1 \\ d_2 \\ . \\ . \\ d_N \end{bmatrix} \quad H = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ . & . \\ . & . \\ .a_N & b_N \end{bmatrix} \quad P = \begin{bmatrix} X_T \\ Y_T \end{bmatrix} \quad C = \begin{bmatrix} c_1 \\ c_2 \\ . \\ . \\ c_N \end{bmatrix} \quad (3.23)$$

D matrisi sıfır matrisine eşitlenebilir. Bu durumda eşitlik C= HP olarak yazılabilir. Ölçüm hataları da sıfır ortalamalı Gauss dağılımına sahip oldukları varsayılarak, eşitliğe dahil edilirse;

C = HP + W (3.24)

Bilinmeyen P vektörünü en küçük kareler kestirim yöntemi ile bulabiliriz. Buradan bulacağımız değerler hedefin konum bilgisini bize verecektir.



Şekil 3.8 Dikgen İzdüşüm Prensibi

$$J(P) = (C-HP)^T(C-HP) = \| C-HP \|^2 \quad (3.25)$$

|| C-HP || fonksiyonu, minimize edilmesi gereken hata fonksiyonudur. Şekil 3.8'de de gösterildiği gibi hatalar verilere diktir. Bu prensibe dikgen prensibi (Orthogonality Principle) adı verilmektedir. Yukarıdaki ifade açık halde yazılırsa;

$$J(P) = C^TC - C^THP - P^TH^TC + P^TH^THP \quad (3.26)$$

$$= C^TC - 2C^THP + P^TH^THP \quad (3.27)$$

$\frac{\partial J(P)}{\partial P} = 0$ hata fonksiyonunun türevi hatayı minimum yapan çözümü verir.

$$= -2H^TC + 2H^THP = 0 \quad (3.28)$$

$H^TH = H^THP$ buradan pozisyon kestirimi (3.29)

$P = (H^TH)^{-1} H^TC$, olarak bulunur. (3.30)

Ağırlıklandırılmış/genelleştirilmiş en küçük kareler kestirim yöntemi ile doğruluk derecesi daha yüksek kestirimler elde edilebilir; ağırlıklandırma matrisi W, ölçüm hatalarının kovaryans matrisi olarak ifade edilecek olursa;

$$W = \left[\sigma^2_{\phi i}\{(x_i - x_t)\cos(\phi_i) + (y_i - y_t)\sin(\phi_i)\}^2\right] \quad (3.31)$$

Hedefin konum kestirim vektörü de

$$P = (H^T W^{-1} H)^{-1} H^T W^{-1} C \tag{3.32}$$

olarak bulunur. Bu kestirimin hata kovaryans matrisi ise aşağıdaki gibidir:

$$Q_o = (H^T W^{-1} H)^{-1} = \begin{bmatrix} \sigma^2{}_x & \rho_{xy}\sigma_x\sigma_y \\ \rho_{xy}\sigma_x\sigma_y & \sigma^2{}_y \end{bmatrix} \tag{3.33}$$

Eşitlik 3.33'te yer alan σ_x ve σ_y, x ve y eksenlerindeki pozisyon hatalarının standart sapmalarını, ρ_{xy} ise x ve y deki hataların arasındaki korelasyon katsayısını belirtmektedir.

Olası hedef pozisyonu kestirimi $P = [X_t \; Y_t]^T$ rasgele vektörü ile gösterilir ve de pozisyon kestirimindeki hataların Gauss dağılama sahip oldukları varsayılırsa P vektörünün olasılık yoğunluk fonksiyonu

$$f(P) = \frac{1}{2\pi \mid \det \sum \mid^{1/2}} \exp\left\{ \frac{-1}{2} \begin{pmatrix} x_t - \mu_x \\ y_t - \mu_y \end{pmatrix}^T \sum{}^{-1} \begin{pmatrix} x_t - \mu_x \\ y_t - \mu_y \end{pmatrix} \right\} \tag{3.34}$$

olarak yazılır ($Q_o = \sum$). Yukarıdaki olasılık yoğunluk fonksiyonunda ortalamaları sıfır varsayıp üstel ifadeyi açacak olursak :

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix}^T \begin{bmatrix} \sigma^2{}_x & \rho_{xy}\sigma_x\sigma_y \\ \rho_{xy}\sigma_x\sigma_y & \sigma^2{}_y \end{bmatrix}^{-1} \begin{pmatrix} x_t \\ y_t \end{pmatrix} =$$

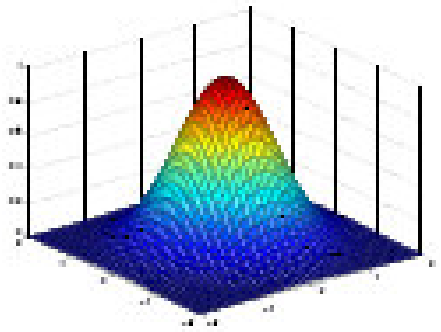
$$= \begin{bmatrix} \dfrac{x_t}{\sigma^2{}_x(1-\rho_{xy}{}^2)} - \dfrac{y_t\rho_{xy}}{\sigma_x\sigma_y(1-\rho_{xy}{}^2)} & \dfrac{-x_t\rho_{xy}}{\sigma_x\sigma_y(1-\rho_{xy}{}^2)} + \dfrac{y_t}{\sigma_y{}^2(1-\rho_{xy}{}^2)} \end{bmatrix} \begin{pmatrix} x_t \\ y_t \end{pmatrix}$$

$$= \begin{bmatrix} \dfrac{x_t{}^2}{\sigma^2{}_x(1-\rho_{xy}{}^2)} - \dfrac{x_t y_t\rho_{xy}}{\sigma_x\sigma_y(1-\rho_{xy}{}^2)} & -\dfrac{x_t y_t\rho_{xy}}{\sigma_x\sigma_y(1-\rho_{xy}{}^2)} + \dfrac{y_t{}^2}{\sigma_y{}^2(1-\rho_{xy}{}^2)} \end{bmatrix}$$

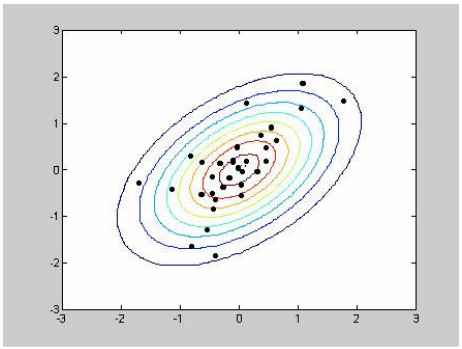
$$= \frac{1}{(1-\rho^2)} \left[\frac{x_t{}^2}{\sigma^2{}_x} - \frac{2x_t y_t\rho_{xy}}{\sigma_x\sigma_y} + \frac{y_t{}^2}{\sigma_y{}^2} \right] \tag{3.35}$$

Yukarıdaki eşitlik elips denklemidir, bu denklem aynı zamanda pozisyon kestirimi için güvenirlik aralığını da göstermektedir.

İki boyutta tipik bir Gauss olasılık yoğunluk fonksiyonu Şekil 3.9 da verilmektedir. İki boyutlu bu fonksiyonun kontur hatları veya izohipsleri elips biçimindedir. Daha öncede belirtildiği gibi ölçüm doğruluğunu gösteren bu metriğe olası eliptik hata (OEH) adı verilmektedir. Şekil 3.10 da elips olarak gösterilen hata bölgesi, korelasyon katsayısının sıfır ($\rho=0$) olması durumunda dairesel bölgeye dönüşür. Bu durumda olası dairesel hata (ODH/CEP) adlandırmasıyla hata bölgesi tanımlanır.



Şekil 3.9 İki değişkenli Gauss Olasılık Yoğunluk Fonksiyonu



Şekil 3.10 İki değişkenli Gauss Olasılık Yoğunluk Fonksiyonunun Konturları ve Ölçülen Pozisyonların Dağılımları

Eşitlik 3.33'te türetilen kovaryans matrisi, Q_o aşağıda türetilen denklemlerde gösterildiği gibi; olası eliptik hata bölgesini tanımlayan elipsin kısa ve uzun eksenleri, elipsin x-eksenine göre rotasyonunu yani elipsin büyüklüğünü dolayısıyla da kestirimin doğruluk derecesini vermektedir.

Hata kovaryans matrisi Q_o'nun özdeğerleri $\det|\lambda I - Q_o| = 0$ eşitliği ile aşağıda elde edilmiştir:

$$\lambda_1 = \frac{1}{2}\left[\sigma_x^2 + \sigma_y^2 + \sqrt{(\sigma_x^2 - \sigma_y^2)^2 + 4\rho_{xy}^2\sigma^2{}_x\sigma^2{}_y}\right] \quad (3.36)$$

$$\lambda_2 = \frac{1}{2}\left[\sigma_x^2 + \sigma_y^2 - \sqrt{(\sigma_x^2 - \sigma_y^2)^2 + 4\rho_{xy}^2\sigma^2{}_x\sigma_y^2}\right] \quad (3.37)$$

Bu değerler güvenirlik aralığını betimleyen hata elipsinin küçük ve büyük yarıçaplarını vermektedir:

$$a^2 = \frac{\sigma^2{}_x\sigma^2{}_y(1-\rho_{xy}^2)C^2}{\lambda_1} \quad (3.38)$$

$$b^2 = \frac{\sigma^2{}_x\sigma^2{}_y(1-\rho_{xy}^2)C^2}{\lambda_2} \quad (3.39)$$

Yukarıdaki eşitliklerde yer alan C katsayısı P noktasının güvenirlik aralığında, yani elipsin içinde olma olasılığını göstermektedir (Torrieri, 1984):

$$C = -2\ln(1-P_e) \quad (3.40)$$

Denklemde verilen P_e kestirilen pozisyon koordinatının elipsin içinde bulunma olasılığını göstermektedir. Tipik olarak P_e değeri, %50, %90 veya %95 seçilmektedir. Bu tez çalışmasında ise P_e değeri 0.95 olarak kullanılmıştır.

Elipsin yatay düzlemle yaptığı eğim açısı ise:

$$\tan 2\theta = \frac{2\rho_{xy}\sigma_x\sigma_y}{\sigma_y^2 - \sigma_x^2} \quad (3.41)$$

olarak tanımlanmaktadır.

Pratikte olası eliptik hata yerine elipsin daireye yaklaşımı yapılarak hata bölgesi elipse eşdeğer olacak biçimde iki boyutta dairesel olarak tanımlanmaktadır. Eğer yükselme açısı da ölçülüyor olsa idi üç boyutta olası küresel hata (spherical error probable, SEP) bölgesinden bahsetmek gerekecekti.

Elipsin daireye çevrimi Q_o matrisinin öz değerleri arasındaki orana bağlı olarak iki farklı yaklaşımla yapılmaktadır :

$$CEP \approx 0.563\sqrt{\lambda_1} + 0.614\sqrt{\lambda_2}\ , \text{ eğer } 0.1<\lambda_2/\lambda_1<0.3 \text{ ise} \qquad (3.42)$$

Bu yaklaşımın doğruluk derecesi özdeğerler arasındaki oran 0.3 veya daha fazla ise %1'e kadar yükselmekte, oran 0.1 ila 0.3 arasında olduğunda %10 civarında, diğer yerlerde ise %20 civarında gözlemlenmektedir (Torrieri, 1984).

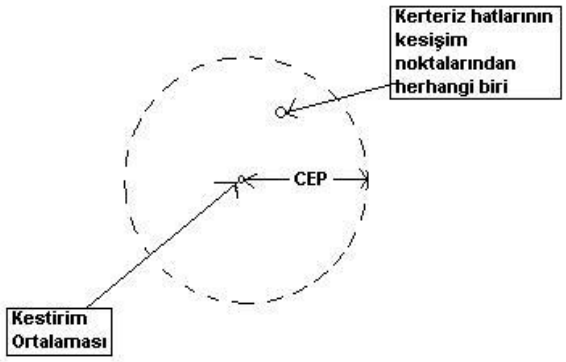
Ortalama %10 civarı bir doğruluk derecesine sahip yaklaşım ise daha basit olarak şu şekilde ifade edilmektedir:

$$CEP \approx 0.75\sqrt{Tr(Qo)} = 0.75\sqrt{\sigma_y^2 + \sigma_x^2} \qquad (3.43)$$

Yukarıdaki eşitliklerden de görüldüğü gibi eliptik hata gölgesi kovaryans matrisi tarafından karakterize edilmektedir.

Şekil 3.11'de bir CEP örneklemesi görülmektedir. Burada dairenin orta noktası yapılan kestirimlerin ortalamasını, dairenin büyüklüğünü anlatan yarıçap da CEP metriğinin büyüklüğünü göstermektedir. Şekildeki iki kerteriz hattının örnek bir kesişim noktası da şekli daha iyi betimlemek için verilmiştir.

CEP değeri, hataların %95'ini içine alan dairenin yarıçapıdır. Kısacası, hedefin gerçek yerinin, kestirim noktası çevresine çizilen CEP yarıçaplı bir dairenin dışında olma olasılığı %5'tir.



Şekil 3.11 CEP İlustrasyonu

Bu tezde temel alınan en küçük kareler kestiricisi yanlı (biased) bir kestirici değildir. Eğer yanlı (biased) olsa idi yukarıdaki şekil daha farklı olacak hedefin konum bilgisi dairesel hata bölgesinin dışına çıkacaktı. Aşağıdaki matematiksel ifadeler kestiricinin bu özelliğini açıklamaktadır. Kestirimin beklenen değeri hesaplanırsa:

$$E\{\overline{P}\}=E\{(H^TW^{-1}H)^{-1}H^TW^{-1}C\}, \quad C = HP+W \tag{3.44}$$

$$= E\{(H^TW^{-1}H)^{-1}H^TW^{-1}(HP+W)\} \text{ ifade açık olarak yazıldığında} \tag{3.45}$$

$$= E\{H^{-1}W(H^T)^{-1}H^TW^{-1}(HP+W)\} \tag{3.46}$$

$$= E\{H^{-1}W(H^T)^{-1}H^TW^{-1}HP\}+ E\{H^{-1}W(H^T)^{-1}H^TW^{-1}W\} \tag{3.47}$$

$$= E\{H^{-1}W\,IW^{-1}HP\} + E\{H^{-1}W\,IW^{-1}W\} \tag{3.48}$$

$$= E\{H^{-1}\,IHP\} + E\{H^{-1}W\} \tag{3.49}$$

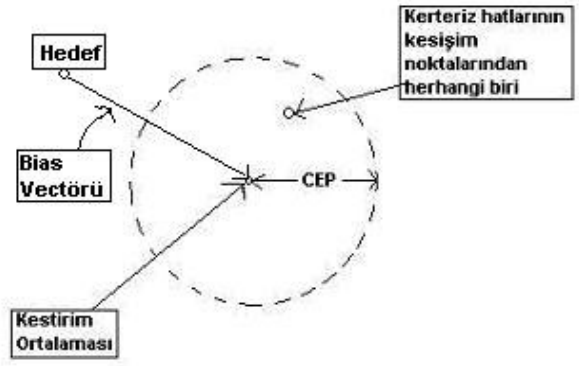
$$= E\{IP\} + H^{-1}E\{W\} \quad W = \mathcal{N}(0,\sigma^2) \tag{3.50}$$

$$= P + H^{-1}.0 \tag{3.51}$$

$$E\{\overline{P}\} = P \tag{3.52}$$

$$\text{Bias} = B = E\{\overline{P}\} - P = 0 \tag{3.53}$$

Denklemlerden görüldüğü gibi kestirici yansız (unbiased) bir kestiricidir. Eğer kestirici yanlı (biased) olsa idi yukarıda verilen şekil aşağıdaki gibi olabilecekti:



Şekil 3.12 Yanlı (Biased) CEP İlustrasyonu

Yanlılık, herhangi bir hedefin pozisyonunu gerçek pozisyondan Bias+CEP kadar uzaklığa taşımaktadır.

RYB istasyonları ile hedef arasındaki geometrik ilişkiyi tanımlayan ve hata ölçüt birimlerinden biri olan Geometrik Konuşlandırmaya Bağlı Hassasiyet Kaybı (Geometric Dilution of Precision, GDOP) ile CEP arasındaki ilişki ise aşağıdaki denklemde verilmektedir (Torrieri, 1984).

$$\text{GDOP} = \text{CEP}/0.75 \approx \sqrt{Tr(Qo)} \qquad (3.54)$$

Literatürde GDOP genellikle GPS sistemleri için kullanılan bir ölçü metriğidir; ESM veya RYB sistemleri için ise CEP metriği kullanılmaktadır. Bir sonraki başlık altında anlatılan ve bu tezin ana temasını oluşturan yerleşim planı (YP), (baseline) seçme bölümünde GDOP daha geniş kapsamda ele alınmaktadır.

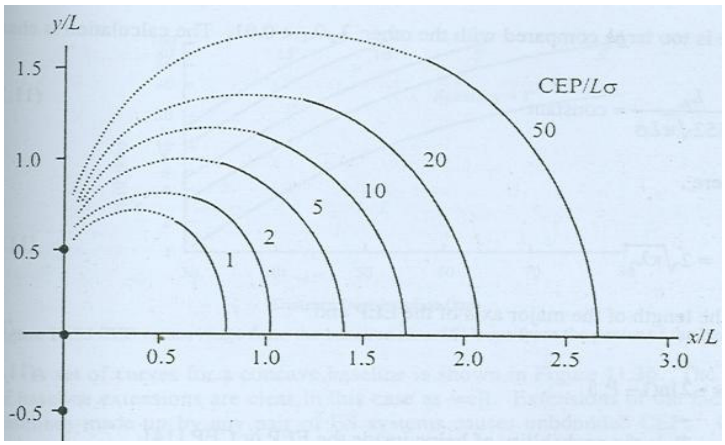
3.3 RYB Sistemlerinin Yerleşim Planını (YP) Seçme ve Sabit Kapsama

Konuşlandırma ya da geometrik yerleşim planı, havada, yerde veya denizde RYB sistemlerini konum belirleme hatalarını minimize edecek en optimal geometride yerleştirmektir. Birden fazla sistem kullanılarak gerçekleştirilen konuşlandırmada herhangi iki sistem arasını gösteren hatta “baseline-yerleşim planı” adı verilmektedir.

Yer veya hava platformlarında ESM gibi RYB fonksiyonu içeren sistemlerin, dinleme yapacağı hedef alanını optimum kapsayacak biçimde konuşlandırılması, farklı faktörlere bağlıdır. Yer sistemlerinin konuşlandırılması özellikle konuşlandırma yapılacak arazinin coğrafi koşullarına bağlı olarak değişmektedir. Örneğin konuşlandırma sahasında olası en yüksek noktalara sistemlerin yerleştirilmesi kapsama alanını artırıcı faktörlerinden birisidir. Hava platformları için ise hava koşulları, konuşlandırma için önemli parametrelerden biri olmaktadır.

Çevresel sınırlayıcı faktörler göz önüne alınarak herhangi bir hedef alanının olası en uygun biçimde kapsanması için sistemlerin bir arada belirli bir geometride

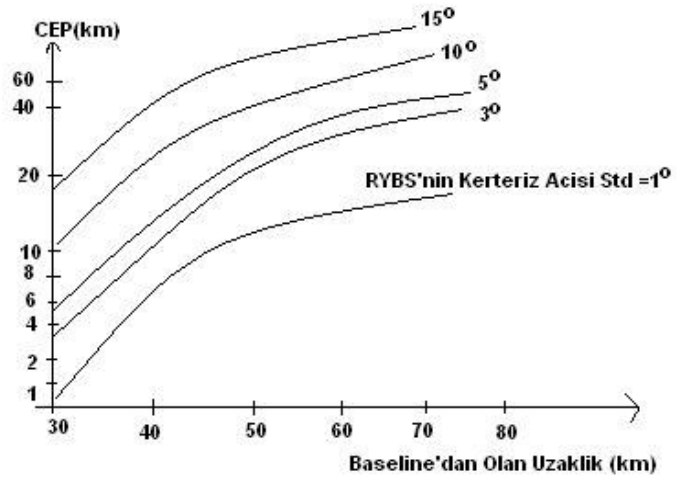
konuşlandırılması gerekmektedir. Daha öncede belirtildiği gibi konuşlandırmanın konfigürasyonu, üçgenleme yöntemlerinde pozisyon kestiriminin doğruluk derecesine (CEP) doğrudan etki etmektedir ve bu etkiye Geometrik Konuşlandırmaya Bağlı Hassasiyet Kaybı (Geometric Dilution of Precision, GDOP) adı verilmektedir. Şekil 3.13'de örnek bir konuşlandırma konfigürasyonu görülmektedir. ESM/RYB sistemleri koordinat sisteminde L= -0.5, 0, +0.5 noktalarında bulunmaktadır. Şekil üzerinden takip ettiğimizde, örneğin x/L ≈ 1 ve y/L ≈ 0.5 alındığında CEP/Lσ oranı 2 çıkmakta. YP uzunluğunu 20km, kerteriz açısı standart sapmasını da 5^o kabul edersek; L=20, σ =5^o=0.09 rad; bu durumda CEP =2x20x0.09 =3,600 m çıkar. Hedefin konumunda yerleşim planının (baseline'ın) orta dikme çizgisinden sağa veya sola kaymalar olduğunda kestirim doğruluk derecesi düşmekte yani CEP metriğinin değeri büyümektedir. Bu etki GDOP etkisidir (Herndon, 1991).



Şekil 3.13 Doğrusal Yerleşim Planı (Baseline) için Sabit CEP/Lσ Konturları [Poisel A. Richard, 2002],

Yerleşim planının uçlarına (kuzey-güney noktaları) hedefi taşırsak, kerteriz hatları birbirleri ile doğrusal bağımlı hale gelip kesişemezler, bu durumda çözüm bulunamaz çünkü CEP veya EEP ölçülemez bir büyüklüğe çıkar.

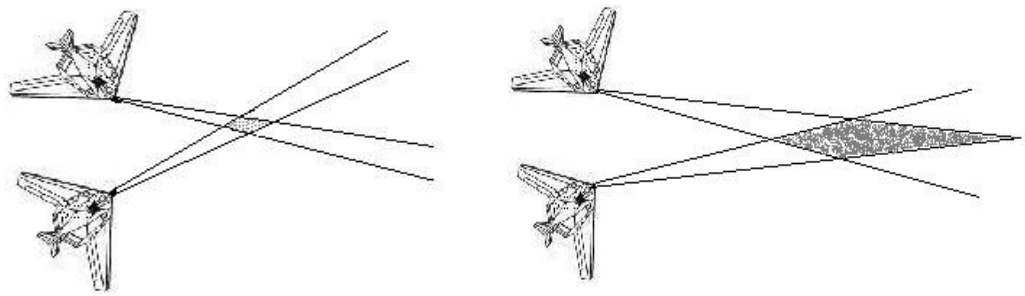
Şekilde nokta-nokta ile belirtilen bölümler CEP'in doğruluk derecesini göstermede uygun bir ölçüt olmadığı değerleri ifade etmektedir. Bu bölümlerde kovaryans matrisinin özdeğerlerinden biri (λ_1, λ_2) diğerine göre çok büyük çıkmaktadır, referans olarak $\lambda_1/\lambda_2 < 0.01$ alınabilir.



Şekil 3.14 Yerleşim Planının Orta Dikmesinden Hedefin Uzaklığı ile CEP Değeri Karşılaştırması

Şekil 3.14'de hedefin yerleşim planına olan farklı uzaklıklar için CEP metriğinin değişimi gösterilmektedir; burada hedefin yerleşim planının orta dikmesi üzerinde ve yerleşim planının da doğrusal olduğu varsayılmaktadır. Kerteriz açısı hassasiyeti küçüldükçe CEP değerindeki artış da şekilde gösterilmiştir.

Konuşlandırmada, hedefin yerleşim planına olan uzaklığının etkisi aşağıdaki şekilde daha açık olarak gösterilmektedir.



Şekil 3.15 Hedefin Uzaklığı ile CEP Değeri İlintisi

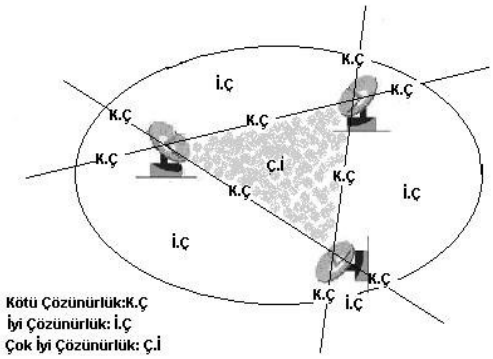
Daha önce açıklandığı üzere en iyi CEP değerleri, hedef YP'nin orta dikmesi üzerinde olduğunda elde edilmektedir. Yukarıda Şekil 3.14 de verildiği gibi hedef orta dikme üzerinde iken YP'ye olan uzaklığı da CEP değerinde etkilidir, en iyi CEP değeri hedef YP'nin orta dikmesi üzerinde ve kerteriz hatları YP ile 45 ^{o}'lik açı yaptığında ve birbirleri ile olası hedefte 90^{o} lik açı ile kesiştiğinde elde edilir.

Kesişim açısı 90^{o}'nin altına indikçe ölçümlerin performansı düşer daha büyük CEP'ler ortaya çıkar bu aynı zamanda YP'nin uzunluğunun istenilen değerde olmadığının bir göstergesidir.

Kesişim açısının 90^{o}'nin üzerinde olması da hedefin YP'ye çok yakın olması anlamına gelmektedir, bu da CEP değerini büyütür. Buradaki limit de hedefin YP üzerinde olmasıdır, bu durumda kerteriz hatları yine birbirleri ile doğrusal bağımlı olur ve CEP değeri sonsuza gider, kestirimin doğruluk derecesi de sıfır olur.

CEP ölçümünün güvenilir bir veri olabilmesi için kerteriz hatlarının hedef bölgesindeki kesişim açılarının eşik değeri Stansfield'ın makalesinde 30^{o} olarak verilmektedir. Bu değer ile en iyi pozisyon kestiriminde elde edilen CEP değerinin beş katı civarında daha kötü bir pozisyon doğruluk derecesi elde edilmektedir.

Konuşlandırmada ikiden fazla RYB istasyonu kullanılıyorsa en iyi CEP değeri veya çözünürlüğü en yüksek kestirim değeri istasyonların hedefi çepeçevre sarmasıyla elde edilmektedir. Örneğin Şekil 3.16'da üç RYB sistemi için optimum konuşlandırma yön bulma sensörlerinin YP'lerinin hedef bölgesini sarmalayacak şekilde eşkenar üçgen oluşturmasıyla elde edilir.



Şekil 3.16 Üç RYBS'li Konuşlandırma Probleminde Oluşan Bölgelerin Beklenen Kapsama Kalite Nitelikleri

Bu tez çalışmasında bu bölümde anlatılan teori ve kabul gören yaklaşımların ışığında verilen bir hedef bölgesini ortalamada en iyi kalitede kapsayacak RYB istasyonlarının optimum konuşlandırma problemi yerleştirilebilecekleri bölgenin sınırları dahilinde incelenmektedir.

4. GENETİK ALGORİTMALAR ve RYB SİSTEMLERİNİ KONUŞLANDIRMA PROGRAMI

Bu bölümde ilk olarak tez çalışmasında kullanılan buluşsal (heuristic) bir optimizasyon yöntemi olan genetik algoritmalar ile ilgili genel bilgiler ardından da genetik algoritmaları optimum konuşlandırma probleminde yöntem olarak kullanan programı ve programda kullanılan temel algoritmaları verilmektedir.

4.1 Genetik Algoritmalar

Genetik algoritmalar (GA) doğal evrimin prensiplerine benzetim yaparak geliştirilmiş buluşsal arama ve optimizasyon algoritmalarıdır ve GA güdümlü rasgele arama (guided random search) teknikleri sınıfına girmektedir. GA temel olarak Charles R. Darwin'in Türlerin Kökeni (1859) adlı kitabında anlatılan "en uygun olanlar yaşar" prensibine dayanmaktadır. GA, optimizasyon problemlerinde evrim teorisi ile ortaya atılmış olan doğal ayıklama (natural selection), eşleştirme, mutasyon gibi mekanizmaları veya operasyonları kullanır.

İlk olarak 1970'lerde John Holland tarafından doğadaki genetik sürecin benzetimini bilgisayar ortamında gerçekleştirme düşüncesi ile ortaya çıktı. Daha sonra öğrencisi Goldberg tarafından geliştirilmiş ve günümüze değin birçok mühendislik probleminde örneğin NP-complete problemlerin çözümünde, sürekli olmayan veya ayrık, kesikli fonksiyonların optimizasyonunda ve kombinetaryol problemlerde uygulana gelmiştir (Haupt, 2004).

Biyolojiden esinlenerek geliştirilen genetik algoritmalarda kullanılan terminoloji de biyoloji terimlerinden kaynaklanmaktadır. Şekil 4.1'de de GA terminolojisinde kullanılan terimlerden bazıları da görsel olarak ifade edilmektedir.

Genetik Algoritmaları anlamak için gerekli kavramlar:

Birey: Çözüm adaylarından birisi.

Kromozom : Çözüm adayının gösterim biçimi, genellikle değişkenlerin oluşturduğu bir vektör formu.

Gen: Kromozomdaki değişkenlerden birisi.

Lokus : Bir genin kromozomdaki yeri.

Alele: Bir genin değeri.

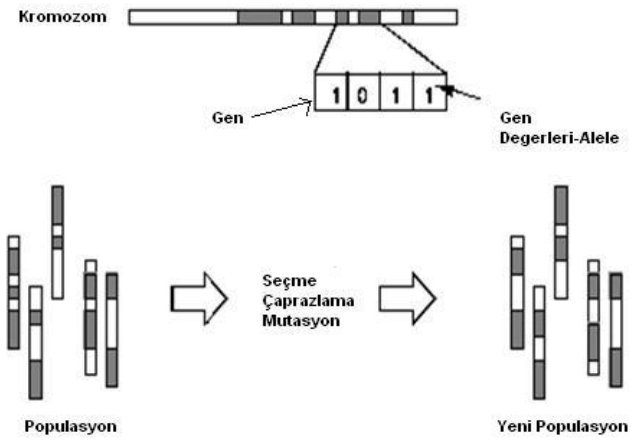
Çaprazlama: Anne ve babanın kromozomlarını (varolan çözüm adaylarını) belirli bir formda birleştirme işlemi, bu işlem sonucunda anne ve babadan gelen kromozomların karması olan yeni nesiller veya yeni çözüm adayları ortaya çıkar.

Mutasyon: Bir kromozomda oluşan rasgele değişiklik, bir veya bir kaç genin değerinin değişmesi işlemidir. Bu işlem popülasyonun farklılığını artırır, böylelikle erken yakınsama veya lokal optimuma takılmayı engeller.

Uygunluk Fonksiyonu: Uygunluk, performans, maliyet veya amaç fonksiyonu olarak da tanımlanabilen fonksiyon, biyolojideki bireyin yaşama olasılığını göstermesine benzer şekilde genetik algoritmalarda da her çözüm adayının sahip olduğu uygunluk değeri çözümün kalitesini göstermektedir. Genetik algoritmalarda probleme özel olan tek yer uygunluk fonksiyonudur. Algoritmanın diğer bölümleri standarttır.

Doğal Ayıklama: En uygun veya çözüme en yakın adayın seçilmesi,

olarak sıralanabilir.



Şekil 4.1 Bazı GA Terimlerinin Bir Gösterimi

Genetik algoritmalar, klasik analitik optimizasyon algoritmalarından dört temel noktada ayrılır:

- Genel olarak GA parametrelerin doğrudan kendilerini değil parametre takımının kodlanmış haliyle çalışır,
- GA aramaya tek bir noktadan değil, rasgele dağıtılmış bir noktalar kümesinden başlar. Dolayısıyla yerel bir optimuma takılmaksızın işlem yapabilirler.
- GA türev bilgisine ihtiyaç duymaksızın doğrudan uygunluk fonksiyonunu kullanır.
- GA'da deterministik değil rastlantısal geçiş kuralları kullanılır. Bu yüzden GA'lar stokastik arama yöntemleri olarak da yorumlanabilir.

Bunların dışında GA'ların diğer avantajları şu şekilde sıralanabilir:

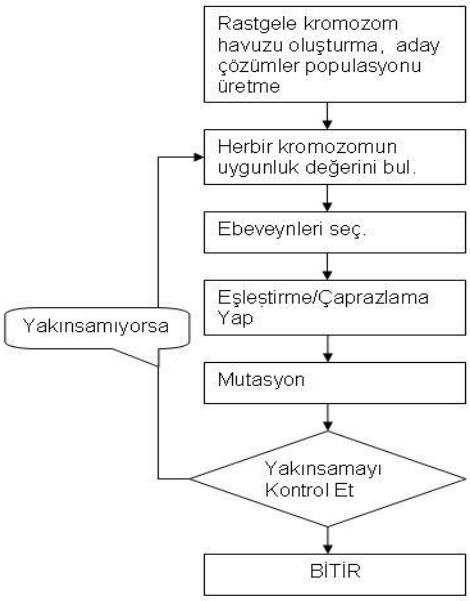
- Türev almak gerekli olmadığından dolayı sürekli veya kesikli değişkenlere kolayca uygulanabilir.
- Çok değişken kullanılması bir problem oluşturmaz.
- Paralel hesaplamaya uyumludur.
- Kompleks fonksiyonların optimizasyonunda kullanılabilir.

Optimizasyon probleminin çözümünde ne tip bir algoritmanın uygulanacağı problemin doğasına göre değişmektedir. Doğal olarak genetik algoritmaların her problem tipi için en iyi çözüm yolu olduğu iddia edilemez. Örneğin birkaç değişkeni olan konveks analitik bir fonksiyonun optimum noktalarını bulma problemini türev tabanlı klasik optimizasyon yöntemleri genetik algoritmalara göre çok daha efektif olarak çözer. Çalışmamızda genetik algoritmaların kullanım nedeni de analitik olarak çözümün zor olmasıdır.

Genetik algoritmaların çalışmasını ve başarısını matematiksel olarak açıklamaya yönelik bir çok çalışma bulunmaktadır. Bu çalışmalardan en çok bilineni ve diğer çalışmalara da kaynaklık edeni John Holland tarafından geliştirilen şema (schemata) kuramıdır.

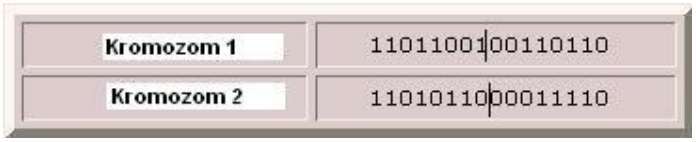
4.2 Genetik Algoritmaların Çalışma Prensipleri

Genetik algoritma optimum nokta aramaya bir noktadan değil arama uzayında bir çok noktadan başlar. Şekil 4.2'de de gösterilen genetik algoritmaların çalışma adımlarını kısaca şu şekilde özetleyebiliriz:

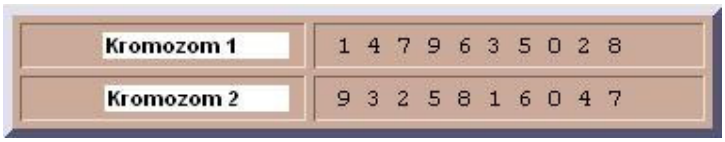


Şekil 4.2 GA Akış Şeması

İlk adımda, kromozomların yapısı tanımlanır ve kromozomu oluşturan parametreler, veya genler problemin doğasına uygun olarak kodlanır. Kodlama, ikili (binary) kodlama veya reel sayılarla yapılabilir. Aşağıdaki şekillerde farklı kodlama biçimleri gösterilmektedir:



Şekil 4.3 İkili Kodlama



Şekil 4.4 Permütasyon Kodlama

Kodlamanın belirlenmesinin ardından kromozomlar belirlenen sayıda rasgele olarak üretilir. Her bir kromozom veya çözüm adayı bireylerin oluşturduğu bu sete ilk popülasyon adı verilir. Popülasyonun büyüklüğü problemin yapısına göre değişmektedir. Çok küçük olursa algoritma lokal optimuma takılabilir çok büyük olursa da çözüme ulaşma zamanı çok uzayabilir. Genel olarak 20-30 aralığında bir büyüklük önerilmektedir. Fakat bu sayılar problemin yapısına, yani çözüm uzayının büyüklüğüne göre artabilir veya azalabilir.

Daha sonraki adımda, rasgele oluşturulan her bir kromozomun uygunluk değeri, tanımlanan uygunluk fonksiyonunun hesaplanması ile bulunur.

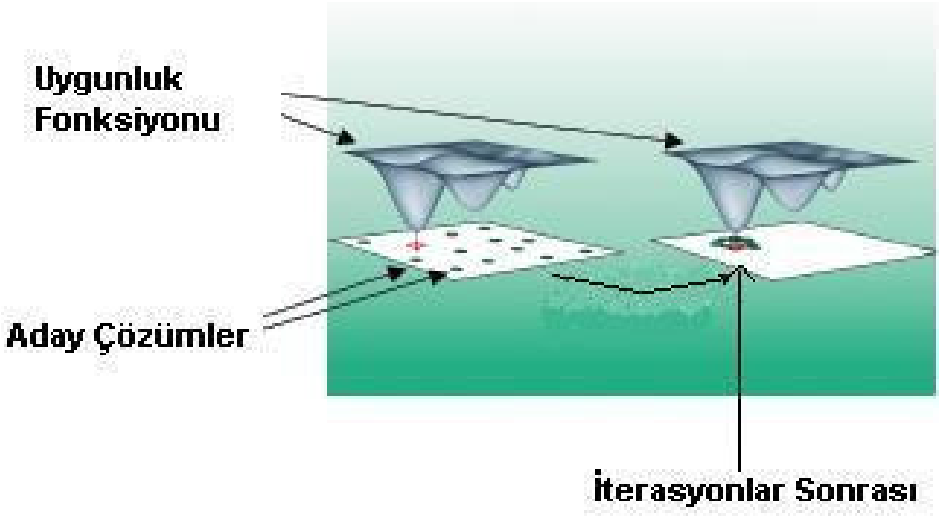
Üçüncü adım eşleştirme ve üreme adımıdır. Bu adımda bir önceki adımda bulunan uygunluk değerlerine göre, kromozomlar çiftler halinde seçilerek belirlenen çaprazlama operatörü ile yeni bireyler üretilir. Kromozomların seçim kriteri de önemlidir. Bu kriter çözüm adayının uygunluk değeri baz alınarak belirli bir algoritma ile yapılmaktadır. Çaprazlama yöntemi en başta probleme uygunluk olmak üzere,

parametrelerin kodlama biçimi gibi bazı faktörlere bağlı olarak değişmektedir. Buradaki amaç bir önceki kuşaktan yaşama şansı daha yüksek olan bireyleri kullanarak daha iyi nitelikleri olan bireyler oluşturmaktır. Oluşan bu yeni bireyler daha sonra mutasyona tabi tutulur. Mutasyondaki amaç gen çeşitliliğini sağlamak veya korumaktır ki böylelikle lokal optimumlara takılıp kalmadan tüm çözüm uzayında tarama yapılması sağlanmış olur. Bu bölümde anlatılan partnerlerin seçim stratejisi, çaprazlama, mutasyon operasyonları bir sonraki bölümde detaylı olarak verilmektedir.

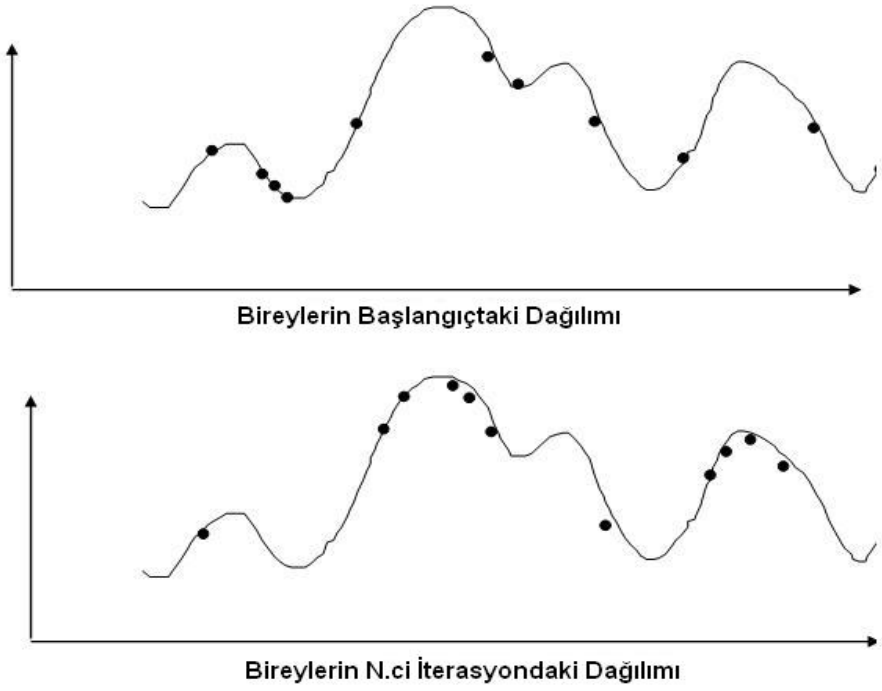
Yukarıdaki GA akış şemasında da gösterildiği gibi son adım yakınsama kontrol adımıdır. Çözüm arayışının sonlanıp sonlanmayacağının kriteri olan yakınsama kontrolünden devam etme durumu ortaya çıkıyorsa algoritma iteratif olarak çalışmasına ikinci adıma yani yeni oluşturulan bireylerin uygunluk değerlerinin hesaplanması adımına dönerek devam eder. Yakınsama koşulları çeşitli şekillerde tanımlanabilir. Bunlardan bazılarını şu şekilde sıralayabiliriz:

- Bireylerin birbirleri ile olan benzerlik oranlarının belli bir eşik değerinin üzerinde olması,
- Zaman kriterine erişilmiş olması,
- İterasyon sayısına erişilmesi veya
- Bir önceki ve bir sonraki iterasyonda bireylerin uygunluk değerleri arasındaki fark belirli bir eşik değerinin altında kalması,

yeterli yakınsamanın oluştuğunu gösterebilir. Çözüme doğru yeterli yakınsama oluştuğunda bireylerin tek bir noktaya doğru toplandığı aşağıdaki Şekil 4.5 ile Şekil 4.6'da temsilen gösterilmektedir.



Şekil 4.5 GA'nın Çözüme Yakınsama Gösterimi-I

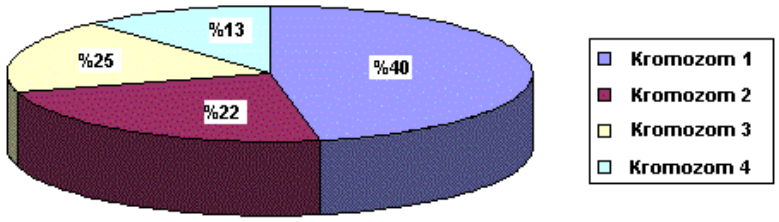


Şekil 4.6 GA'nın Çözüme Yakınsama Gösterimi-II

4.2.1 Genetik Algoritmalarda Seçim Yöntemleri

Genetik algoritmalarda seçim, bir sonraki kuşakta yer alacak bireylerin (çözüm adaylarının) üretilmesi için gerekli ebeveynlerin belirlenmesi işlemidir. Bu doğal seçimdeki en uygun adayın yaşaması durumuna benzerdir. Yöntemin amacı ortalama uygunluğun üzerindeki bireylerin çoğalmasına fırsat vermektir. Dolayısıyla yeni nesiller için oluşturulacak gen havuzuna mevcut bireyin girme olasılığı onun uygunluk değeri ile doğru orantılıdır. Günümüze değin farklı farklı seçim algoritmaları geliştirilegelmiştir. Bunlarda bazıları;

Rulet Tekerleği Yöntemi Bu yöntem stokastik geri koymalı örnekleme (stochastic sampling with replacement) yöntemi olarak da bilinmektedir. Yöntemin temelinde yatan ana fikir her bir çözüm adayına sahip olduğu uygunluk değeri oranında yaşama şansı tanınmasıdır. Her bir birey yaşama şansı oranında, oluşturulan hayali rulet tekerleğinde bir sektör veya bölge gösterir. Doğal olarak uygunluk değeri yüksek olan bireyin bu tekerlekte kapladığı alan Şekil 4.7'de da gösterildiği gibi daha büyük olacaktır.



Şekil 4.7 Rulet Tekerleği Gösterimi

Yöntemin algoritması şöyledir:

1) Her bir bireyin uygunluk değeri $f_1, f_2, \ldots, f_N$ olarak verilsin. Her biri için olasılık değerlerini hesapla.

$$P_i = f_i / (f_1 + f_2 + \ldots + f_N) \qquad (4.1)$$

Daha sonra eşleşme havuzuna girecek ebeveynlerin seçimi şu adımlar ile gerçekleştirilir:

2) Rasgele bir sayı üret, bu sayı rulet tekerine atılan topu göstermektedir.

$$0 \leq r < 1 \qquad (4.2)$$

Hangi bireyin seçileceğini aşağıdaki döngüye göre oluştur.

```
for i=1 den Birey Sayısı'na Kadar Yap
toplam =0;
başla:
        toplam:=toplam+P i;
        eğer   toplam ≥ r ise
                i değerini döndür;
bitir
```

Yukarıdaki döngü ile yeni bireylerin üretiminde kullanılacak mevcut bireylerin yaşama şansı yüksek olanlarından anne ve baba bireyler belirlenmektedir (Coley, 1999).

Turnuva Seçimi Yöntemi Bu yöntem de rulet tekerleği yöntemi gibi klasik bir yöntemdir. Uygunluk değerlerine göre kromozomlar sıralanmaksızın rasgele, turnuva sayısı kadar, k, mevcut popülasyondan kromozom seçilir. Bu alt kümeden ebeveynler rulet tekerleği yöntemi kullanarak veya en iyi olan kromozomlar arasından ya da başka olasılıksal yöntemle seçilebilir. Özellikle popülasyonun büyük seçildiği problemlerde uygunluk değerlerine göre bireyleri sıralama uzun süreceği için turnuva seçimi yöntemi başarılıdır (Haupt, 2004).

Elitist Seçim Yöntemi En az bir tane olmak üzere en iyi kromozomlar belirli bir oranda (seçilme oranında) korunarak gelecek nesillere aktarılır. Yöntem de en iyi çözümler korunmasına rağmen hızlı ve erken yakınsamadan dolayı lokal minimuma/maksimuma takılma olasılığı vardır (Haupt, 2004).

Rank Tabanlı Seçim Yöntemi Bu yöntemde bireyler direkt uygunluk değerlerine göre değil rank değerlerine göre seçilme olasılıkları alır. En az uygunluk değerine sahip çözüm adayından başlayarak doğrusal bir fonksiyona göre her birine bir rank

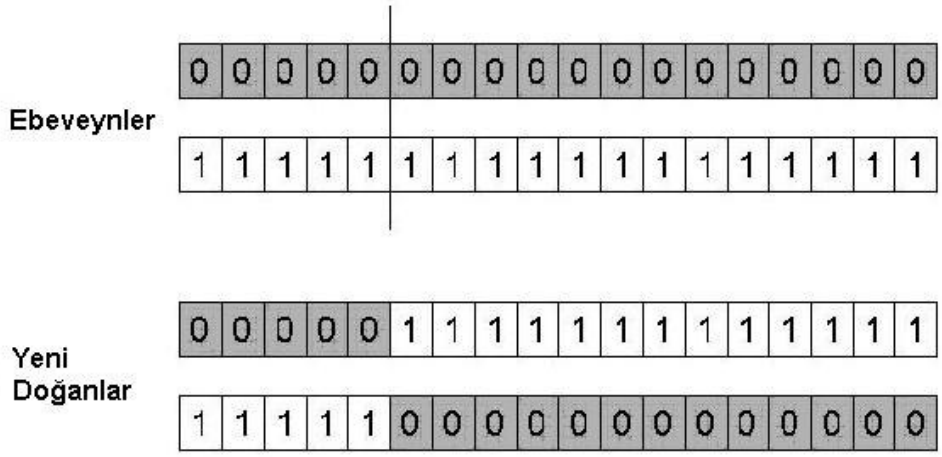
atanır. Bu rank değerleri toplanarak her bir aday için seçilme olasılığı hesaplanır. Genellikle rank değeri verildikten sonra rulet tekerleği yöntemi uygulanır.

Bu yöntemler dışında da literatürde çeşitli yöntemler yer almaktadır. Birden fazla yöntemi aynı anda kullanan hibrid çözümler de araştırmacıların başvurdukları teknikler arasındadır. Genellikle, seçilen yöntemin uygulanan problemde erken yakınsamaya yani genetik farklılığın kaybolmasına yol açıp açmadığına bakılarak aynı probleme farklı farklı yöntemlerle çözüm aranır.

4.2.2 Genetik Algoritmalarda Çaprazlama Yöntemleri

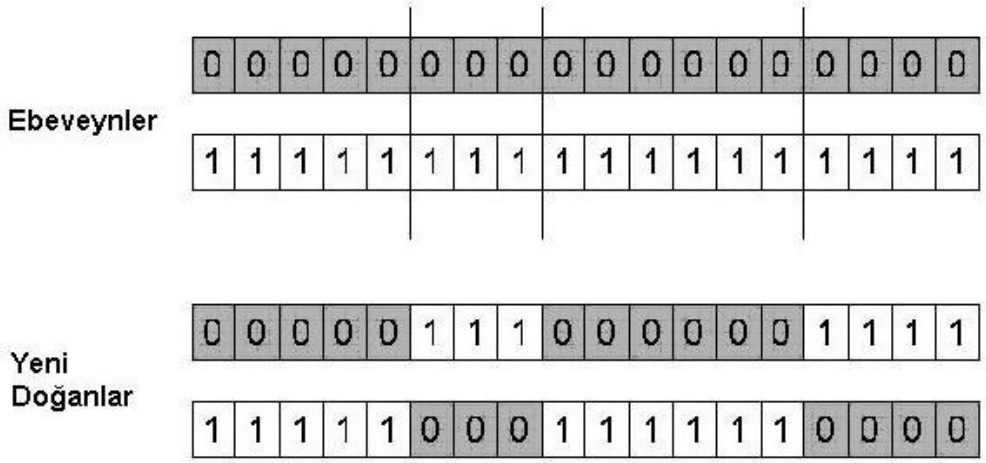
Çaprazlama operatörü genetik algoritmadaki en önemli operatördür ve iki çözüm adayını kullanarak daha çok yaşama şansına sahip yeni çözüm adayları ortaya çıkarma işlemidir. Yukarıda anlatılan ebeveyn seçimi ardından ebeveyn olarak seçilen kromozomlara uygulanan bu işlem ile arama uzayında farklı noktaların çözüm adayı olarak değerlendirilmesi ve büyük adımlar ile çözüm uzayının taranması gerçekleştirilir. Genetik algoritmalarda çaprazlama oranı tipik olarak 0.6-0.95 aralığında seçilmektedir. Algoritma çalışma zamanında işlerken rasgele bir sayı üretilmekte ve bu sayının önceden belirtilen çaprazlama oranından büyük olması durumunda ebeveyn havuzundan rasgele seçilen iki çözüm adayı üzerinde çaprazlama yapılmaktadır. Çaprazlama yöntemleri ise genellikle problemin doğasına uygun olarak kromozomların kodlanma yöntemlerine göre algoritmada uygulanmaktadır. Bunlardan bazıları:

Tek noktalı çaprazlama Bu yöntemde kromozomun uzunluğu dikkate alınarak üretilen rasgele bir sayı çaprazlama yapılacak noktayı göstermektedir. Bu noktanın sağ ve sol tarafı karşılıklı olarak kromozomlar arasında değiş-tokuş edilir. Görsel olarak da aşağıdaki şekilde yöntem ikili kodlanmış kromozomlar ile gösterilmektedir.



Şekil 4.8 Tek Noktalı Çaprazlama (Eiben, 2003)

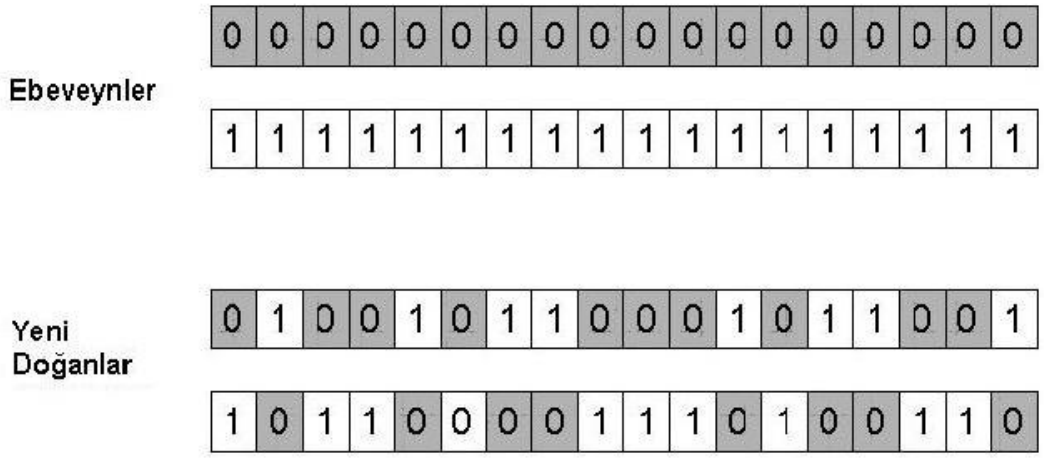
Çok noktalı çaprazlama Bu yöntemde birden fazla nokta için kromozomun uzunluğu dikkate alınarak üretilen rasgele sayılar çaprazlama yapılacak noktaları göstermektedir. Şekil 4.9 da yöntem gösterilmektedir.



Şekil 4.9 Çok Noktalı Çaprazlama (Eiben, 2003)

Tekdüze (uniform) çaprazlama Çok noktalı çaprazlamadan esinlenerek geliştirilmiş olan bu yöntem bir kaç nokta kümesi yerine her lokustaki bit değerini değiştirmeye dayanmaktadır. Ebeveynlerden birine "yazı" diğerine "tura" atanır. İlk yeni doğan için

yazı-tura atılarak aleleri belirlenir. Diğerinin aleleri ise ilkinin tam tersi olacak şekilde yeni çözüm adayları üretilir. Şekil 4.10'da ikili kodlanmış kromozomlar için örnek bir tekdüze çaprazlama gösterilmektedir.



Şekil 4.10 Tekdüze Çaprazlama (Eiben, 2003)

Aritmetik çaprazlama Reel sayılarla kodlanmış kromozomlar genellikle aritmetik işlemler ile çaprazlanır.

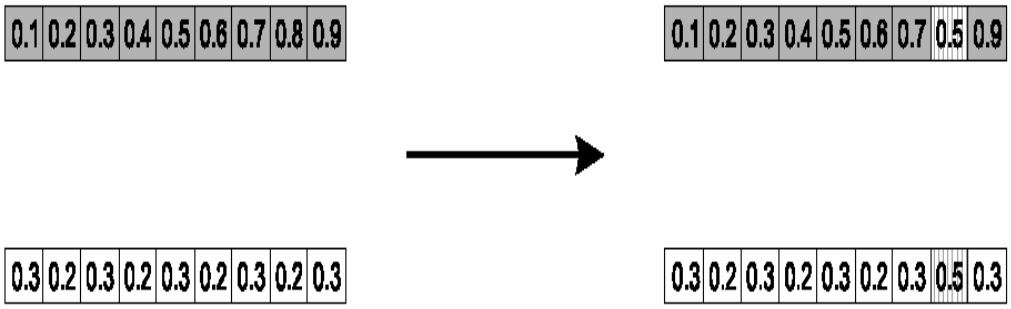
1) Ebeveynler: $<x_1,\ldots,x_n>$ ve $<y_1,\ldots,y_n>$

2) Rastgele herhangi bir kromozom alınır,

3) Yeni bireylerden biri: $\alpha\, y_k+(1-\alpha)x_k$ ile üretilir

4) Diğeri ise : $(1-\alpha)\, y_k+\alpha\, x_k$ ile üretilir.

α =0.5 değeri için anne ve baba kromozomların sekizinci genlerine yukarıdaki kural uygulandığında yeni bireyler şu şekilde belirlenir.



Şekil 4.11 Aritmetik Çaprazlama (Eiben, 2003)

4.2.3 Genetik Algoritmalarda Mutasyon

Mutasyondaki amaç genetik çeşitliliği korumaktır bunun için de mutasyon operatörü ile varolan bir kromozomun genlerinin bir ya da bir kaçının değerleri değiştirilerek yeni kromozomlar elde edilir. Çaprazlama ile çözüm uzayı büyük adımlar ile taranmaktayken mutasyon operatörü ile çözüm uzayında ayak basmadık yer bırakılmamaya çalışılır. Genetik algoritmalarda çaprazlamadan sonra çözüm uzayının taranmasında ikinci derecede rol oynar.

Yeniden ve sürekli yeni nesil üretimi sonucunda belirli bir süre sonra nesildeki kromozomlar birbirini tekrarlamaya başlar ve bunun sonucunda da farklı kromozom üretimi durabilir veya çok azalabilir. Bu nedenle bir nesildeki kromozomların çeşitliliğini arttırmak için kromozomlardan bazıları mutasyona tabi tutulur. Böylelikle popülasyondaki bireyler arası farklılık korunmaktadır (İşçi, 2003).

Şekil 4.12 Mutasyon Örneği (İşçi, 2003)

Şekil 4.12'de ikili kodlanmış bir kromozomda örnek bir mutasyon işlemi gösterilmektedir. Mutasyon çaprazlamadaki gibi programda 0-1 aralığında rasgele üretilen bir sayının önceden belirlenen mutasyon oranından büyük olması durumunda gerçekleştirilir. Tipik olarak mutasyon oranı 0.005-0.5 aralığında seçilir. Bu değer ve çaprazlama oranı değiştirilerek daha uygun çözümler elde edilebilir. Eğer mutasyon oranı büyük bir değer seçilirse genetik arama rastsal aramaya dönüşür. Fakat bu da daha önce kaybolan veya bulunamamış olan iyi çözümlerin elde edilmesini sağlayabilir.

4.3 RYB Sistemlerini Konuşlandırma Programı

RYB sistemlerini sınırları belirli herhangi bir bölgeye optimum olarak konuşlandırmak için geliştirilen programda kullanılan yöntem ve algoritmaların teori ve matematiksel modelleri yukarıdaki bölümlerde ayrıntılı olarak açıklanmıştır. Bu bölümde MATLAB mühendislik aracı kullanılarak geliştirilen programın ara yüz ve çalışma zamanında oluşan koşulların olduğu ekranlar ve programın kullanım kılavuzu anlatılmaktadır.

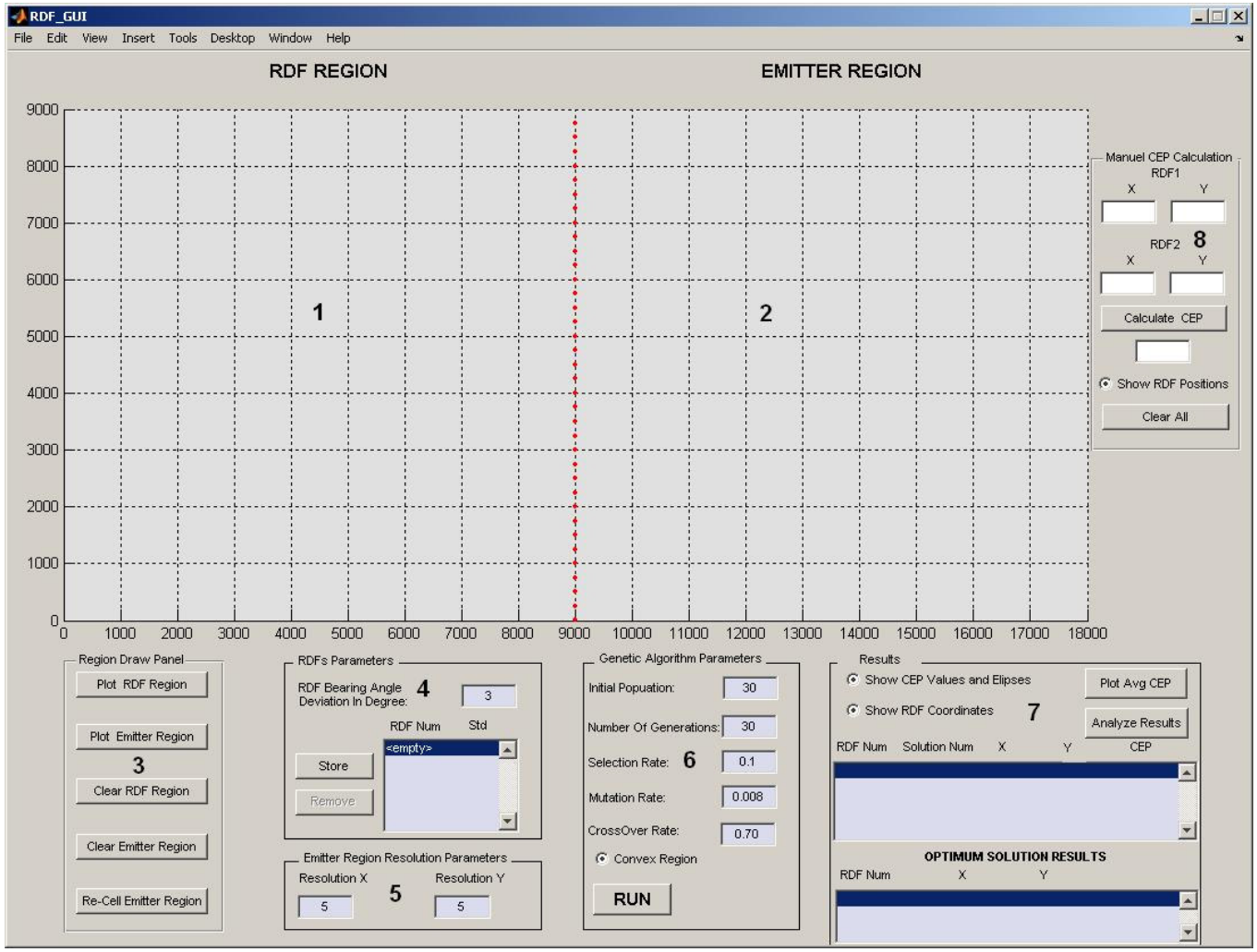
Şekil 4.13'de gösterilen programın ara yüzü şekilde de gösterildiği gibi sekiz bölümden oluşmaktadır.

Bir numara ile gösterilen ve *RDF Region* (RYBS Bölgesi) adı verilen alan kullanıcı tarafından *Region Draw Panel*'den (Bölge Çizim Paneli) *Plot RDF Region* (RYBS Bölgesi Çiz) butonuna basıp herhangi bir kısıtlama olmaksızın fare yardımı ile poligonel olarak çizilen RYB sistemlerinin bulunabileceği alanın tanımlandığı uzayı göstermektedir. Poligon olarak tanımlanan ve Şekil 4.14'te de örnek bir alan üzerinden gösterilen bu bölge optimizasyon algoritmasında arama uzayını temsil etmektedir. Konuşlandırılacak olan ve kullanıcının dilediği sayıda girebileceği RYB sistemleri bu alanın içinde kalacaktır.

İki numara ile gösterilen ve *Emitter Region* (Hedef Bölgesi) adı verilen bölge ise yine kullanıcı tarafından *Region Draw Panel*'den *Plot Emitter Region* (Hedef Bölgesi Çiz) butonuna basıp herhangi bir kısıtlama olmaksızın fare yardımı ile poligonel olarak çizilen, radyo kaynaklarının bulunabileceği alanın tanımlandığı bölgeyi temsil

etmektedir. Poligon olarak tanımlanan bu bölgede radyo kaynaklarının nerelere yerleştirildiği bilinmemektedir. Alanın herhangi bir yerine yerleştirilmiş olduğu varsayılan radyo kaynaklarından yayınlar alındığı varsayılmıştır. Bu varsayımlar temel alınarak, kullanıcı tarafından tanımlanan alan hayali küçük dikdörtgenlere ayrılarak her bir dikdörtgenin orta noktasında bir radyo kaynağının yer aldığı varsayılmıştır. Poligonun içinde kalan bu noktalar kırmızı renkte, dışında kalan noktalar ise Şekil 4.14'te de gösterildiği gibi pembe renkte gösterilmektedir. *RDF Region* da *Emitter Region* da Şekil 4.13'te de görüldüğü gibi 9.000x18.000m'lik bir alan üzerinde çizilmektedir. CEP değerleri de metre birimindedir.

Program otomatik olarak, çizilen bölgeyi 5x5 büyüklüğünde dikdörtgenlere ayırmaktadır. Beş numara ile gösterilen panel yardımı ile alanın istenilen çözünürlükte dikdörtgenlere ayrılması olanağı da bulunmaktadır. Kullanıcı bu panelde alanın x ve y eksenlerinde kaç küçük parçaya ayrılacağını klavye yardımı ile girdikten sonra *Region Draw Panel*'den *Re-Cell Emitter Region* (Hedef Bölgesini Yeniden Bölümlere Ayır) butonuna basarak radyo kaynaklarının bulunduğu bölgenin çözünürlüğünü artırıp azaltabilmektedir. Amaç radyo kaynaklarının bulunduğu alanın bütün olarak istenilen çözünürlükte kapsanabilmesi olduğu için bu özellik programda yer almaktadır. Alanın yeniden verilen sayılarda dikdörtgenlere ayrılması program tarafından otomatik olarak gerçekleştirilmektedir.



Şekil 4.13 RYBS Konuşlandırma Programı Ön Yüzü

Kullanıcı dilediği zaman *Region Draw Panel*'den *Clear Emiter Region* (Hedef Bölgesini Sil) veya *Clear RDF Region* (RYBS Bölgesini Sil) butonlarına basarak bir önceki belirlenmiş olan RYB sistemlerinin veya radyo kaynaklarının bulunabileceği alanları silip yenilerini yukarıda anlatıldığı gibi tekrar tanımlayabilmektedir.

Dört numaralı bölgede yer alan *RDFs Parameters* (RYBS Parametreleri) paneli kullanıcıya herhangi bir kısıtlama olmaksızın dilediği sayıda ve dilediği hassasiyette RYB sistemi tanımlama olanağının verildiği paneldir. Burada kullanıcı yalnızca, tanımlanan alana konuşlandırmak istediği RYB sistemlerinin ölçüm hatalarının standart sapmalarını derece olarak girmektedir. Program kaç tane RYB sistemi girildiğini otomatik olarak kendisi hesaplamaktadır. *Store* (Yükle) butonu verilen standart sapma değerlerini en küçük kareler algoritmasında kullanılmak üzere program tarafından hafızaya alınmasını sağlamaktadır. *Remove* (Kaldır) butonu ise kullanıcının daha önceden sisteme girmiş olduğu RYB sisteminin kaldırılması istendiğinde kullanılmaktadır.

Altı numara ile gösterilen panel programda optimizasyon algoritması olarak kullanılan genetik algoritmanın ihtiyaç duyduğu parametrelerin girildiği paneldir. Ayrıca bu panelde GA ile doğrudan ilgili olmamakla beraber çizilen *RDF Region*'ın konveks olup olmadığını da kullanıcı programa bildirmek zorundadır. Eğer tanımlanan alan konveks ise program tek-noktalı çaprazlama yapmakta, değil ise aritmetik çaprazlama yapmaktadır. Böyle bir farklılığa gidilmesinin nedeni konveks bir alanda, alanın dışında çözüm adaylarının üretilmesinin önüne geçmektir.

Burada yer alan *Initial Population* (Populasyon Sayısı) genetik algoritmada başlangıçta kaç tane çözüm adayının üretileceğinin belirtildiği alandır. Burada belirtilen sayı kadar tekdüze dağılıma sahip ilk çözüm adayları rastsal olarak RYB sistemlerinin tanımlandığı bölge sınırları içinde Şekil 4.14'te de gösterildiği gibi üretilmektedir. Daha sonra kuşaklar boyunca evrilecek olan çözüm adaylarının evrilme yöntemini güdümleyecek olan parametreler *Selection Rate* (Seçilme Oranı), *Mutation Rate* (Mutasyon Oranı) ve *CrossOver Rate* (Çaprazlama Oranı) ve kuşak sayısını gösteren *Number Of Generations* (İterasyon Sayısı) alanları bu panelin içinde kullanıcının gerekli değerleri girebilmesi için yer almaktadır. Program başladığında bu alanlar ilk değerleri ile kullanıcıya verilmektedir. Kullanıcı dilerse bu alanları değiştirebilir.

Yedi numara ile gösterilen alan ise programın çalışmasının ardından kullanıcıya sonuçların verildiği alandır. Bu alandaki butonlar ve liste kutucukları ayrıntılı olarak aşağıdaki bölümlerde anlatılmaktadır.

Sekiz numara ile gösterilen *Manuel CEP Calculation* panelinde ise kullanıcı iki RYB sisteminin koordinatlarını kendisi girip konuşlandırarak CEP değerini hesaplayabilmektedir. Bu panel ile kullanıcıya GA kullanmaksızın hesaplama yapabilme olanağı tanınmaktadır.

GA parametre değerlerinin optimizasyon performansına ve programın çalışma zamanına etkileri

Initial Population (Çözüm Adayları Sayısı) Çözüm adaylarının sayısını gösteren bu parametrenin değeri arttıkça programın çalışması yavaşlamaktadır. Çünkü her bir çözüm adayı uygunluk fonksiyonuna sokularak uygunluk değeri yani olası dairesel hata (ODH) hesaplanmaktadır. Fakat çözüm adaylarının çok olması genetik algoritmalarda sıklıkla karşılaşılan popülasyon farklılığının kaybedilmesine, yani erken dönemde çözüm adaylarının birbirine yakın değerler alarak tüm çözüm uzayı araştırılmadan erken yakınsama yapma problemine önerilen çözümlerden birisidir.

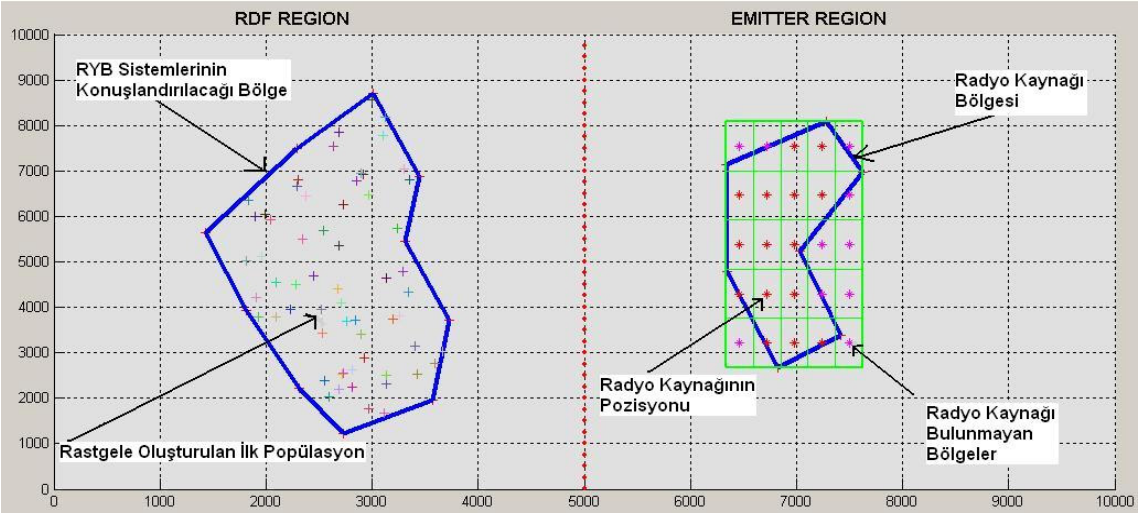
Selection Rate (Seçilme Oranı) Bu oran genetik algoritmadaki elitist yöntem için kullanılmaktadır. İterasyonlar sırasında en iyi çözüm adaylarından kaç tanesinin saklanacağını göstermektedir. Bu değer programın çalışma zamanını etkilememektedir, fakat optimizasyon performansını etkilemekte bulunan en iyi çözüm adayları korunarak optimum noktaların bulunması kolaylaşmaktadır.

Mutation Rate (Mutasyon Oranı) Bu oran genetik algoritmalarda yine popülasyon çeşitliliğini koruma amacıyla kullanılmaktadır ve genellikle çok küçük seçilmektedir. Büyük seçilmesi durumunda programın tamamen rasgele arama yapmaya başladığı optimum noktalara yakınsama yapmadığı gözlenmiştir.

CrossOver Rate (Çaprazlama Oranı) Bu oranın tipik değerleri önceki bölümlerde verilmektedir. Bu oranın programın çalışma zamanına bir etkisi yoktur. Optimum noktalara erişim performansı açısından ise tipik değerlerin genellikle iyi sonuçlar verdiği gözlemlenmiştir.

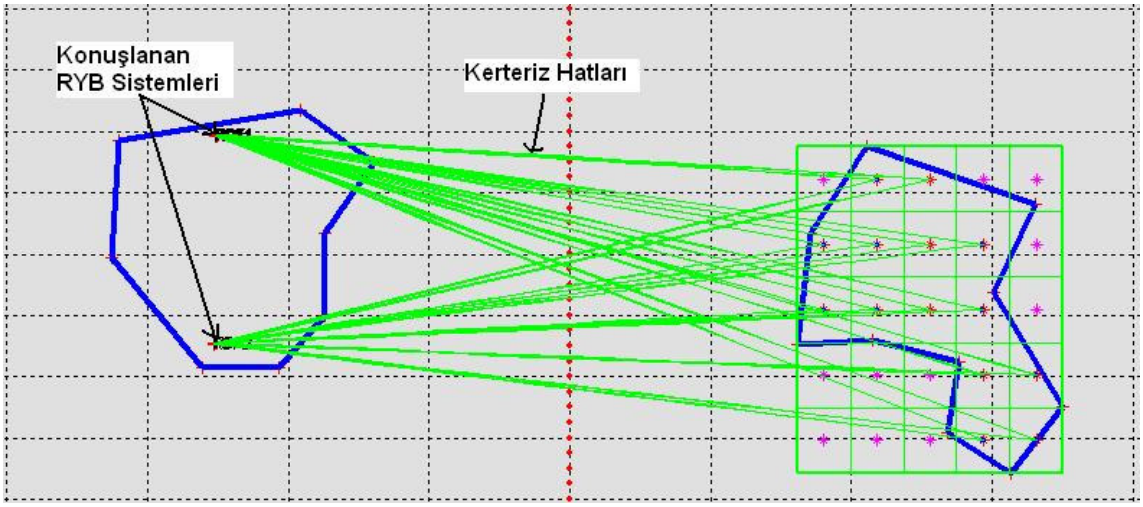
Number Of Generations (İterasyon Sayısı) Programın çalışma zamanını en çok etkileyen parametredir. İterasyon sayısını göstermektedir ne kadar çok olursa program o kadar uzun çalışmaktadır. Fakat programın çıktılarına bakıldığında belirli bir iterasyon sonrasında programın artık daha fazla iyileşmediği bulduğu optimum noktalarda takıldığı gözlemlenmiştir. Bundan dolayı kullanıcı deneme-yanılma yöntemi ile en uygun iterasyon miktarını kendisi belirlemelidir.

RYB sistemlerinin ve radyo kaynaklarının yer alacağı alanlar çizildikten ve RYBS'lerin hassasiyet değerleri ile genetik algoritmada kullanılacak olan parametrelerde programa girildikten sonra program *Run GA* butonu ile çalıştırılır. Bu butona basıldığında işlem bitinceye kadar optimizasyon işleminin devam ettiğini betimlemek için ekranda farenin imi değişerek kum saati şeklini alır. İşlem bittiğinde fare işaretçisi tekrar eski haline döner. Program işlem bitmeden ekranda başka butonlara basılmasına izin vermemektedir.



Şekil 4.14 RYBS ve Radyo Kaynaklarının Bölgelerini Belirleme ve İlk Popülasyonun Üretilmesi

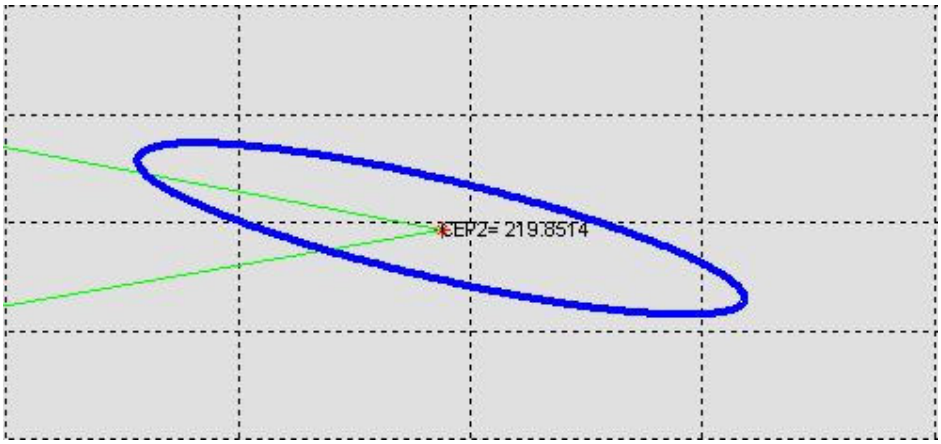
İşlem bittiğinde ise aşağıdaki Şekil 4.15'te de gösterildiği gibi RYB sistemlerinin kendilerine belirlenen alan içine konuşlanması ve konuşlanan bu istasyonlardan olası radyo kaynaklarına doğru çizilen kerteriz hatları programın çıktısı olarak verilmektedir.



Şekil 4.15 Program Çıktısı Örneği

İlk başta Şekil 4.14'te gösterildiği gibi kendileri için atanan uzayda rasgele dağılım yapan RYB sistemlerinin nesilden nesile evrilerek optimum noktalara yakınsama yapmaları Şekil 4.15'te gösterilmektedir.

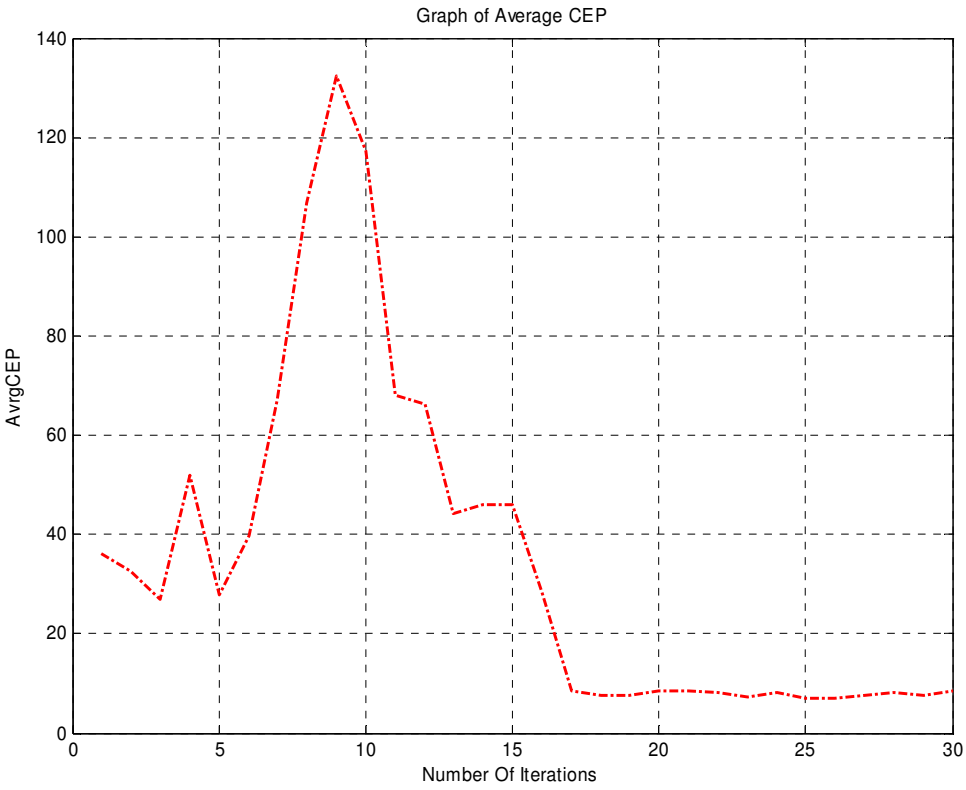
Yedi numaralı panelde yer alan *Show CEP Values and Ellipses* butonuna basılıp olası radyo kaynaklarının bulunduğu noktalara doğru şekil büyütüldüğünde bir radyo kaynağı için oluşan hata elipsi ve CEP değeri de gösterilmektedir. Örnek bir çıktı aşağıda Şekil 4.16'da gösterilmektedir.



Şekil 4.16 Hata Elipsi Çıktısı Örneği

Genetik algoritmada kullanılan uygunluk fonksiyonu radyo kaynakları için oluşan hata elipslerinden elde edilen olası eliptik/dairesel hata değerlerini geri döndürmektedir. Her hangi bir nesilde, tanımlı sınırları içinde konuşlanan olası her bir RYB istasyonu diziliminin her bir radyo kaynağı için oluşturduğu CEP değerleri hesaplanmaktadır. Daha sonra spesifik RYBS diziliminin tüm bölge için çıkardığı CEP değerlerinin ortalaması hesaplanır. İstenilen, radyo bölgesini ortalamada en küçük CEP değeri ile dinleyen konuşlandırmayı bulmak olduğu için optimizasyon algoritması CEP ortalamasını azaltmaya güdümlenerek çalışmaktadır.

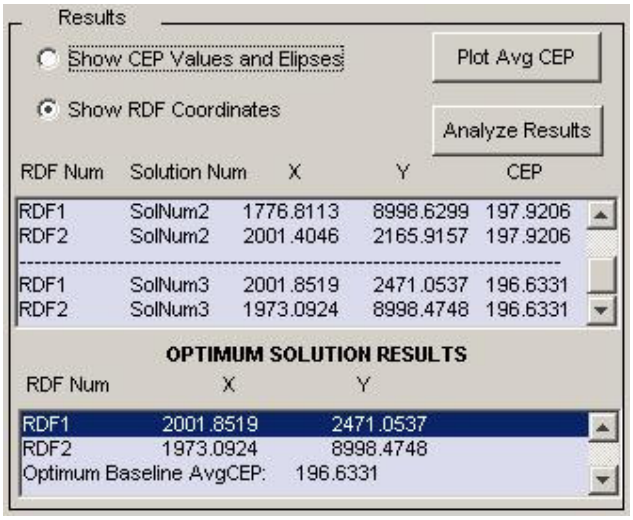
Çözüm adaylarının CEP metriğini kriter olarak alan evrim sürecini gösteren grafik de, Şekil 4.13'de yedinci sırada gösterilen *Results* (Sonuçlar) panelinde *Plot Avg CEP* (Ortalama CEP Grafiğini Çiz) butonuna basarak elde edilmektedir. Örnek bir program çıktısı aşağıda Şekil 4.17'de verilmiştir :



Şekil 4.17 Ortalama CEP Metriği Yakınsaması Örneği

Şekil 4.17'de de gösterildiği gibi tüm alan için ortalama CEP değeri, iterasyonlar boyunca azalmakta ve genetik algoritma RYB istasyonlarını optimum noktalara konuşlandırılmaktadır.

Yedinci bölümdeki diğer kontrol butonları ve alanlar ise sırasıyla : Daha öncesinde elips gösterimi ve CEP değeri gösteriminde de belirtilen *Show CEP Values and Ellipses* (Elipsleri ve CEP Değerlerini Göster) radyo butonu her bir çıkan elips bölgesi için CEP değerini ve elipsi göstermek/gizlemek için kullanılmaktadır. *Show RDF Coordinates* (RYBS'lerin Koordinatlarını Göster) radyo butonu da son iterasyonda oluşan çözüm adayları arasından en iyi uygunluk değerine sahip olan çözüm adayının, çözüm uzayındaki koordinatlarını göstermek/gizlemek için kullanılmaktadır. Bu bölümdeki diğer iki alan da ise *Analyze Results* (Sonuçları Analiz Et) butonu ile aktif hale gelmektedir. Optimizasyon algoritması olarak kullanılan GA'nın çalışma prensipleri rasgelelik üzerine kurulu olduğu için program her çalışmasında birbirine çok yakında olsa farklı sonuçlar üretmektedir. Genetik algoritmalarda prensip olarak bir kez programı çalıştırma ile alınan sonuçların yeterli güvenilirlikte olmayabileceği literatürde belirtilmektedir. Bundan dolayı programda kullanıcıya aynı senaryo için birden fazla çalıştırma olanağı verilmektedir. Her çalıştırmada elde edilen sonuçlar bir dosyaya yazıldıktan sonra Analyze Results butonuna basıldığında bu sonuçlar dosyadan okunup istatiksel bir değerlendirmeye tabii tutulmaktadır. Toplamda kaç kez çalıştırıldı ise program sonuçların ortalama değerleri ve standart sapmaları hesaplanmakta ve her bir çalıştırmada elde edilen çözüm adayları diğerleri ile karşılaştırılmaktadır. Bunlar arasından en uygun çözüm adayı da yine yedinci bölümde yer alan *Optimum Solution Results* (Optimal Çözüm Sonuçları) başlığı altında yer alan alana yazdırılıp kullanıcıya verilmektedir. Aşağıdaki şekilde, verilen herhangi bir senaryo için program iki kez çalıştırıldığında elde edilen sonuçlar yer almaktadır:



Şekil 4.18 Optimum Çözüm Sonucu Örneği

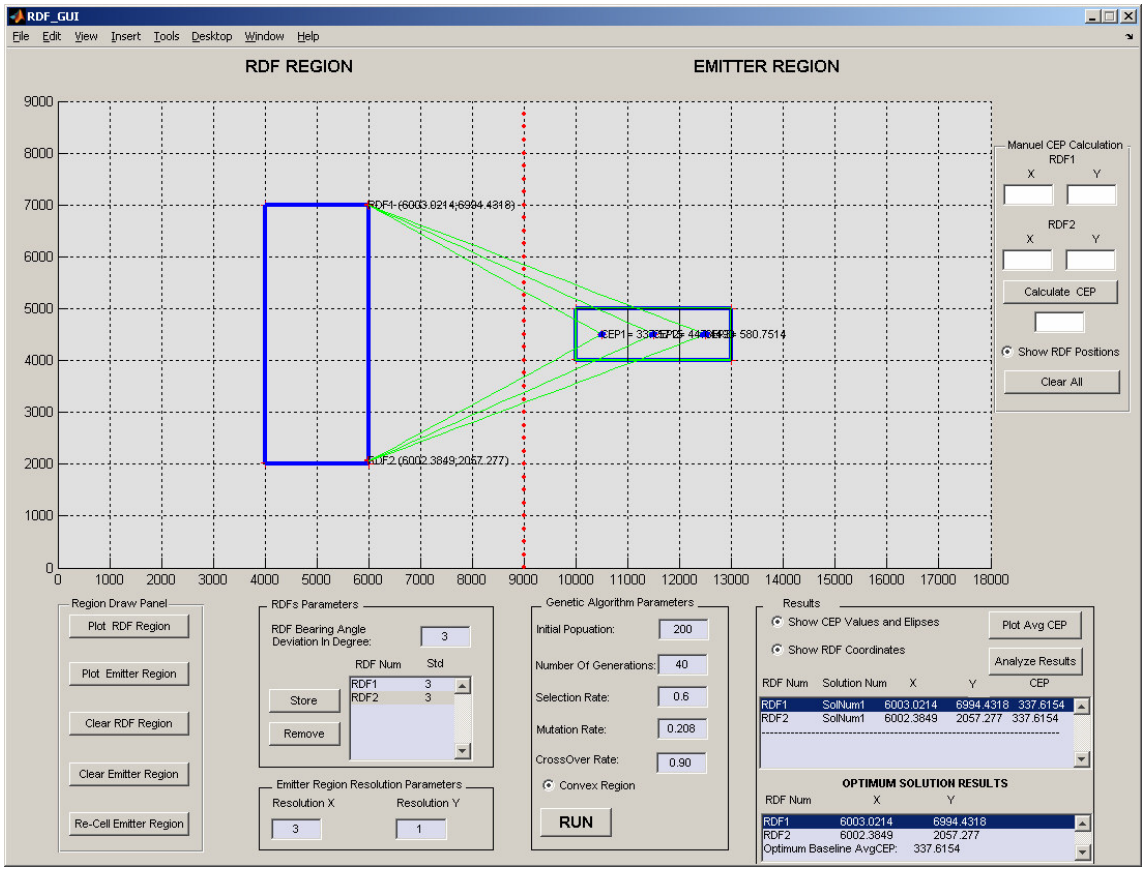
5. SİMULASYONLAR

Bu bölümde RYB sistemlerinin optimum konuşlandırma problemi için geliştirilen programda çalıştırılmak üzere altı farklı senaryo belirlenerek her bir senaryo için programın çıktıları üzerinde yorum yapılmaktadır.

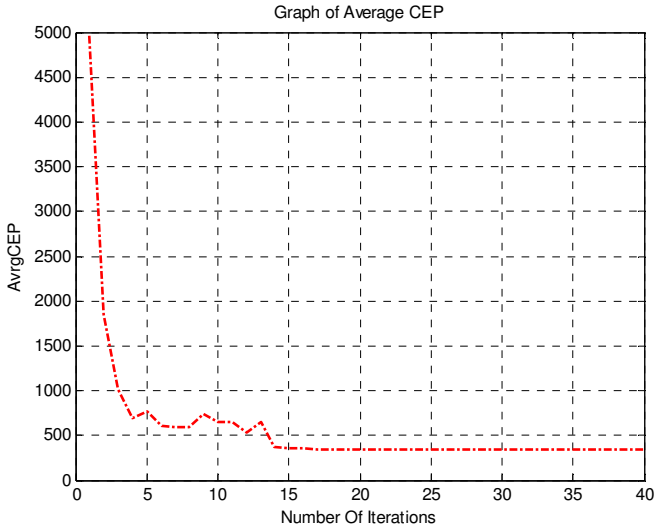
5.1 Senaryo 1

İlk senaryoda RYBS bölgesi de hedef bölgesi de dikdörtgen biçiminde tanımlanmıştır. Hedef bölgesi 3x1'lik çözünürlükte 3 parçaya ayrılmıştır. Her bir parça da dikdörtgen şeklinde olup her birinin ortasında elektromanyetik yayın yapan bir hedef olduğu varsayılmaktadır. Senaryoda RYB bölgesine iki adet radyo yön bulma sistemi yerleştirilmesine karar verilmiştir. Her iki sistemin de kerteriz açısı ölçüm hata standart sapma değerleri eşit olarak alınmış ve 3^o olarak belirlenmiştir.

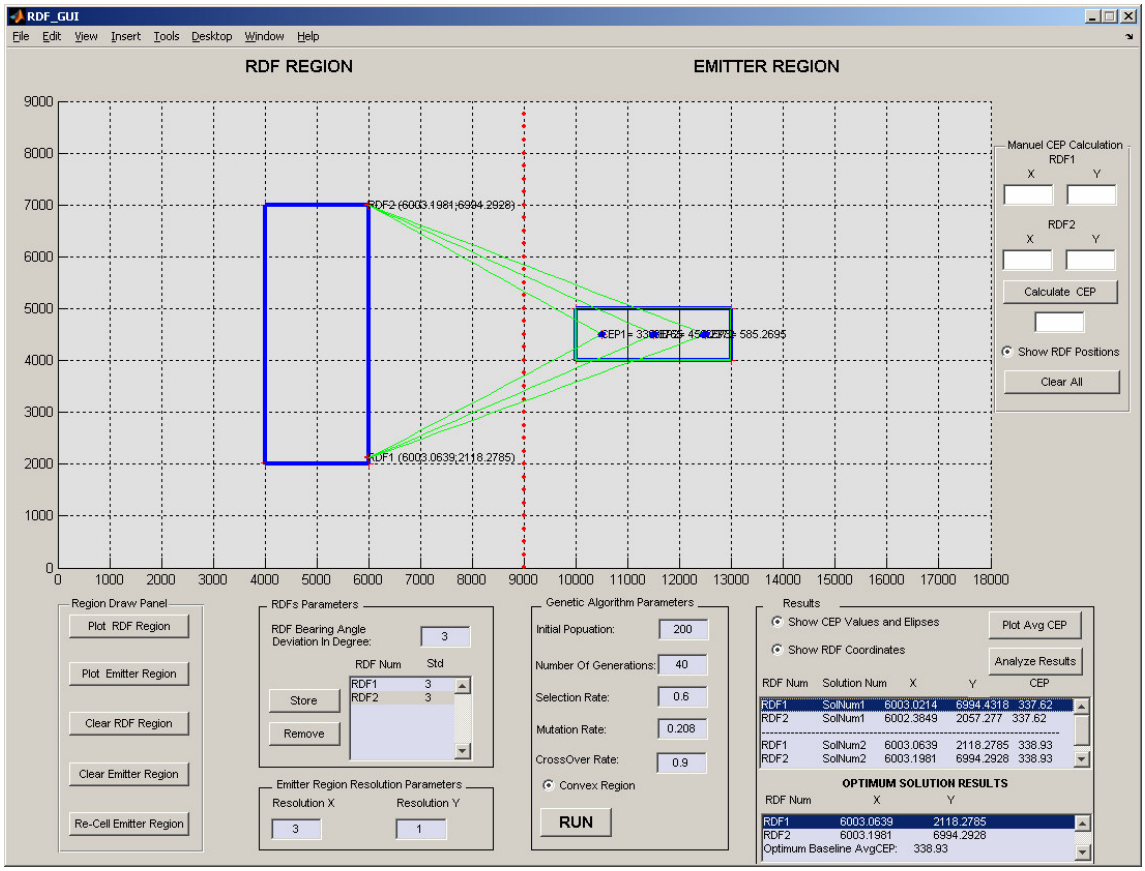
Genetik algoritma parametreleri 200 çözüm adayı ile çözüm uzayında en uygun değerlere sahip adayları belirlemek üzere 40 iterasyon yapacak şekilde ayarlanmıştır. En uygun adayların %60'ı nesiller boyu korunmaya alınarak, en uygun bireyler için çözüm arayışında mutasyon oranı 0.208 çaprazlama oranı da 0.90 olarak belirlenmiştir. Bu senaryo da belirlenen RYBS alanı konveks olduğu için *Convex Region* butonu 'on' durumuna getirilerek aritmetik çaprazlama yapılması sağlanmıştır. Senaryo üç kez yeni baştan çalıştırılarak üç çalıştırmada da elde edilen değerler *Results* panelinde kullanıcıya verilmiştir. Bunlara ait şekiller aşağıda sırasıyla verilmektedir :



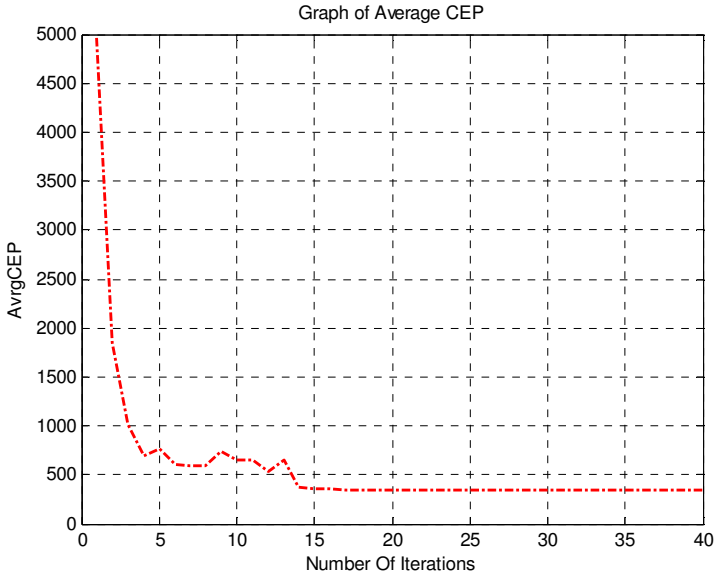
Şekil 5.1 Senaryo-1, I. Çalıştırma



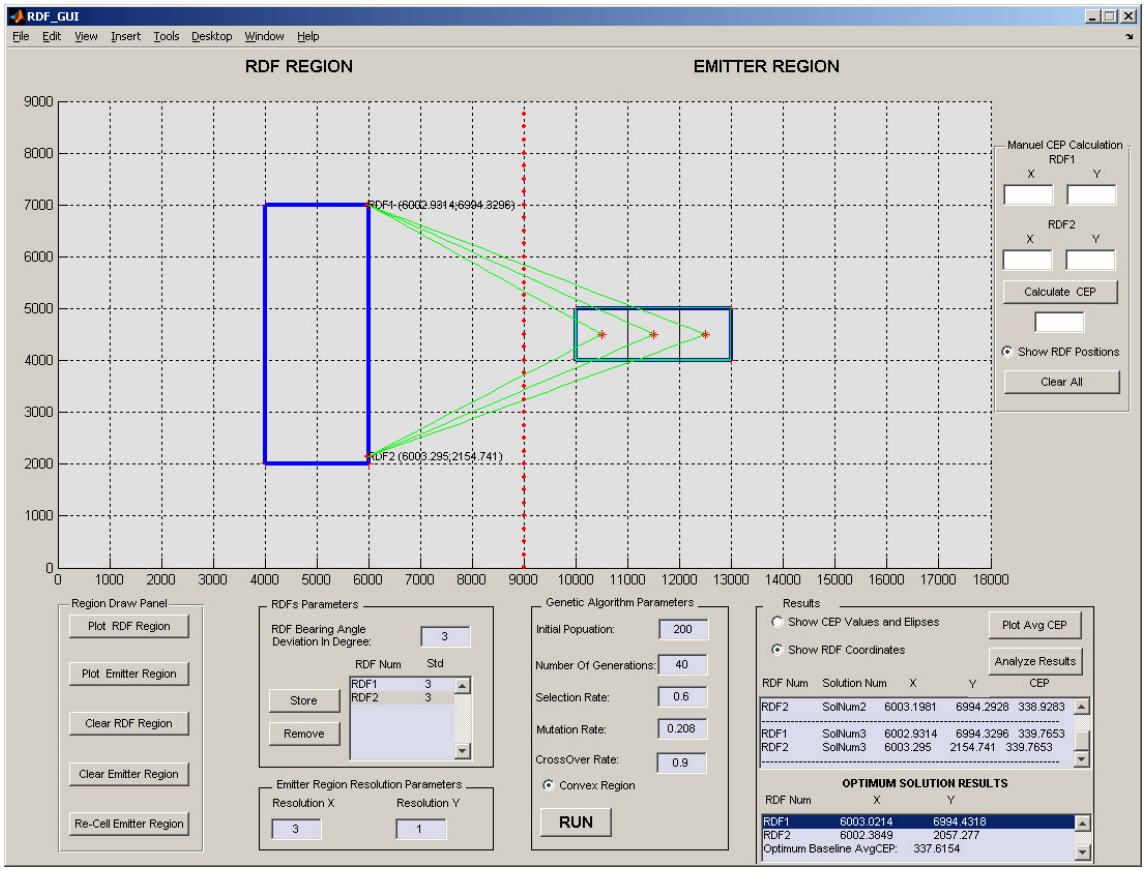
Şekil 5.2 Senaryo-1, I. Çalıştırma En İyi Ortalama CEP Değerleri



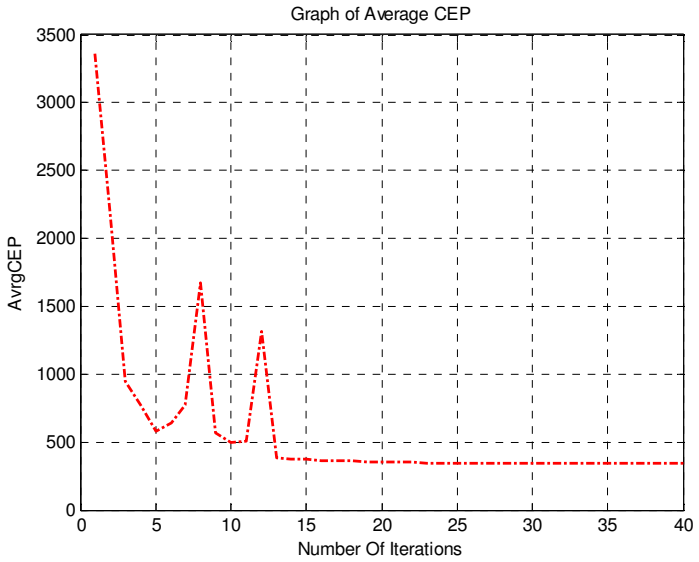
Şekil 5.3 Senaryo-1, II. Çalıştırma



Şekil 5.4 Senaryo-1, II. Çalıştırma En İyi Ortalama CEP Değerleri

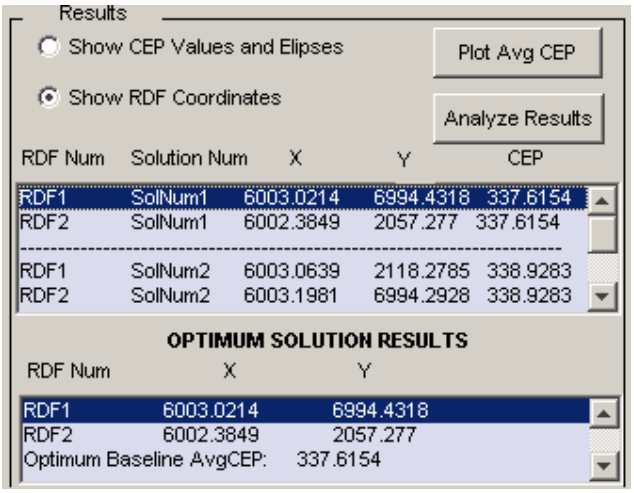


Şekil 5.5 Senaryo1, 3. Çalıştırma



Şekil 5.6 Senaryo-1, III. Çalıştırma En İyi Ortalama CEP Değerleri

Bu senaryo üç kez çalıştırıldıktan sonra elde edilen sonuçlar ise aşağıda Şekil 5.7'de verilmektedir.



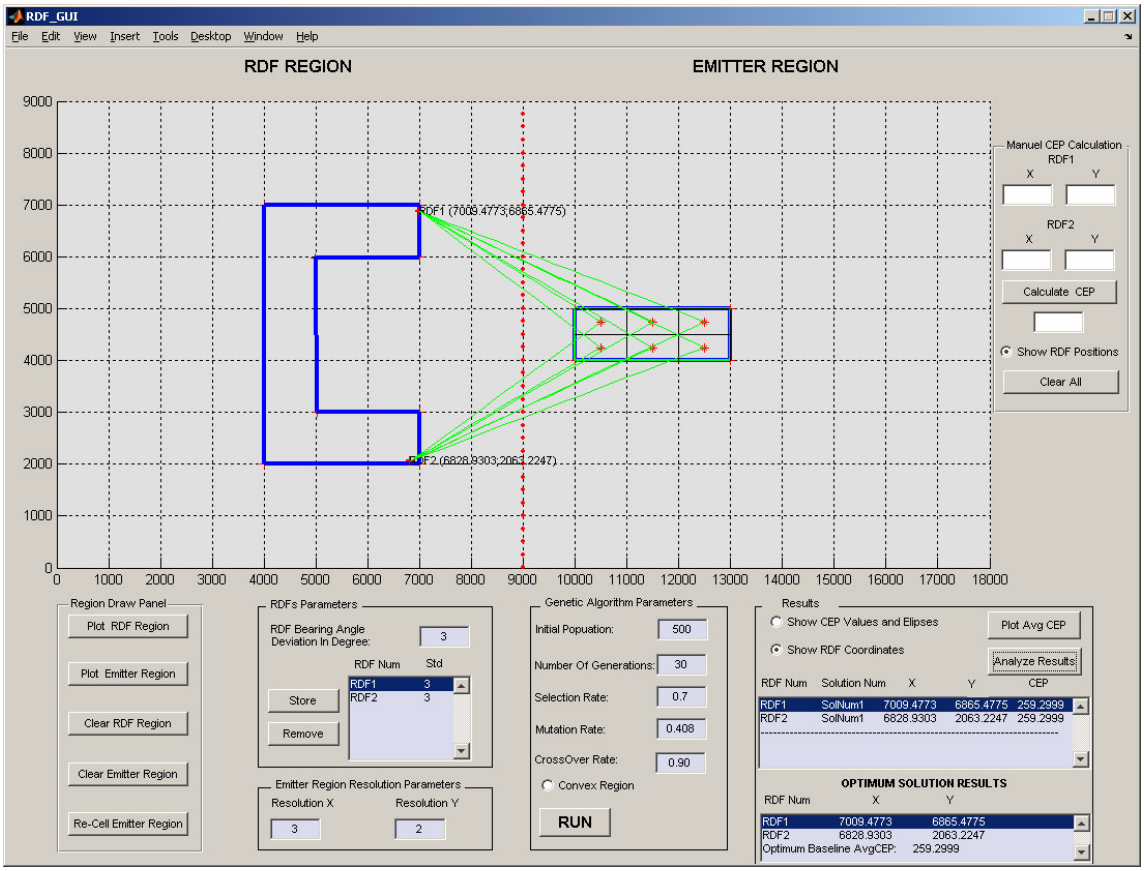
Şekil 5.7 Senaryo-1, Optimum Çözüm Sonuçları

Bu sonuçlara göre en iyi çözüm ortalamada en iyi CEP değerine sahip olan ilk çalıştırmada elde edilen çözümdür. Burada RDF1'in koordinatları (6003.0214; 6994.4318) ve RDF2'nin koordinatları (6002.3849; 2057.277) olarak bulunmuştur. RYBS'lerin yerleşim planı bu koordinatlara göre yapıldığı takdirde bu konuşlandırma için CEP değeri 337.62m çıkmaktadır. İkinci ve üçüncü çalıştırmalarda elde edilen CEP değeri ise sırasıyla yaklaşık olarak 338.93m ve 339.77m çıkmıştır.

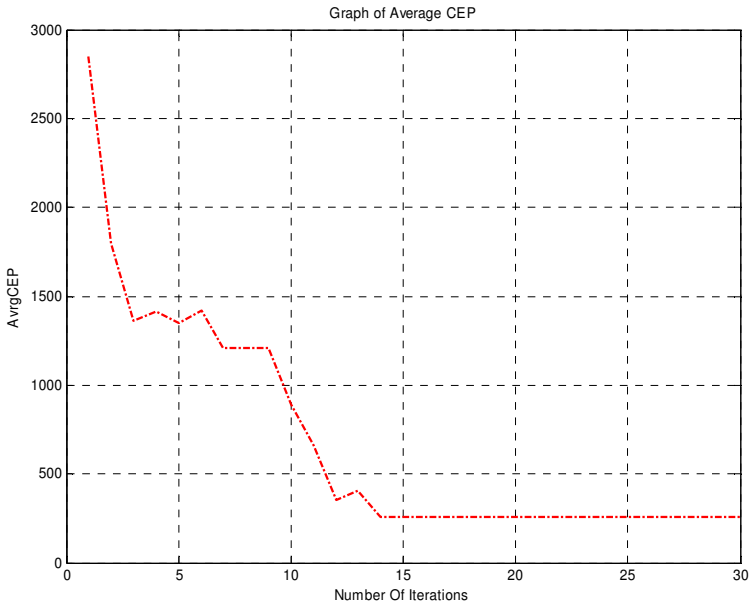
5.2 Senaryo 2

İkinci senaryoda RYBS bölgesi konveks olmayan bir bölge olarak, hedef bölgesi ise yatay dikdörtgen biçiminde tanımlanmıştır. Hedef bölgesi 3x2'lik çözünürlükte 6 parçaya ayrılmıştır. Her bir parça da dikdörtgen şeklinde olup her birinin ortasında elektromanyetik yayın yapan bir hedef olduğu varsayılmaktadır. Senaryoda RYB bölgesine iki adet radyo yön bulma sistemi yerleştirilmesine karar verilmiştir. Her iki sistemin de kerteriz açısı ölçüm hata standart sapma değerleri bir önceki senaryo da olduğu gibi eşit olarak alınmış ve 3^{o} olarak belirlenmiştir.

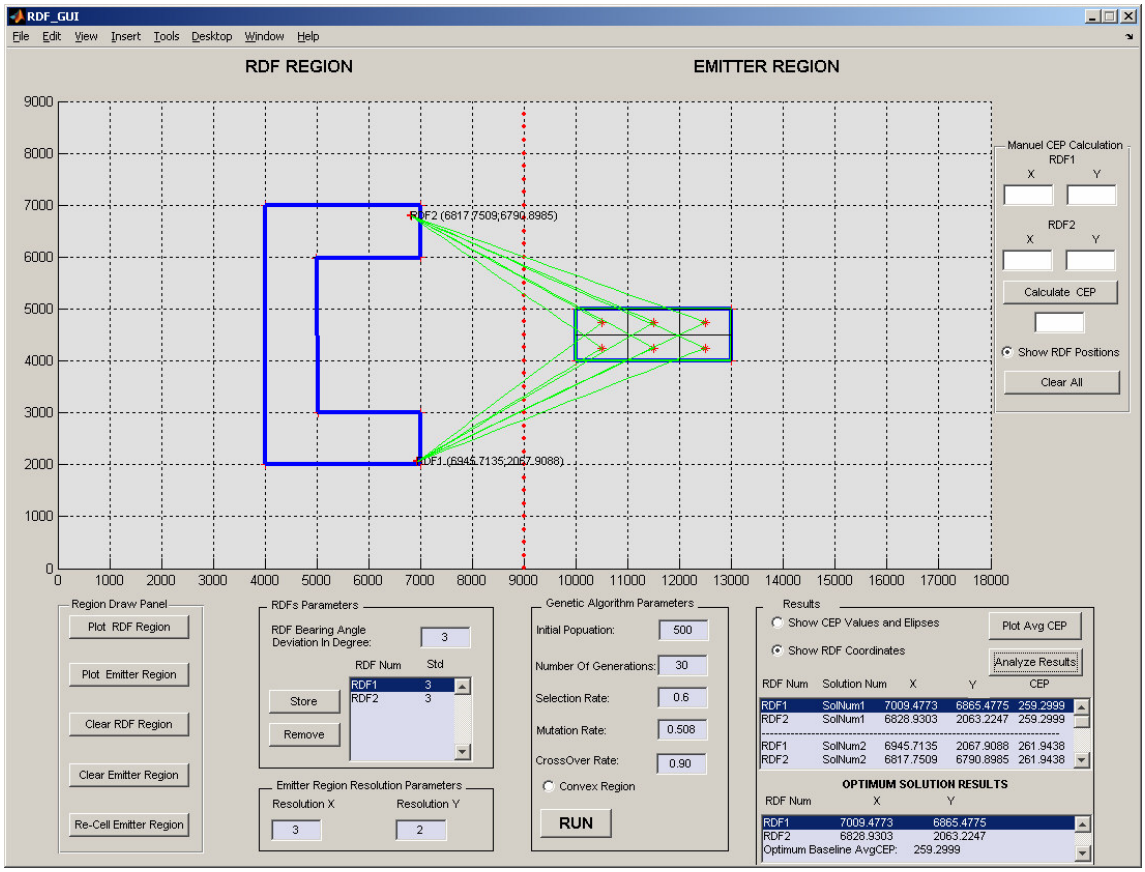
Genetik algoritma parametreleri 500 çözüm adayı ile çözüm uzayında en uygun değerlere sahip adayları belirlemek üzere 30 iterasyon yapacak şekilde ayarlanmıştır. En uygun adayların ilk çalıştırmada %70'i daha sonraki çalıştırmalarda ise %60'ı nesiller boyu korunmaya alınarak, en uygun bireyler için çözüm arayışında mutasyon oranı önce 0.408 daha sonra ise 0.508; çaprazlama oranı da 0.90 olarak belirlenmiştir. Bu senaryoda belirlenen RYBS alanı konveks olmadığı için *Convex Region* butonu 'off' durumuna getirilerek tek-noktalı çaprazlama yapılması sağlanmıştır. Çözüm adayı sayısının ve mutasyon oranının diğer konveks RYBS alanları içeren senaryolar için belirlenen değerlerden daha fazla olmasının nedeni de tek-noktalı çaprazlama ile bireylerin genetik yapılarındaki farklılaşmanın sadece değiş-tokuşa bağlı olmasıdır. Senaryo iki kez yeni baştan çalıştırılarak her iki çalıştırmada da elde edilen değerler *Results* panelinde kullanıcıya verilmiştir. Bunlara ait şekiller aşağıda sırasıyla verilmektedir :



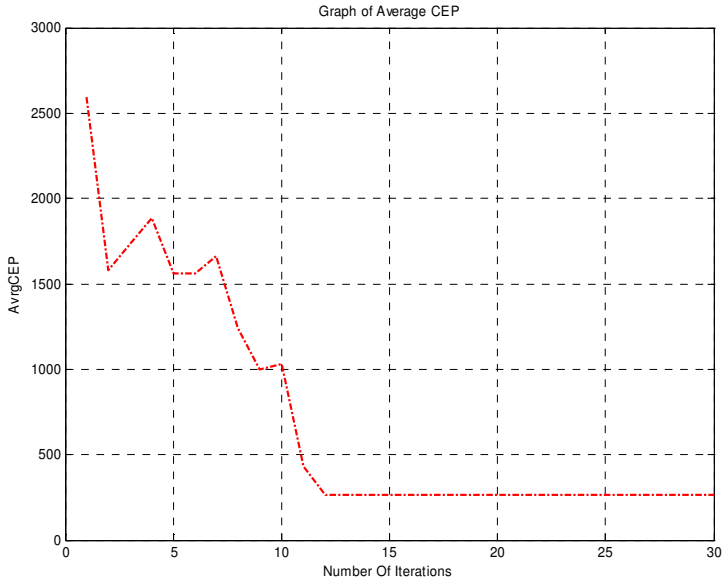
Şekil 5.8 Senaryo-2, I. Çalıştırma



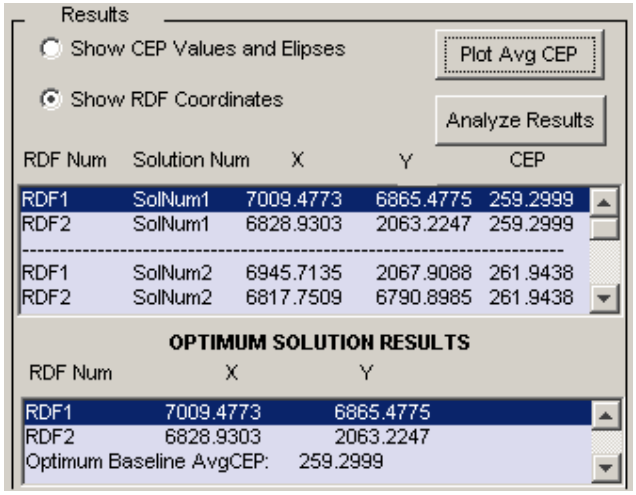
Şekil 5.9 Senaryo-2, I. Çalıştırma En İyi Ortalama CEP Değerleri



Şekil 5.10 Senaryo-2, II. Çalıştırma



Şekil 5.11 Senaryo-2, II. Çalıştırma En İyi Ortalama CEP Değerleri

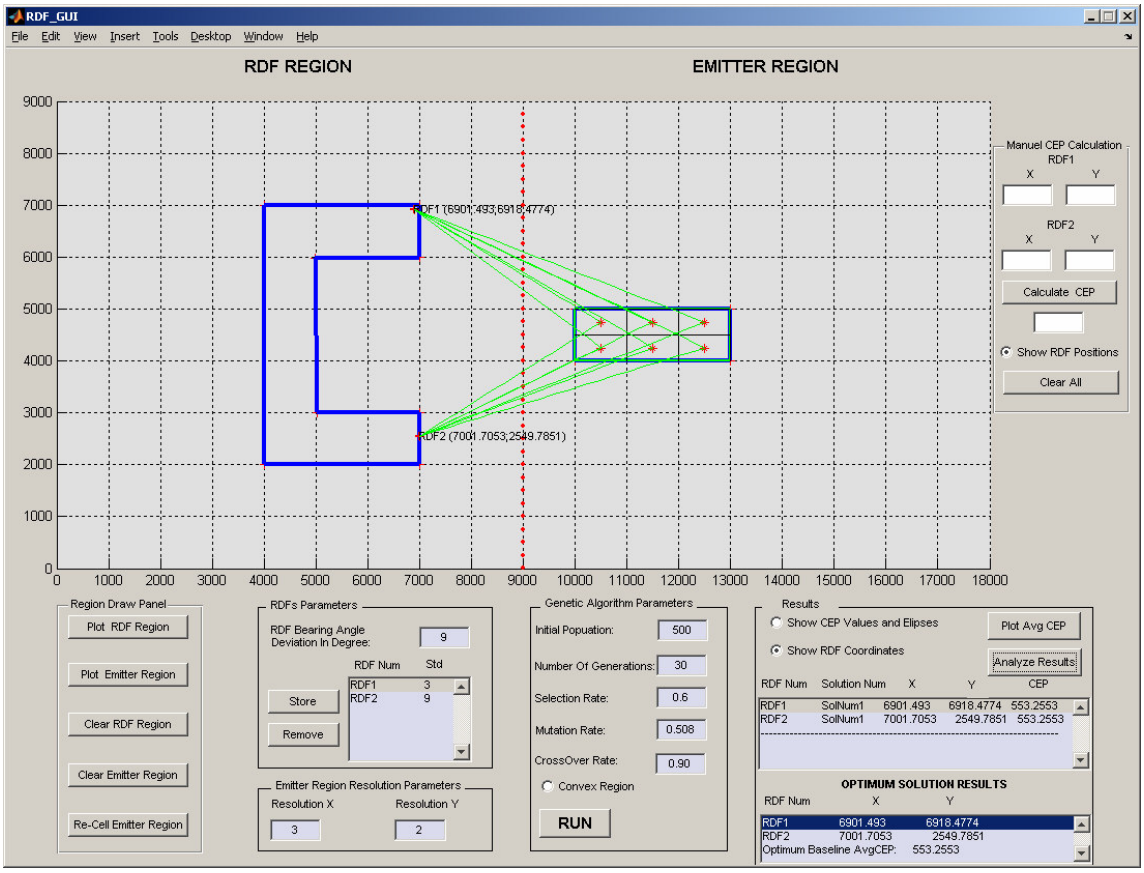


Şekil 5.12 Senaryo-2, Optimum Çözüm Sonuçları

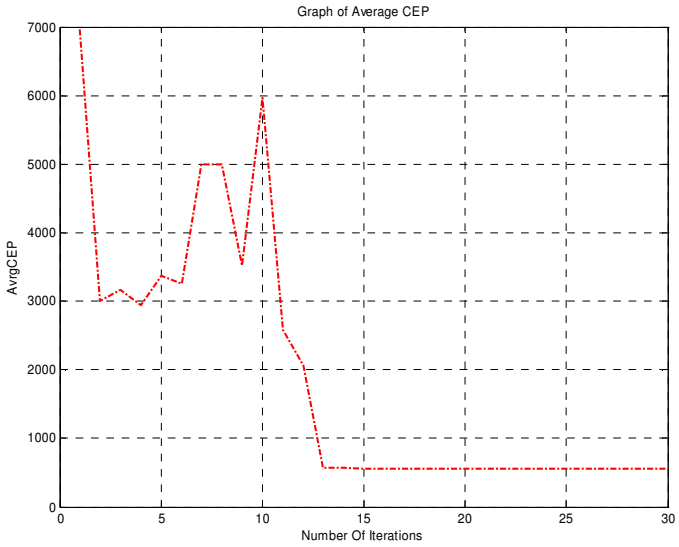
Bu sonuçlara göre en iyi çözüm ortalamada en iyi CEP değerine sahip olan birinci çalıştırmada elde edilen çözümdür. Burada RDF1'in koordinatları (7009.4773; 6865.4775) ve RDF2'nin koordinatları (6828.9303; 2063.2247) olarak bulunmuştur. RYBS'lerin yerleşim planı bu koordinatlara göre yapıldığı takdirde bu konuşlandırma

için CEP değeri 259.299 m çıkmaktadır. İkinci çalıştırmada elde edilen CEP değeri ise 261.944m'dir.

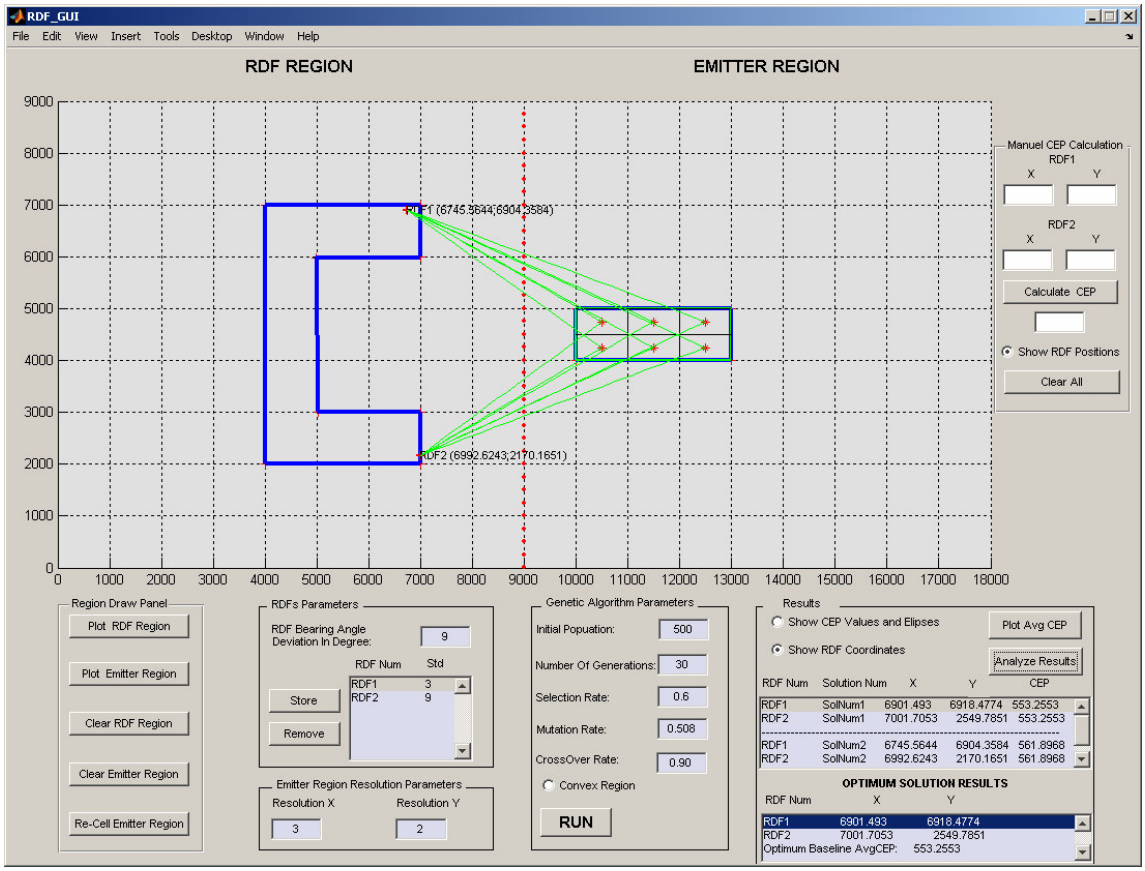
İkinci senaryo bu kez de RDF1'in ölçüm hassasiyet standart sapma oranı değiştirilmeden bırakılıp, fakat RDF2'nin ölçüm hassasiyet standart sapma oranı üç katına çıkarılarak (9 yapılarak) iki defa çalıştırılmış ve aşağıda sonucu verilmiştir.



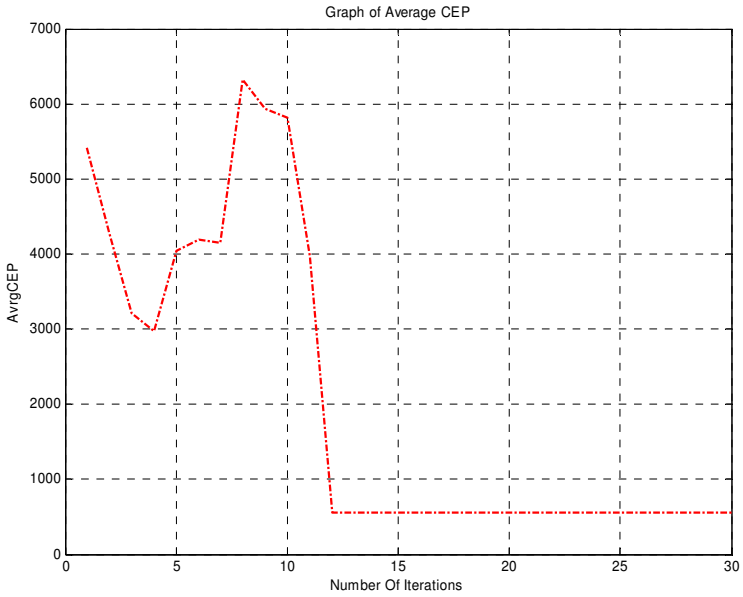
Şekil 5.13 Senaryo-2, Farklı Standart Sapmalı RYBS Kullanılması Sonucu-I



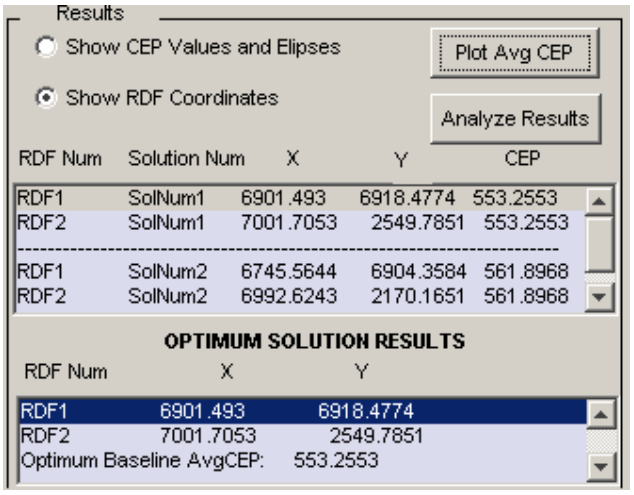
Şekil 5.14 Senaryo-2, Farklı Standart Sapmalı RYBS Kullanılması Sonucu-I En İyi Ortalama CEP Değerleri



Şekil 5.15 Senaryo-2, Farklı Standart Sapmalı RYBS Kullanılması Sonucu-II



Şekil 5.16 Senaryo-2, Farklı Standart Sapmalı RYBS Kullanılması Sonucu-II En İyi Ortalama CEP Değerleri



Şekil 5.17 Senaryo-2, Farklı Standart Sapmalı RYBS Kullanılması Sonucu-II Optimum Çözüm Sonuçları

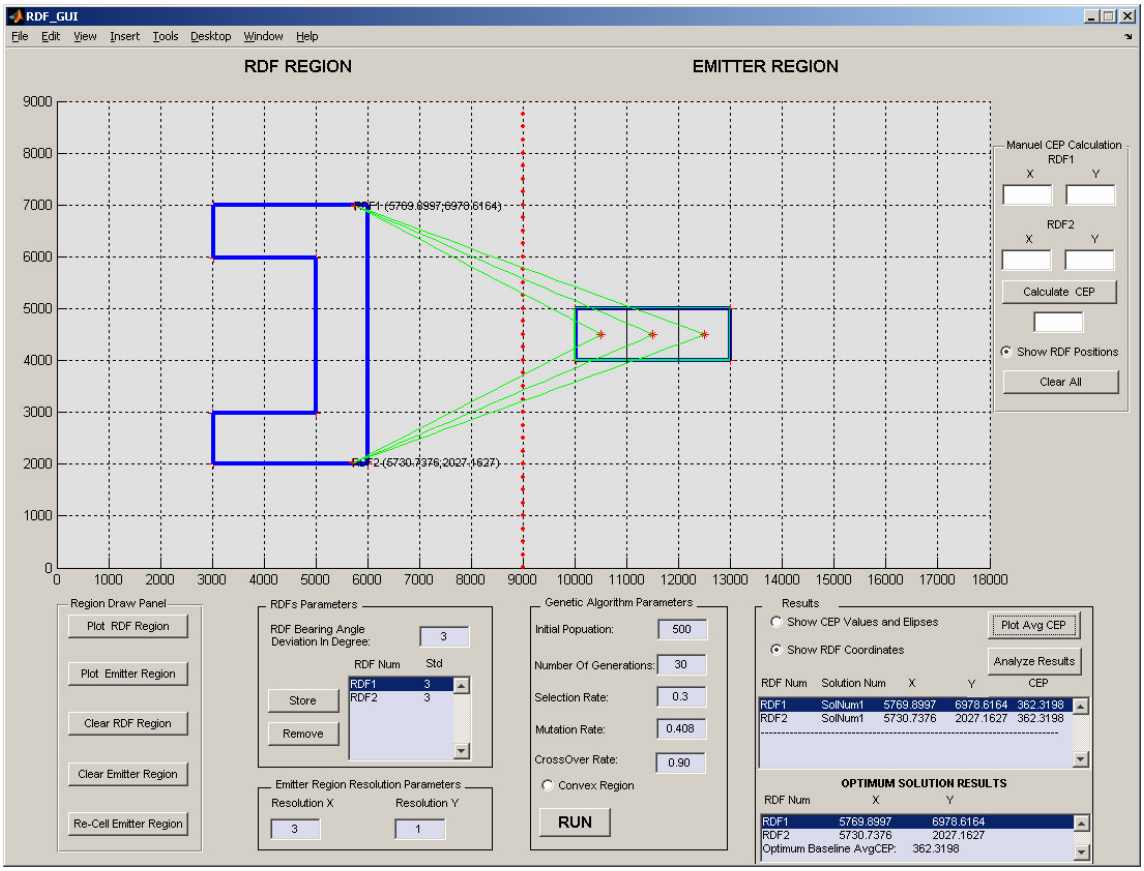
Bu sonuçlara göre en iyi çözüm ortalamada en iyi CEP değerine sahip olan birinci çalıştırmada elde edilen çözümdür. Burada RDF1'in koordinatları (6901.493; 6918.4774) ve RDF2'nin koordinatları (7001.7053; 2549.7851) olarak bulunmuştur. RYBS'lerin yerleşim planı bu koordinatlara göre yapıldığı takdirde bu konuşlandırma için CEP değeri 553.25m çıkmaktadır. İkinci çalıştırmada elde edilen CEP değeri ise

561.89m'dir. Sonuç olarak RYB sistemlerinden birinin hassasiyeti üç kat azaltıldığında CEP değeri yaklaşık olarak %114 oranında bir artış göstermiştir.

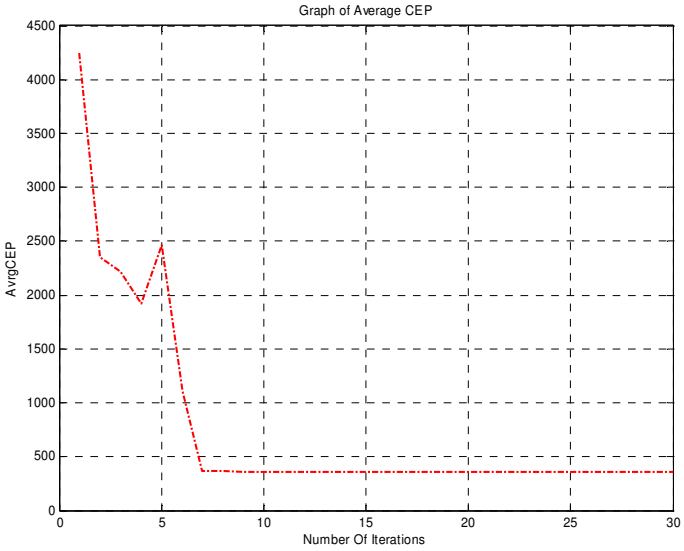
5.3 Senaryo 3

Üçüncü senaryoda RYBS bölgesi konveks olmayacak biçimde, hedef bölgesi de yatay dikdörtgen biçiminde tanımlanmıştır. Hedef bölgesi 3x1'lik çözünürlükte 3 parçaya ayrılmıştır. Her bir parça da dikdörtgen şeklinde olup her birinin ortasında elektromanyetik yayın yapan bir hedef olduğu varsayılmaktadır. Senaryoda RYB bölgesine iki adet radyo yön bulma sistemi yerleştirilmesine karar verilmiştir. Her iki sistemin de kerteriz açısı ölçüm hata standart sapma değerleri bir önceki senaryo da olduğu gibi eşit olarak alınmış ve 3^{o} olarak belirlenmiştir.

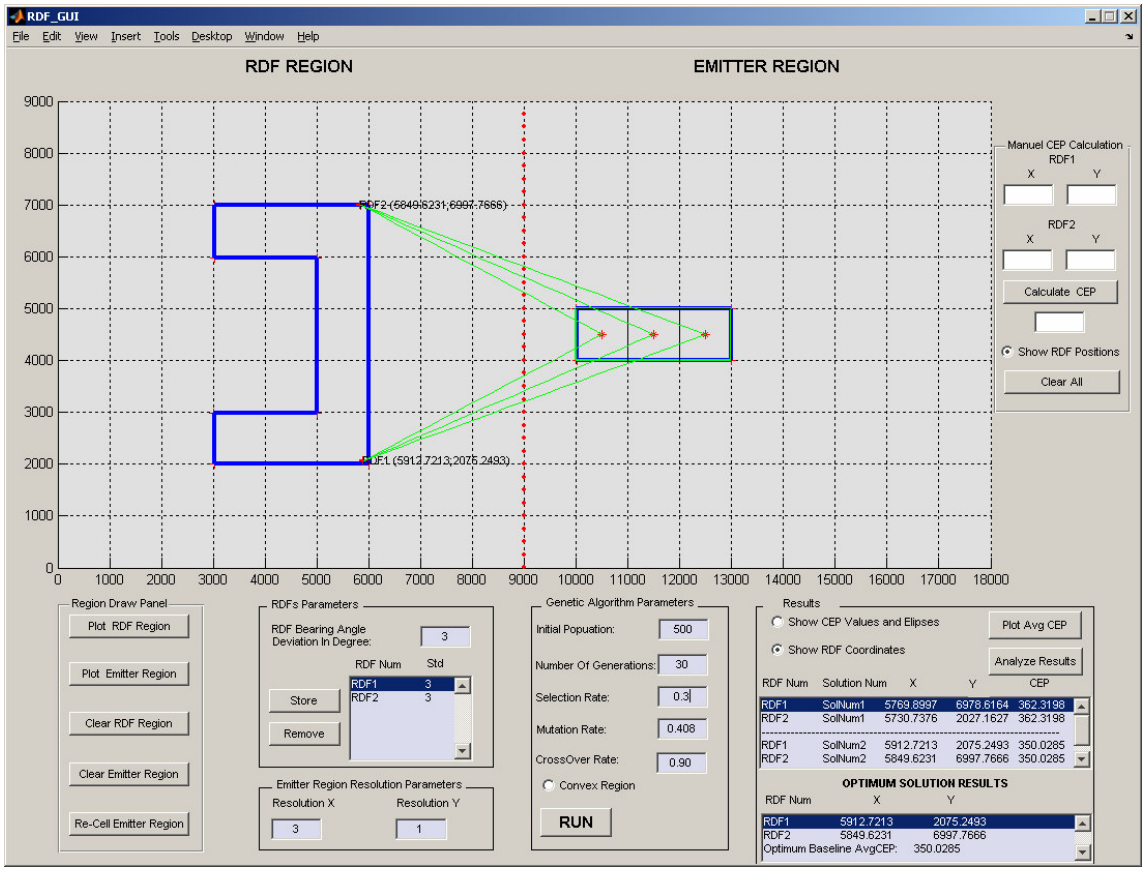
Genetik algoritma parametreleri 500 çözüm adayı ile çözüm uzayında en uygun değerlere sahip adayları belirlemek üzere 30 iterasyon yapacak şekilde ayarlanmıştır. En uygun adayların %30'ı nesiller boyu korunmaya alınarak, en uygun bireyler için çözüm arayışında mutasyon oranı 0.408; çaprazlama oranı da 0.90 olarak belirlenmiştir. Bu senaryoda belirlenen RYBS alanı konveks olduğu için *Convex Region* butonu 'off' durumuna getirilerek tek-noktalı çaprazlama yapılması sağlanmıştır. Bu senaryoda da GA parametre değerlerine ikinci senaryoda anlatılan bilgiler göz önünde tutularak büyük değerler verilmiştir. Senaryo iki kez yeni baştan çalıştırılarak her iki çalıştırmada da elde edilen değerler *Results* panelinde kullanıcıya verilmiştir. Bunlara ait şekiller aşağıda sırasıyla verilmektedir :



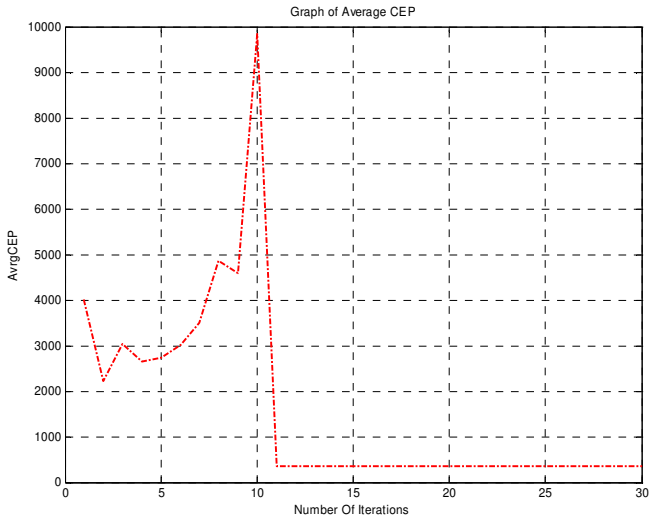
Şekil 5.18 Senaryo-3, I. Çalıştırma



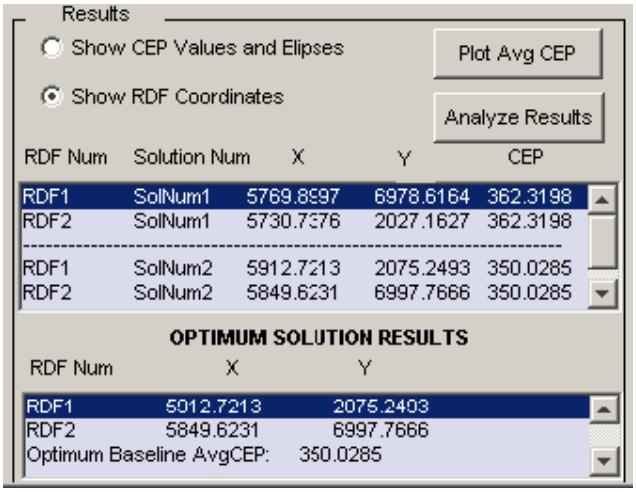
Şekil 5.19 Senaryo-3, I. Çalıştırma En İyi Ortalama CEP Değerleri



Şekil 5.20 Senaryo-3, II. Çalıştırma



Şekil 5.21 Senaryo-3, II. Çalıştırma En İyi Ortalama CEP Değerleri



Şekil 5.22 Senaryo-3, Optimum Çözüm Sonuçları

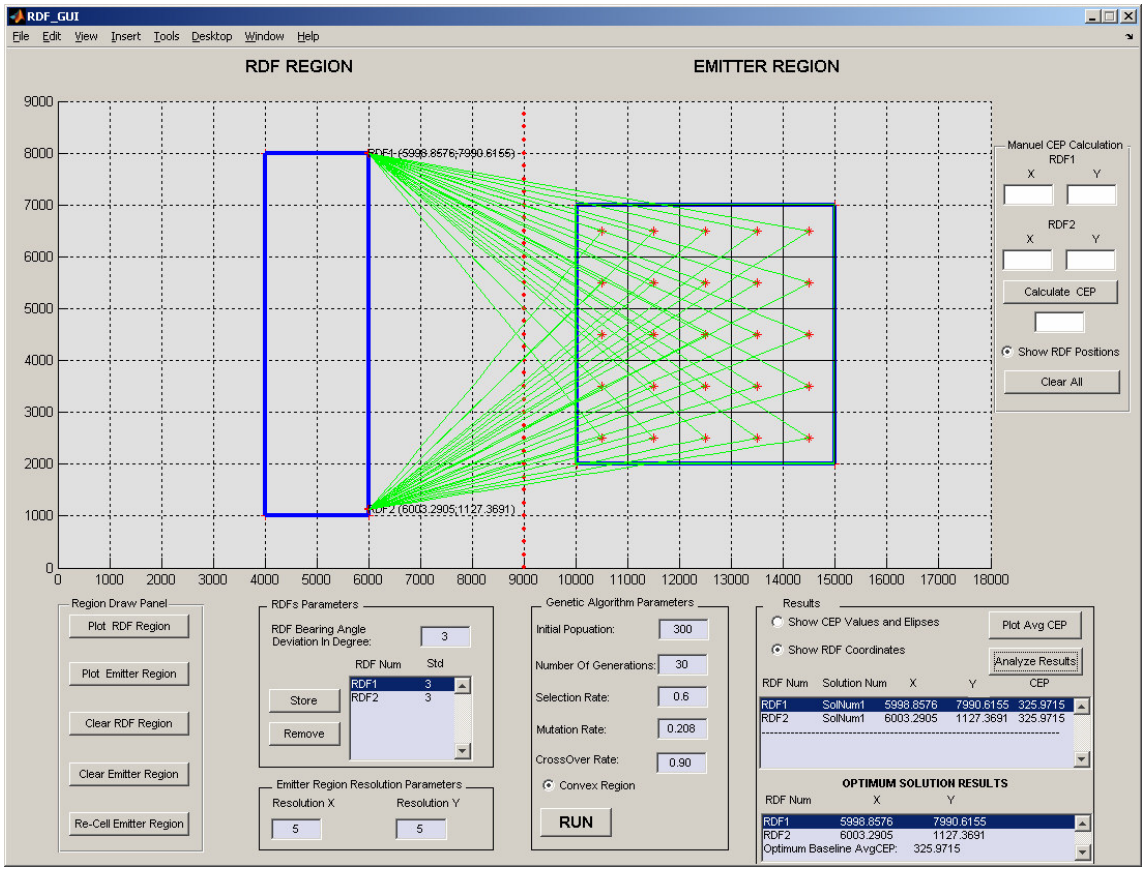
Bu sonuçlara göre en iyi çözüm ortalamada en iyi CEP değerine sahip olan ikinci çalıştırmada elde edilen çözümdür. Burada RDF1'in koordinatları (5912.7213; 2075.2493) ve RDF2'nin koordinatları (5849.6231; 6997.7666) olarak bulunmuştur. RYBS'lerin yerleşim planı bu koordinatlara göre yapıldığı takdirde bu konuşlandırma için CEP değeri 350.03 m çıkmaktadır. İlk çalıştırmada çıkan CEP değeri ise yaklaşık olarak 362m'dir.

5.4 Senaryo 4

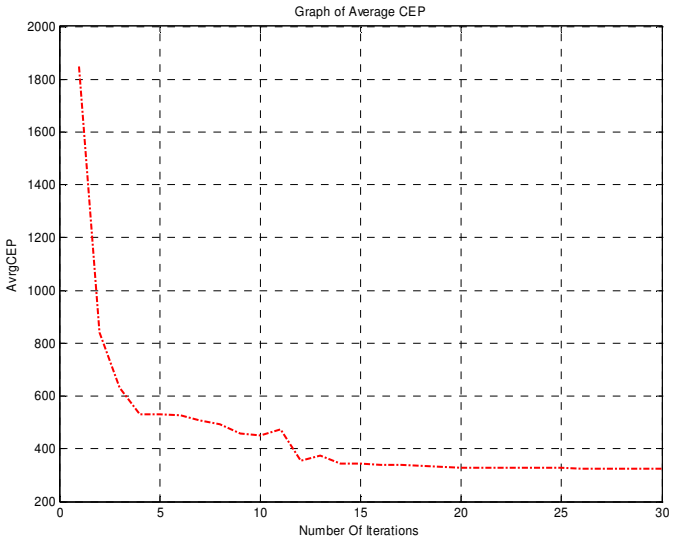
Dördüncü senaryoda RYBS bölgesi dikey dikdörtgen, hedef bölgesi de kare biçiminde tanımlanmıştır. Hedef bölgesi 5x5'lik çözünürlükte 25 parçaya ayrılmıştır. Her bir parça da dikdörtgen şeklinde olup her birinin ortasında elektromanyetik yayın yapan bir hedef olduğu varsayılmaktadır. Senaryoda RYB bölgesine iki adet radyo yön bulma sistemi yerleştirilmesine karar verilmiştir. Her iki sistemin de kerteriz açısı ölçüm hata standart sapma değerleri bir önceki senaryo da olduğu gibi eşit olarak alınmış ve 3^{o} olarak belirlenmiştir.

Genetik algoritma parametreleri 300 çözüm adayı ile çözüm uzayında en uygun değerlere sahip adayları belirlemek üzere 30 iterasyon yapacak şekilde ayarlanmıştır. En uygun adayların %60'u nesiller boyu korunmaya alınarak, en uygun bireyler için

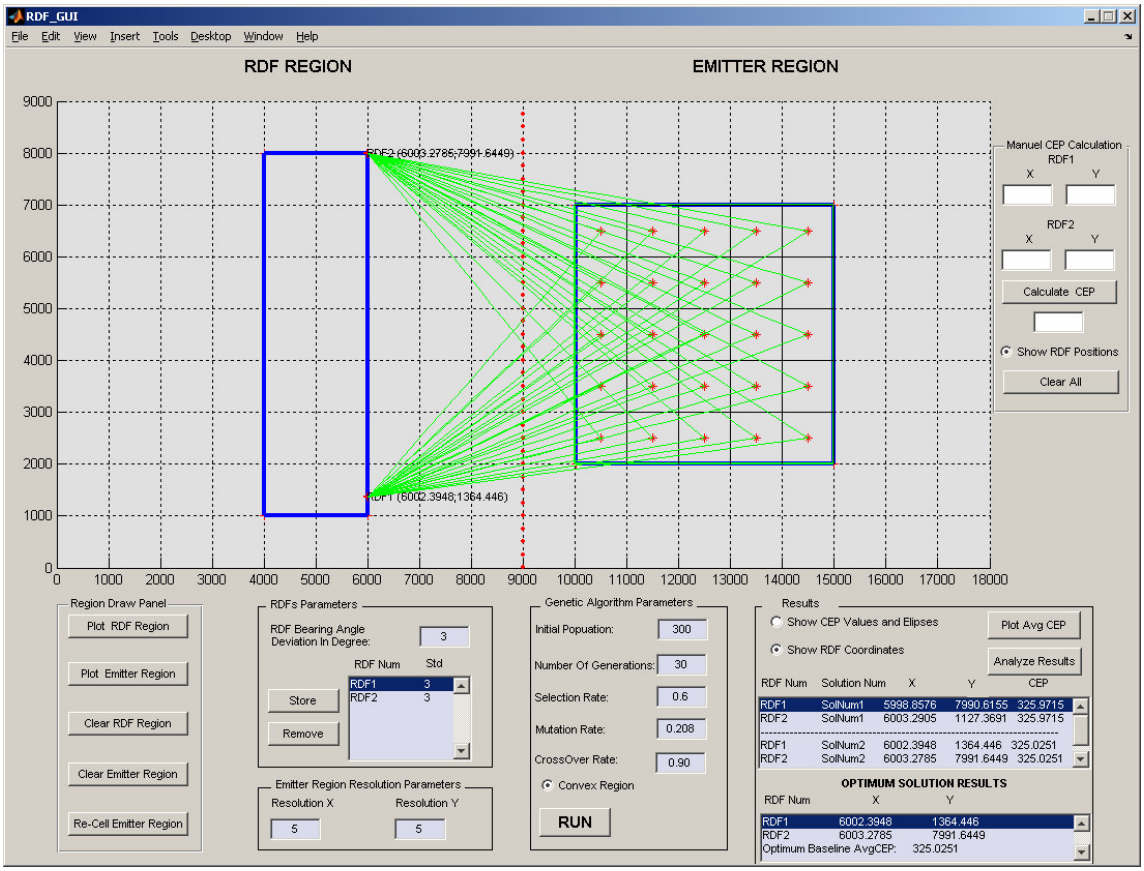
çözüm arayışında mutasyon oranı 0.208 çaprazlama oranı da %90 olarak belirlenmiştir. Bu senaryoda belirlenen RYBS alanı konveks olmadığı için *Convex Region* butonu 'off' durumuna getirilerek aritmetik çaprazlama yapılması sağlanmıştır. Senaryo iki kez yeni baştan çalıştırılarak her iki çalıştırmada da elde edilen değerler *Results* panelinde kullanıcıya verilmiştir. Bunlara ait şekiller aşağıda sırasıyla verilmektedir :



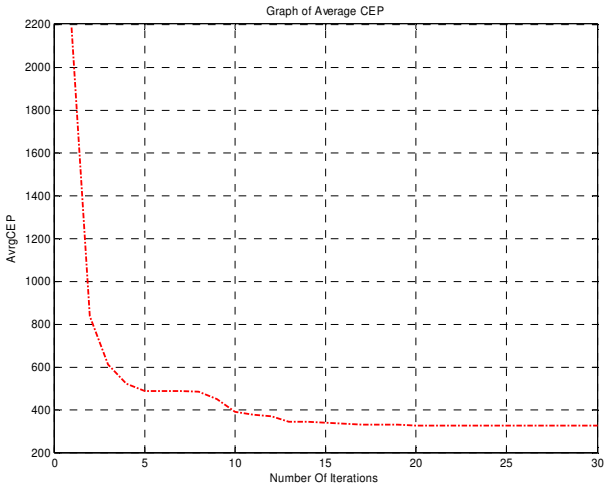
Şekil 5.23 Senaryo-4, I. Çalıştırma



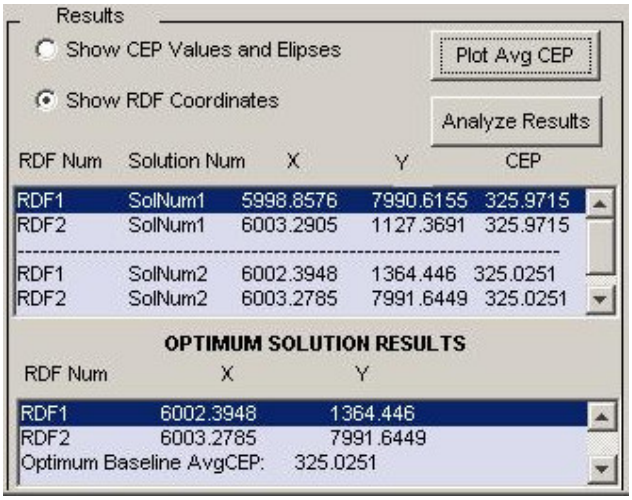
Şekil 5.24 Senaryo-4, I. Çalıştırma En İyi Ortalama CEP Değerleri



Şekil 5.25 Senaryo-4, II. Çalıştırma



Şekil 5.26 Senaryo-4, II. Çalıştırma En İyi Ortalama CEP Değerleri



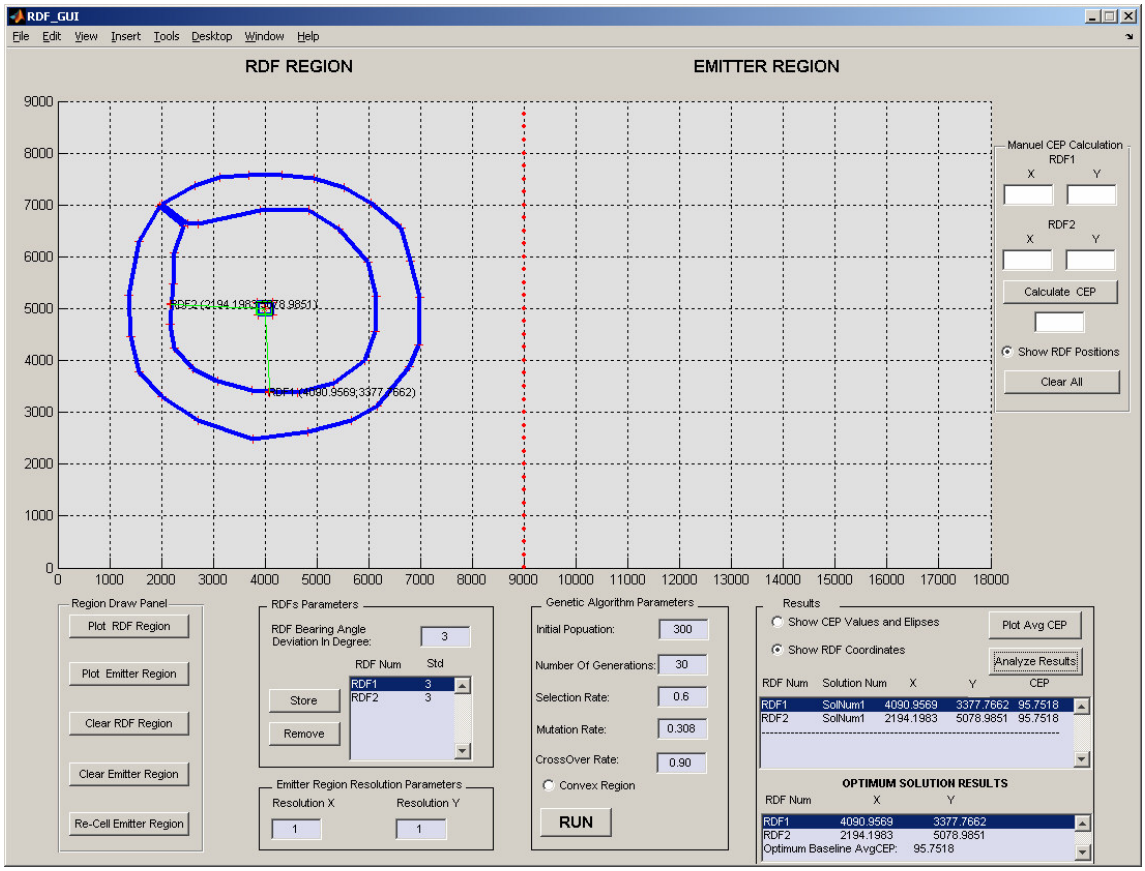
Şekil 5.27 Senaryo-4, Optimum Çözüm Sonuçları

Bu sonuçlara göre en iyi çözüm ortalamada en iyi CEP değerine sahip olan ikinci çalıştırmada elde edilen çözümdür. Burada RDF1'in koordinatları (6002.3948; 1364.446) ve RDF2'nin koordinatları (6003.2785; 7991.6449) olarak bulunmuştur. RYBS'lerin yerleşim planı bu koordinatlara göre yapıldığı takdirde bu konuşlandırma için CEP değeri yaklaşık olarak 325.03m çıkmaktadır. İlk çalıştırmada elde edilen CEP değeri ise yaklaşık olarak 325.97m'dir.

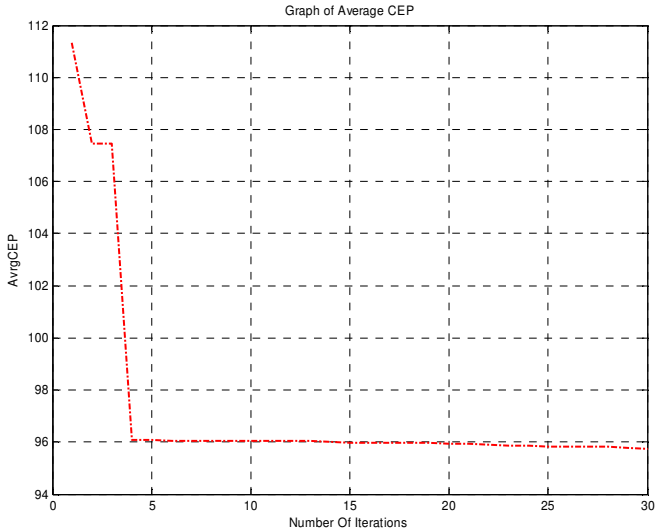
5.5 Senaryo 5

Beşinci senaryoda RYBS bölgesi halka, hedef bölgesi de halkanın içinde boyutları oldukça küçük bir dikdörtgen biçiminde tanımlanmıştır. Senaryoda hedef bölgesi bir nokta halinde düşünüldüğü için ebatları çok küçük belirlendi. Hedef bölgesi 1x1'lik çözünürlüktedir. Hedef bölgesinin ortasında elektromanyetik yayın yapan tek bir hedef olduğu varsayılmaktadır. Senaryoda RYB bölgesine ilk çalıştırmalarda iki adet radyo yön bulma sistemi yerleştirilmesine karar verilmiştir. Daha sonrada üç adet RYBS yerleştirilecek şekilde senaryo birkaç kez çalıştırıldı. Her iki/üç sistemin de kerteriz açısı ölçüm hata standart sapma değerleri bir önceki senaryo da olduğu gibi eşit olarak alınmış ve 3^{o} olarak belirlenmiştir.

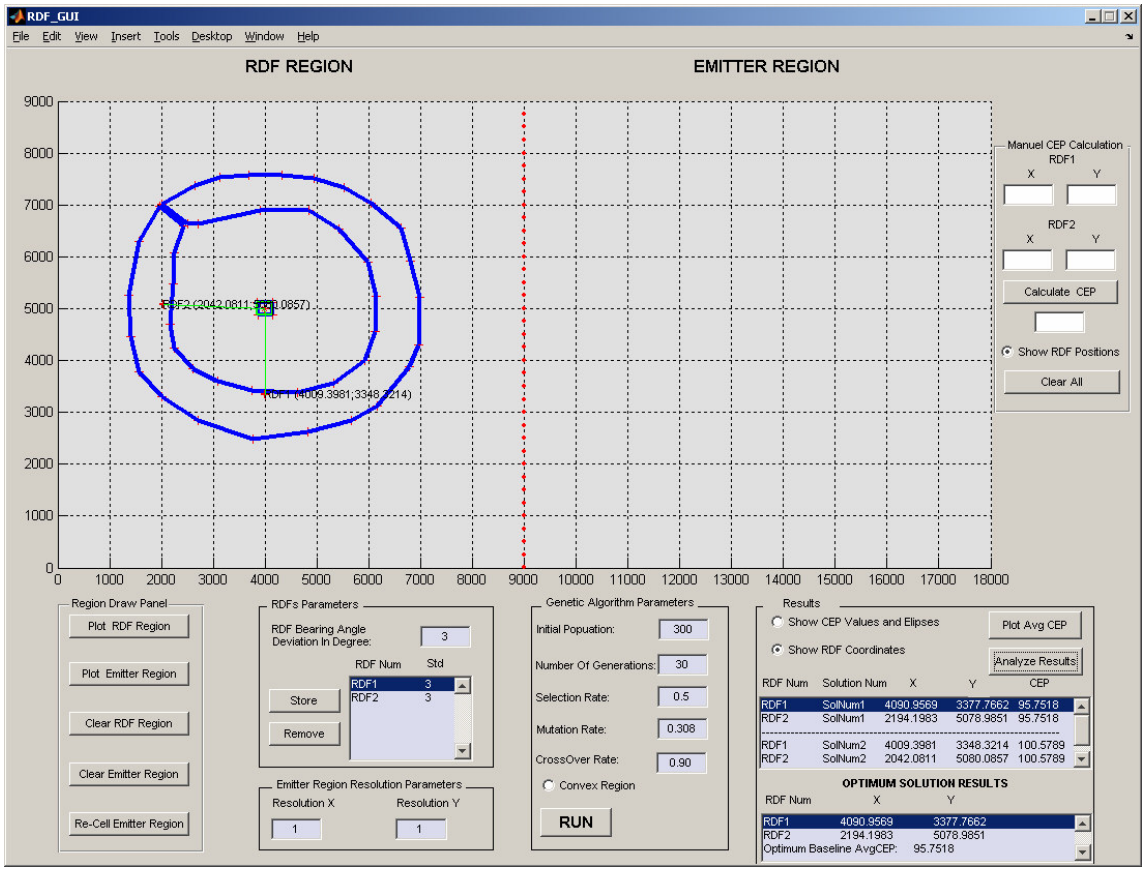
Genetik algoritma parametreleri 300 çözüm adayı ile çözüm uzayında en uygun değerlere sahip adayları belirlemek üzere 30 iterasyon yapacak şekilde ayarlanmıştır. En uygun adayların %60'ı nesiller boyu korunmaya alınarak, en uygun bireyler için çözüm arayışında farklı mutasyon oran ve çaprazlama oranları denenerek sırasıyla 0.308 mutasyon oranı, 0.90'de çaprazlama oranı olarak belirlenmiştir. Bu senaryoda belirlenen RYBS alanı da konveks olmadığı için *Convex Region* butonu 'off' durumuna getirilerek tek-noktalı çaprazlama yapılması sağlanmıştır. Senaryo iki kez yeni baştan çalıştırılarak her iki çalıştırmada da elde edilen değerler *Results* panelinde kullanıcıya verilmiştir. Bunlara ait şekiller aşağıda sırasıyla verilmektedir :



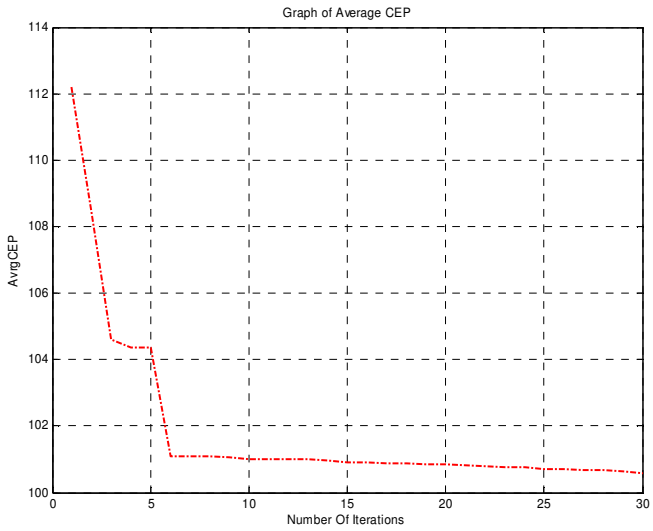
Şekil 5.28 Senaryo-5, I. Çalıştırma



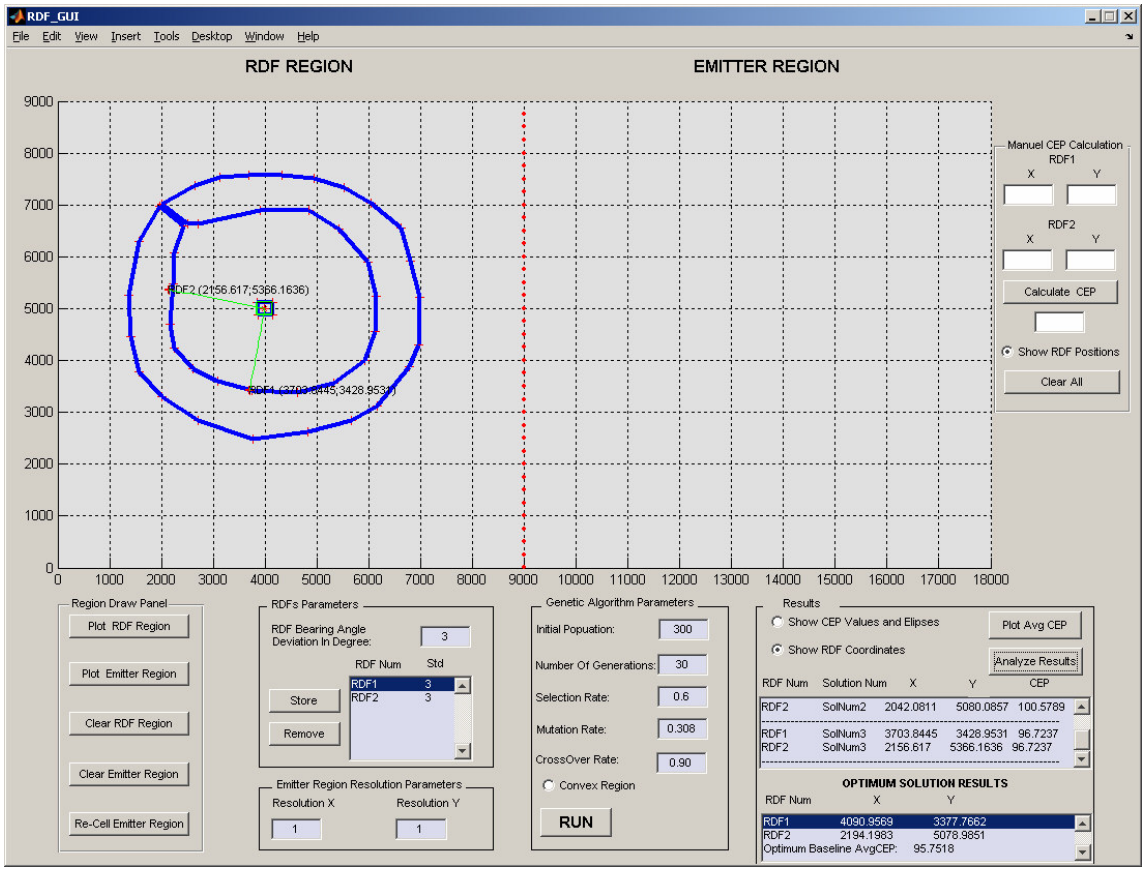
Şekil 5.29 Senaryo-5, I. Çalıştırma En İyi Ortalama CEP Değerleri



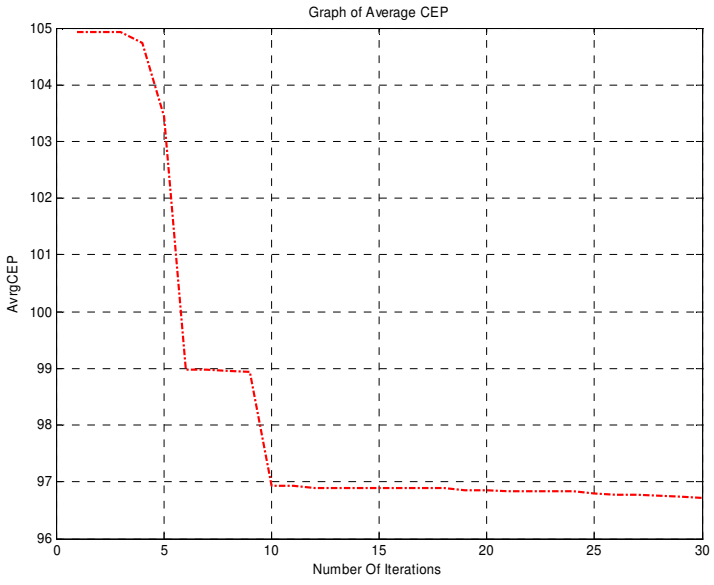
Şekil 5.30 Senaryo-5, II. Çalıştırma



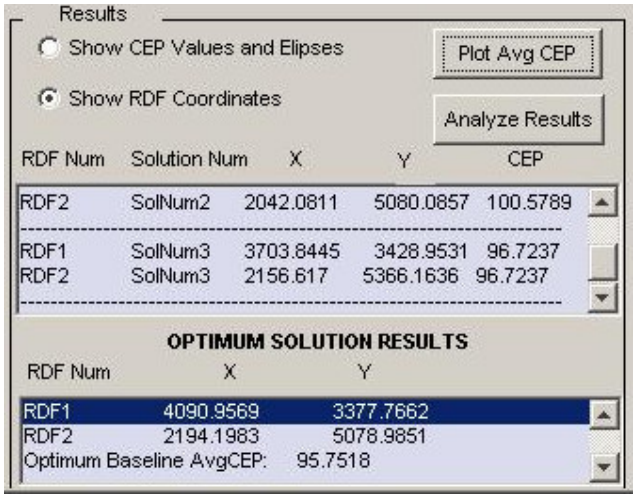
Şekil 5.31 Senaryo-5, II. Çalıştırma En İyi Ortalama CEP Değerleri



Şekil 5.32 Senaryo-5, III. Çalıştırma



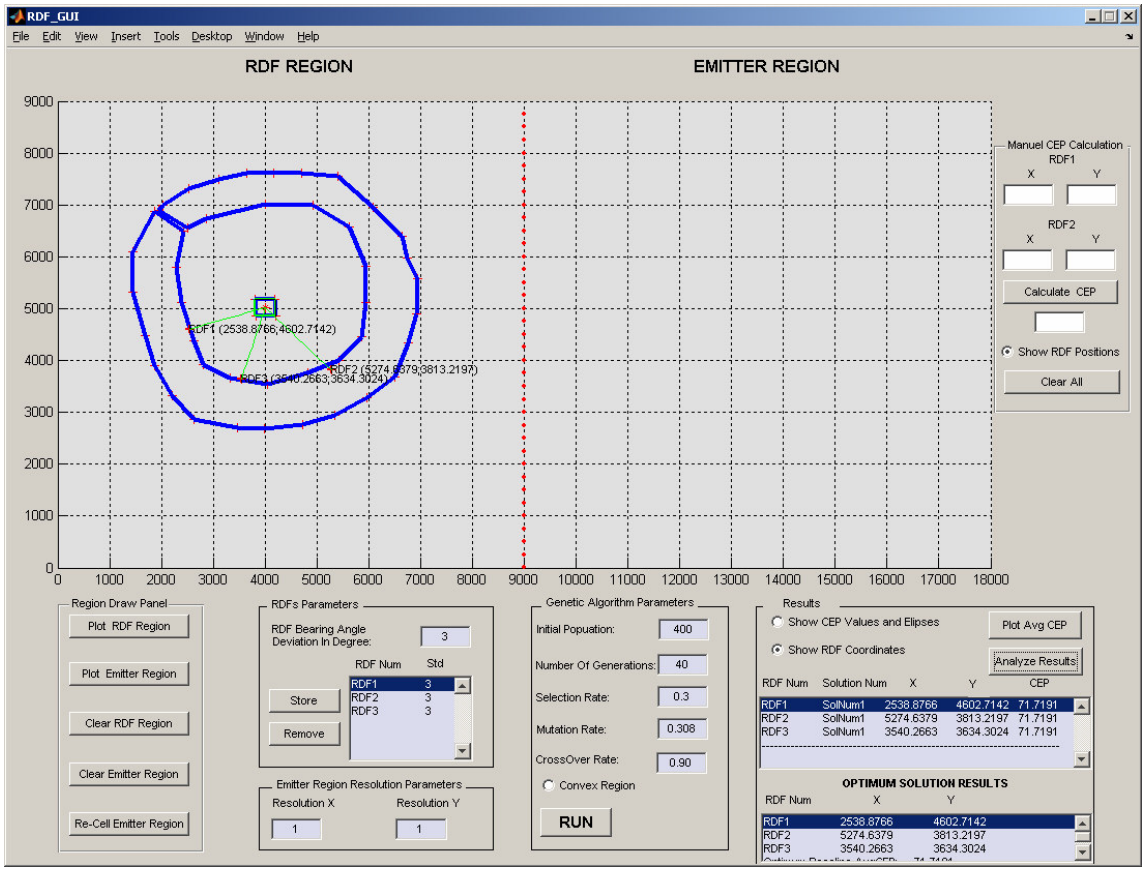
Şekil 5.33 Senaryo-5, III. Çalıştırma En İyi Ortalama CEP Değerleri



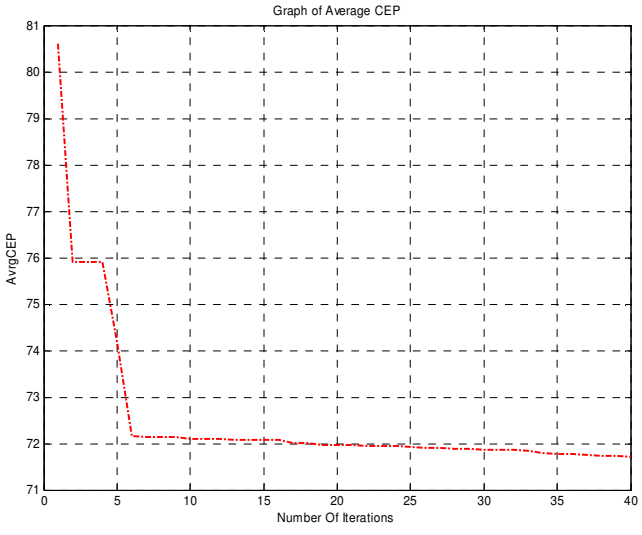
Şekil 5.34 Senaryo-5, Optimum Çözüm Sonuçları

Bu sonuçlara göre iki RYBS kullanıldığında en iyi çözüm ortalamada en iyi CEP değerine sahip olan birinci çalıştırmada elde edilen çözümdür. Burada RDF1'in koordinatları (4009.9569; 3377.7662) ve RDF2'nin koordinatları (2194.1983; 5078.9851) olarak bulunmuştur. RYBS'lerin yerleşim planı bu koordinatlara göre yapıldığı takdirde bu konuşlandırma için CEP değeri yaklaşık olarak 95.75m çıkmaktadır. İkinci ve üçüncü çalıştırmalarda elde edilen CEP değerleri ise yaklaşık olarak sırasıyla 100.58m ve 96.72m'dir.

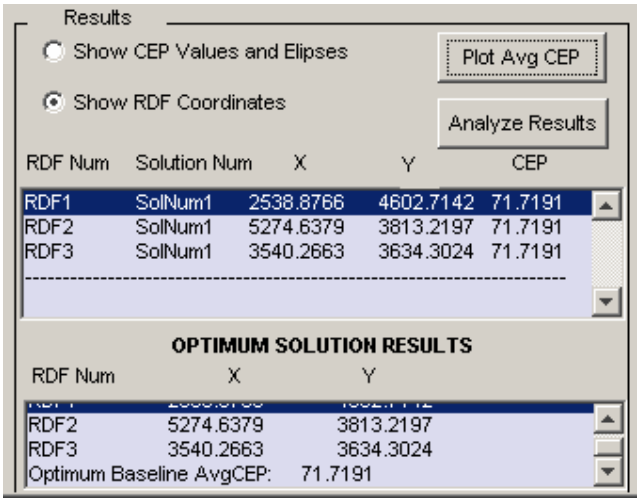
Benzer bir konfigürasyonda aynı ölçüm hassasiyetine sahip üç RYBS kullanıldığında ve genetik algoritma parametreleri Şekil 5.35'te görüldüğü gibi girildiğinde elde edilen sonuca ait şekiller de aşağıda verilmektedir.



Şekil 5.35 Senaryo-5-3 RYBS, I. Çalıştırma



Şekil 5.36 Senaryo-5-3 RYBS, I. Çalıştırma En İyi Ortalama CEP Değerleri



Şekil 5.37 Senaryo-5-3 RYBS, Optimum Çözüm Sonuçları

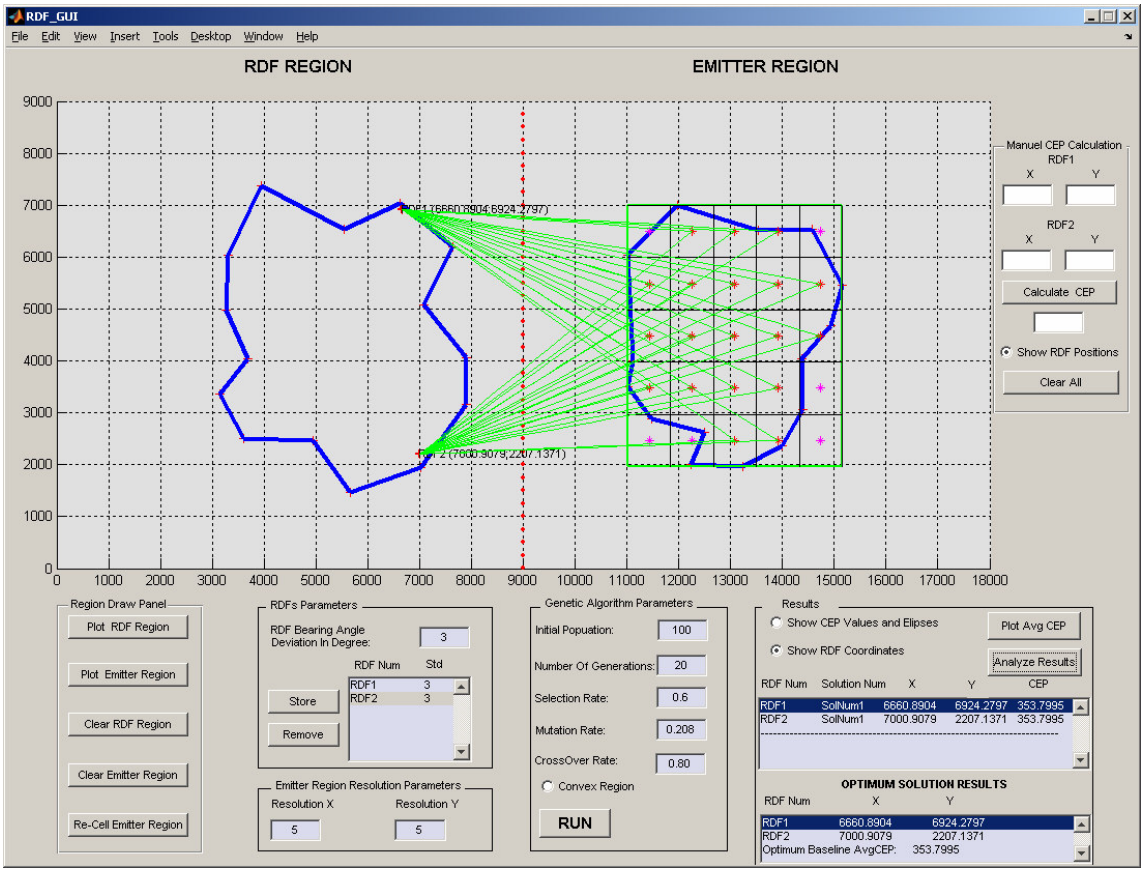
Bu sonuçlara göre RDF1'in koordinatları (2538.8766; 4602.7142), RDF2'nin koordinatları (5274.6379; 3813.2197) ve RDF3'ün koordinatları (3540.2663; 3634.3024) olarak bulunmuştur. RYBS'lerin yerleşim planı bu koordinatlara göre yapıldığı takdirde bu konuşlandırma için CEP değeri 71.72m çıkmaktadır.

5.6 Senaryo 6

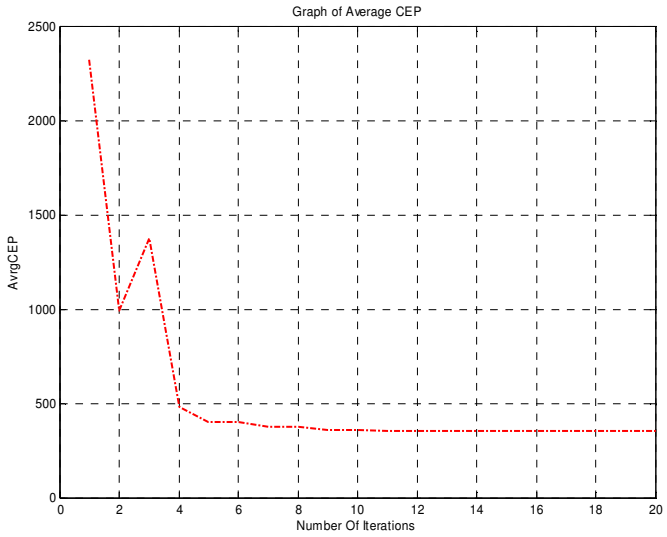
Altıncı senaryoda RYBS bölgesi de hedef bölgesi de tamamen rasgele biçimde tanımlandığında ne gibi sonuçlar elde edilebilir sorusuna cevap aranmıştır. Hedef bölgesi 5x5'lük çözünürlükte 25 parçaya ayrılmıştır. Her bir parça da dikdörtgen şeklinde olup her birinin ortasında elektromanyetik yayın yapan bir hedef olduğu varsayılmaktadır. Senaryoda RYB bölgesine iki adet radyo yön bulma sistemi yerleştirilmesine karar verilmiştir. Her iki sistemin de kerteriz açısı ölçüm hata standart sapma değerleri bir önceki senaryo da olduğu gibi eşit olarak alınmış ve 3^{o} olarak belirlenmiştir.

Genetik algoritma parametreleri 100 çözüm adayı ile çözüm uzayında en uygun değerlere sahip adayları belirlemek üzere 20 iterasyon yapacak şekilde ayarlanmıştır. En uygun adayların %60'u nesiller boyu korunmaya alınarak, en uygun bireyler için

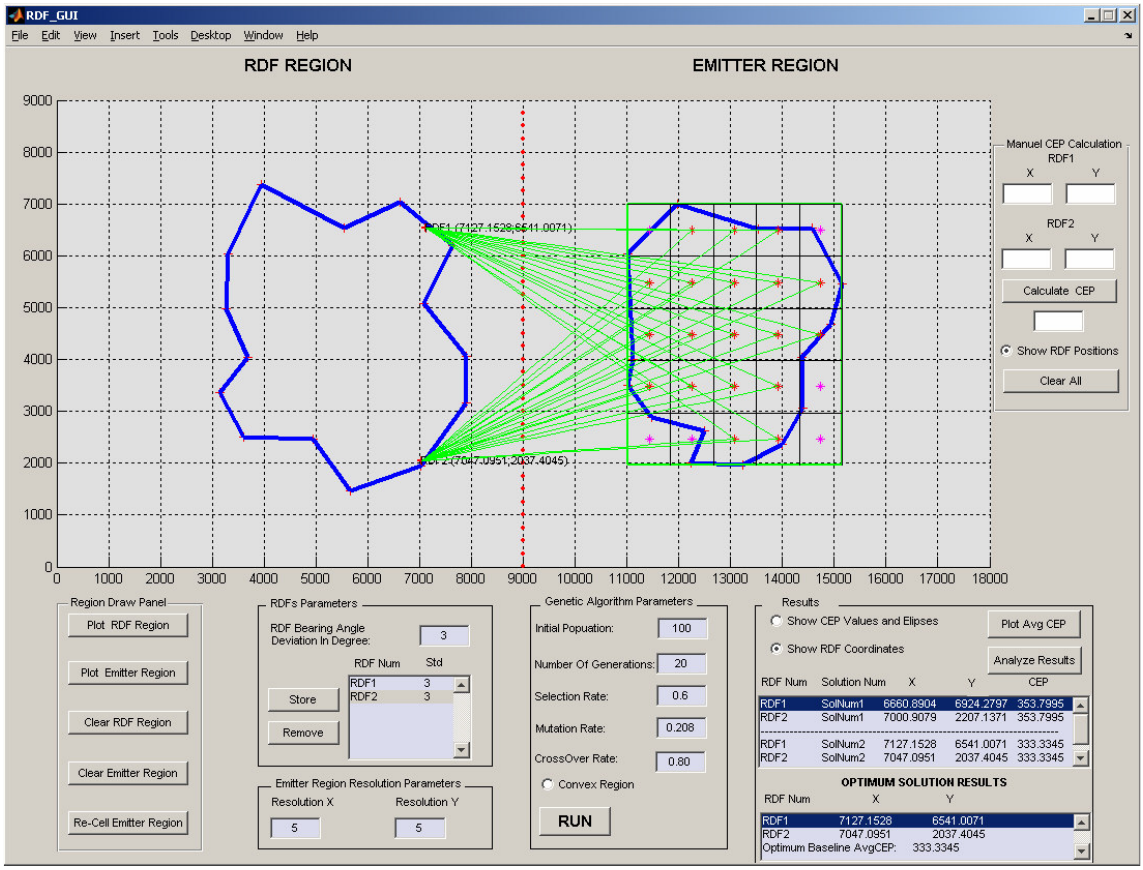
çözüm arayışında farklı mutasyon oran ve çaprazlama oranları denenerek sırasıyla 0.208 mutasyon oranı, 0.80'de çaprazlama oranı olarak belirlenmiştir. Bu senaryo da belirlenen RYBS alanı konveks olduğu için *Convex Region* butonu 'off' durumuna getirilerek tek-noktalı çaprazlama yapılması sağlanmıştır. Senaryo iki kez yeni baştan çalıştırılarak her iki çalıştırmada da elde edilen değerler *Results* panelinde kullanıcıya verilmiştir. Bunlara ait şekiller aşağıda sırasıyla verilmektedir :



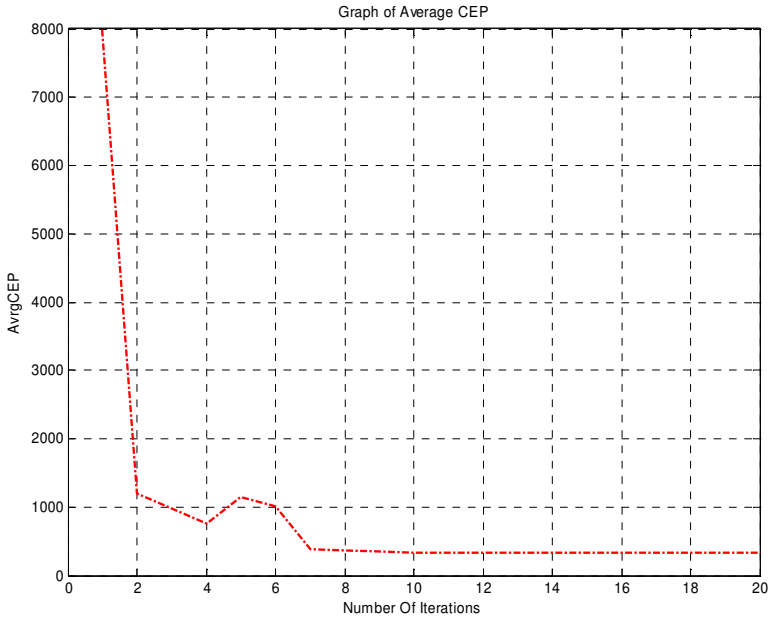
Şekil 5.38 Senaryo-6, I. Çalıştırma



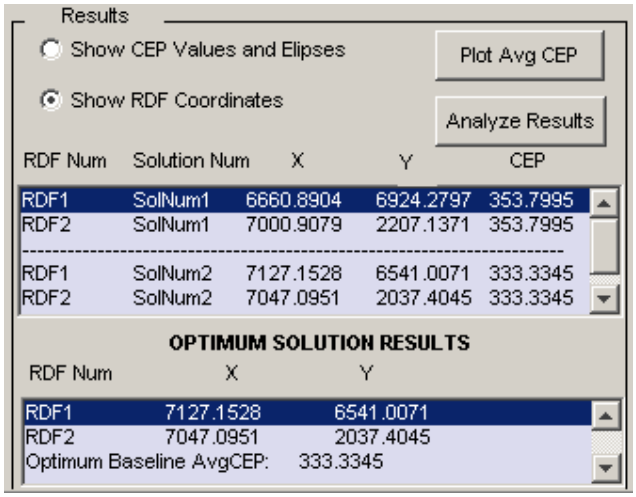
Şekil 5.39 Senaryo-6, I. Çalıştırma En İyi Ortalama CEP Değerleri



Şekil 5.40 Senaryo-6, II. Çalıştırma



Şekil 5.41 Senaryo-6, II. Çalıştırma En İyi Ortalama CEP Değerleri



Şekil 5.42 Senaryo-6, Optimum Çözüm Sonuçları

Bu sonuçlara göre en iyi çözüm ortalamada en iyi CEP değerine sahip olan ikinci çalıştırmada elde edilen çözümdür. Burada RDF1'in koordinatları (7127.1528; 6541.0071) ve RDF2'nin koordinatları (7047.0951; 2037.4045) olarak bulunmuştur. RYBS'lerin yerleşim planı bu koordinatlara göre yapıldığı takdirde bu konuşlandırma için CEP değeri yaklaşık olarak 333.34m çıkmaktadır. İlk çalıştırmada elde edilen

CEP değeri ise yaklaşık olarak 354m'dir. Bu senaryonun çalıştırılması ve kararlı sonuçlar elde edilmesi ile geliştirilen yazılımın herhangi bir biçimde tanımlanmış bölge haritaları için çalışabildiği kanıtlanmıştır.

6. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında yapılan etütler ve simülasyonlarla elde edilen sonuçlar şöyle sıralanabilir;

Genetik algoritmalar rasgele geçiş kuralları ile çalıştıklarından dolayı global optimuma yakınsamak için algoritmanın parametreleri deneme-yanılma yolu ile bulunmaya çalışılmalıdır. Dördüncü bölümde anlatılan esaslar göz önünde tutularak simülasyonlarda bu konuya dikkat edilmiştir. Yine algoritmanın doğasında yer alan rasgelelikten dolayı program birden fazla sayıda çalıştırılarak her bir çalışma sonrasında elde edilen sonuçları veri kabul edip istatiksel yöntemlerle bu veriler incelenmeli ve alt-optimum noktalardan en uygunu seçilmelidir. Simülasyonlarda bu konu da göz önünde bulundurularak bir senaryo en az iki defa çalıştırılmıştır. Çok sayıda çalıştırma sonucunda da simülasyon sonuçları arasında verilemese dahi algoritmanın belirli bir değere yakınsadığı hep gözlemlenmiştir. Bu da geliştirilen programda kullanılan tekniklerin kararlı yapıda olduklarını bize göstermektedir.

Her bir simülasyon sonucunda elde edilen grafiklerde de görüldüğü gibi RYB sistemleri birbirlerine göre genel olarak bir yerleşim planında simetrik ve hedefe en yakın koordinatlara yerleşme eğilimindedir; dolayısıyla oluşan kerteriz hatlarının birbirleri ile en az doğrusal bağımlı olma eğiliminde oldukları açıktır. Bu tip bir yerleşim planı en az CEP değerini verir ve hedef bölgesinin kapsanma kalitesini yükseltir. Mümkün olan en az hata ile hedeflerin yer bilgisinin bulunabilmesi daha önceki bölümlerde de belirtildiği gibi savaş ortamında çok kritik bir bilgidir.

İncelemelerde RYB sistemlerinin sayılarının artırılması ve her bir RYB sisteminin kerteriz açısı ölçüm hassasiyetinin yüksek olması teoride de anlatıldığı gibi CEP değerinin düşmesine neden olarak sonucun kalitesini artırmaktadır.

Radyo yön bulma teorisi, konum belirleme algoritmaları ve optimizasyon yöntemi olan GA'nın öğrenilip uygulanması ile geliştirilen bu yazılım pratikte de kullanılabilir. Geliştirilen yazılımın ana hatları değiştirilmeden iki boyutta gerçek Coğrafi Bilgi İşleme (CBİ) veri ve haritalarının kullanımına yönelik gerekli değişiklikler yapılarak da program çalıştırılabilir. Sonuç olarak program kullanıcının sahip olduğu RYB sistemlerinin optimal konuşlandırılmasının nasıl yapılabileceği sorusuna cevap

vermektedir. Ayrıca kullanıcı dilerse optimizasyon algoritmasını kullanmaksızın da elindeki herhangi iki RYB sistemini kendisi konuşlandırarak belirlediği konuşlandırmanın kalitesini ölçebilmektedir.

Bu çalışmada kullanıcıya yalnızca iki boyutta arazi tanımlama olanağı verilmektedir; ileride hem yer sistemleri ve hem de hava sistemlerinin bir arada olduğu, yükselti ve engebeler içeren haritaların kullanıldığı senaryolar için bu tezde verilen temel bilgiler ışığında çalışmalar yapılabilir.

KAYNAKLAR

Kitaplar:

[Adamy, 2004] Adamy, L. D., 2004, A Second Course in Electronic Warfare, Artech House

[Coley, 1999] Coley A. D., 1999, An Introduction to Genetic Algorithms For Scientists and Engineers, World Scientific Publishing

[Eiben, 2003] Eiben E. A., Smith E. J., 2003, Introduction to Evolutionary Computing, Springer

[Haupt, 2004] Haupt, L. R., Haupt, S. E., 2004, Practical Genetic Algorithms, Wiley Interscience

[Herndon, 1991] Herndon, H. J., 1991, Small Aperture Radio Direction Finding, Artech House

[Mitchell, 1999] Mitchell, M., 1999, An Introduction to Genetic Algorithms, MIT Press

[Poisel, 2002] Poisel, A. R., 2002, Introduction to Communication Electronic Warfare Systems, Artech House

[Poisel, 2005] Poisel, A. R., 2005, Electronic Warfare Target Location Methods, Artech House

Makaleler:

[Ahn, 2003] Ahn W. C., 2003, Elitism-Based Compact Genetic Algorithms, IEEE Transactions on Evolutionary Computation, Vol. 7, No.4, August 2003

[Emel, 2002] Emel G. G., Taşkın Ç., 2002, Genetik Algoritmalar ve Uygulama Alanları, Uludağ Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, Cilt XXI, Sayı 1, s.129-152, 2002

[İşçi, 2003] İşçi Ö., Korukoğlu S., 2003, Genetik Algoritma Yaklaşımı ve Yöneylem Araştırmasında Bir Uygulama, Yönetim ve Ekonomi, Yıl:2003, Cilt:10, Sayı:2

[Levanon, 2000] Levanon N., 2000, Lowest GDOP in 2-D Scenarios, IEE Proc. Radar, Sonar Navig. ,Vol.147. No.3, June 2000

[Paradowski, 1997] Paradowski R. L., 1997, Uncertainity Ellipses and Their Application to Interval Estimation of Emitter Position, IEEE Transactions Aerospace and Electronic Systems, vol. 33, no.1, January 1997

[Poirot, 1976] Poirot L. J., 1976, Navigation by Back Triangulation, IEEE Transactions Aerospace and Electronic Systems, vol. AES-12, no.2, March 1976

[Poirot, 1978] Poirot L. J., Arbid, G., 1978, Position Location: Triangulation versus Circulation, IEEE Transactions Aerospace and Electronic Systems, vol. AES-14, no.1, Jan. 1978

[Stansfield, 1947] Stansfield R. G., 1947, Statistical Theory of D.F Fixing, J. IEE., Vol.94, Part IIIa, 1947, pp. 762-770

[Torrieri, 1984] Torrieri J. D., 1984, Statistical Theory of Passive Location Systems, IEEE Transactions Aerospace and Electronic Systems, vol. AES-20, no.2, March 1984

[Wade, 1976] Wade H. F., 1976, Position Location Solutions by Taylor-Series Estimation, IEEE Transactions Aerospace and Electronic Systems, vol. AES-12, no.2, March 1976

Tezler:

[Chow, 1998] Chow L. T., 1998, Passive Emitter Location Using Digital Terrain Data, Carnegie Mellon University, Electrical and Computer Engineering Department, MSc Thesis

[Güler, 2004] Güler Y. N., 2004, Indoor Source Localization via Direction Finding Technique, Sabancı Üniversitesi, Elektrik Elektronik Mühendisliği Bölümü Yüksek Lisans Tezi

Ders Notları:

[US Army Electronics Course, 1999] US Army Electronics Course, 1999, Introduction to Radio Direction Finding

[Mahir, 1999] Mahir N., Azcan H., 1999, Analitik Geometri, Anadolu Üniversitesi Yayınları

[Özkazanç, 2004] Özkazanç, Y., 2004 ,Seyir ve Güdüm Ders Notları, Hacettepe Üniversitesi, Elektrik Elektronik Mühendisliği Bölümü

İnternet kaynakları:

[BAE Systems, 2006] BAE Systems, 2006, Study Into Spectral Efficient Radar Systems in L and S Bands

[RF Products, 1998] RF Products, 1998, WN-001 Web Note, “Questions & Answers: A User’s Guide To Radio Direction Finding Basics”, www.rdfproducts.com/wn001_apl_01.pdf

EKLER

EK- A RDF_GUI.m

```
function varargout = RDF_GUI(varargin)
% RDF_GUI M-file for RDF_GUI.fig
%      RDF_GUI, by itself, creates a new RDF_GUI or raises the existing
%      singleton*.
%      H = RDF_GUI returns the handle to a new RDF_GUI or the handle to
%      the existing singleton*.
%      RDF_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in RDF_GUI.M with the given input arguments.
%      RDF_GUI('Property','Value',...) creates a new RDF_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before RDF_GUI_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property
%      application
%      stop.  All inputs are passed to RDF_GUI_OpeningFcn via varargin.
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help RDF_GUI
% Last Modified by GUIDE v2.5 06-Apr-2007 05:06:43
% global  X_minimum Y_minimum X_maximum Y_maximum RecX_array RecY_array
% Begin initialization code - DO NOT EDIT
clc;
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
   'gui_Singleton',  gui_Singleton, ...
   'gui_OpeningFcn', @RDF_GUI_OpeningFcn, ...
   'gui_OutputFcn',  @RDF_GUI_OutputFcn, ...
```

```
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before RDF_GUI is made visible.
function RDF_GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to RDF_GUI (see VARARGIN)
% Choose default command line output for RDF_GUI
%extraGUIControls;
global RunCounter
handles.output = hObject;
handles.clearRDF = 0;
handles.clearEmit = 0;
handles.RDFPoly = 0;
handles.EmitterPoly = 0;
% Update handles structure
RunCounter =0;
save runGA_Counter RunCounter;
guidata(hObject, handles);
```

```
load CEPValues bestAvgCEP bestRDFPoints;
if (length(bestAvgCEP)>0) || (length(bestRDFPoints)>0)
   bestAvgCEP =[];
   bestRDFPoints=[];
   save CEPValues bestAvgCEP bestRDFPoints;
end
% UIWAIT makes RDF_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = RDF_GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
set(gca,'XGrid','on','YGrid','on');
% text(2000,10150,'RDF REGION','fontsize',13,'Color','r')
% text(6500,10150,'EMITTER REGION','fontsize',13,'Color','r')
northX1 = 5000:5000;
northY1 = 2:250:10000;
plot(northX1,northY1,'Marker','.','LineWidth',1,'Color','r');
% --- Executes on button press in polyrdfdraw.
function polyrdfdraw_Callback(hObject, eventdata, handles)
% hObject    handle to polyrdfdraw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Make a graphic object for the polygon
% global  X_minimum Y_minimum X_maximum Y_maximum X_array Y_array
if (handles.RDFPoly == 1)
```

```
    warndlg('You can not determine RDF Region Again, Delete the existent RDF Region First!','!! Warning !!');

    return;

end

gpt=line(0,0,'Marker','.','Erasemode','none');

x=[];

y=[];

disp('Left button for new point, right button to finish making polygon');

[x,y,button]=ginput(1);

set(gpt,'XData',x,'YData',y,'LineStyle','+','Color','r');drawnow

[xx,yy,button]=ginput(1);

while (button ~=3),

    x=[x;xx];

    y=[y;yy];

    set(gpt,'XData',x,'YData',y,'LineStyle','-','Color','r');drawnow

    [xx,yy,button]=ginput(1);

    w = x(:)+i*y(:);

end;

beta = scangle(w);

polyRDF = polygon(flipud(w),1-flipud(beta));

ployRDF_plot=plot(polyRDF)

handles.gpt=gpt;

handles.polyRDF=polyRDF;

handles.ployRDF_plot=ployRDF_plot;

handles.RDFPoly=1;

for j=1:length(real(w))

    X_array(j)=real(w(j));

    Y_array(j)=imag(w(j));

end

handles.RDFPolyX_array=X_array;
```

```
handles.RDFPolyY_array=Y_array;
if (handles.clearRDF)
   handles.clearRDF=1;
else
   handles.clearRDF=0;
end
if (handles.clearEmit)
   handles.clearEmit=1;
else
   handles.clearEmit=0;
end
guidata(hObject,handles);
% --- Executes on button press in polyemitterdraw.
function polyemitterdraw_Callback(hObject, eventdata, handles)
% hObject    handle to polyemitterdraw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (handles.EmitterPoly == 1)
   warndlg('You can not determine Emitter Region Again, Delete the existent Emitter Region First!','!! Warning !!');
   return;
end
global  X_minimum Y_minimum X_maximum Y_maximum X_array Y_array
X_array=0;
Y_array=0;
gptE=line(0,0,'Marker','.','Erasemode','none');
x=[];
y=[];
disp('Left button for new point, right button to finish making polygon');
[x,y,button]=ginput(1);
```

```
set(gptE,'XData',x,'YData',y,'LineStyle','+','Color','r');drawnow
[xx,yy,button]=ginput(1);
while (button ~=3),
   x=[x;xx];
   y=[y;yy];
   set(gptE,'XData',x,'YData',y,'LineStyle','-','Color','r');drawnow
   [xx,yy,button]=ginput(1);
   w = x(:)+i*y(:);
end;
beta = scangle(w);
polyEmit = polygon(flipud(w),1-flipud(beta));
ployEmit_plot=plot(polyEmit)
handles.gptE=gptE;
handles.polyEmit=polyEmit;
handles.ployEmit_plot=ployEmit_plot;
c1=length(get(gca,'children'));
for j=1:length(real(w))
   X_array(j)=real(w(j));
   Y_array(j)=imag(w(j));
end
X_minimum=min(X_array);
X_maximum=max(X_array);
Y_minimum=min(Y_array);
Y_maximum=max(Y_array);
Emit_SplitNumX =  str2num(get(handles.eSplitX,'string'));
Emit_SplitNumY =  str2num(get(handles.eSplitY,'string'));
set(gcf, 'Pointer', 'watch') ;pause(.10);
[handle_Frame_line handle_X_line handle_Y_line handle_Stars inPoints]
=drawLinesOfRec([X_minimum Y_minimum X_maximum Y_minimum ...
   X_maximum Y_maximum X_minimum
Y_maximum],Emit_SplitNumX,Emit_SplitNumY,X_array,Y_array);
```

```
set(gcf, 'Pointer', 'arrow');
handles.handle_Frame_line=handle_Frame_line;
handles.handle_X_line=handle_X_line;
handles.handle_Y_line=handle_Y_line;
handles.handle_Stars=handle_Stars;
handles.inPoints=inPoints;
% handles.handle_Stars_Out=handle_Stars_Out;
% c2=length(get(gca,'children'));
% newObjLenEm=c2-c1-4;
% handles.newObjLenEm=newObjLenEm;
handles.cellified=0;
handles.EmitterPoly=1;
if (handles.clearRDF)
   handles.clearRDF=1;
else
   handles.clearRDF=0;
end
if (handles.clearEmit)
   handles.clearEmit=1;
else
   handles.clearEmit=0;
end
%handles.clearEmit = 0;
guidata(hObject,handles);
% --- Executes on button press in clearRDFPoly.
function clearRDFPoly_Callback(hObject, eventdata, handles)
% hObject    handle to clearRDFPoly (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% global  X_minimum Y_minimum X_maximum Y_maximum RecX_array RecY_array
```

```
%----DELETE LIST BOXES-----
currentVal1 = get(handles.resultsLBox,'Value');
OptimalResultsStr = get(handles.resultsLBox,'String');
numResults1 = size(OptimalResultsStr,1);
currentVal2 = get(handles.resultsLBox,'Value');
OptimalResultsStr2 = get(handles.resultsOptimumBox,'String');
numResults2 = size(OptimalResultsStr2,1);
% Remove the data and list entry for the selected value
OptimalResultsStr =[];
handles.OptimalResultsStr=[];
OptimalResultsStr2 =[];
handles.OptimalResultsStr2=[];
% If there are no other entries, change the list sting to <empty>
if isequal(numResults1,length(currentVal1)),
   resultsStr1 = {''};
   currentVal1 = 1;
end
if isequal(numResults2,length(currentVal2)),
   resultsStr2 = {''};
   currentVal2 = 1;
end
% Ensure that list box Value is valid, then reset Value and String
currentVal1 = min(currentVal1,size(OptimalResultsStr,1));
set(handles.resultsLBox,'Value',currentVal1,'String',OptimalResultsStr)
currentVal2 = min(currentVal2,size(OptimalResultsStr2,1));
set(handles.resultsOptimumBox,'Value',currentVal2,'String',OptimalResultsStr2)
% Store the new ResultsData
guidata(hObject,handles)
%----END DELETE LIST BOXES-----
c =handles.polyRDF;
```

```
delete(handles.gpt);
delete(handles.ployRDF_plot);
if (isfield(handles,'rand_plot_handle') && ~isempty(handles.rand_plot_handle)) ...
        &&(isfield(handles,'lob_plot_handle') && ~isempty(handles.lob_plot_handle)) ...
        &&(isfield(handles,'ellipse_plot_handle') && ~isempty(handles.ellipse_plot_handle))...
        &&(isfield(handles,'cand_plot_rdf_handle') && ~isempty(handles.cand_plot_rdf_handle))
    delete(handles.rand_plot_handle);
    delete(handles.cand_plot_rdf_handle);
    numOfInPoints = length(handles.inPoints);
    initialPop = str2num(get(handles.initialPop,'string'));
    numberOfRDF = length(handles.ResultsData);
    for k=1:numOfInPoints
        for j = 1:numberOfRDF
            delete(handles.lob_plot_handle(k,j));
        end
        if (handles.clearEmit ==0)&&(handles.ellipse_plot_handle(k,j)~=0)
            delete(handles.ellipse_plot_handle(k,j));
            if (handles.showCEPVal==1)
                delete(handles.cepText_handle(k,j));
            end
        end
    end
    handles.ellipse_plot_handle=[];
    if (handles.showCEPVal==1)
        handles.cepText_handle = [];
    end
    handles.lob_plot_handle=[];
    handles.clearRDF =1;
    handles.RDFPoly =0;
    guidata(hObject,handles);
```

```
else
   handles.RDFPoly =0;
end
%handles.RDFPoly =1
handles.clearRDF = 1;
guidata(hObject,handles);
% --- Executes on button press in clearEmitPoly.
function clearEmitPoly_Callback(hObject, eventdata, handles)
% hObject    handle to clearEmitPoly (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%----DELETE LIST BOXES-----
currentVal1 = get(handles.resultsLBox,'Value');
OptimalResultsStr = get(handles.resultsLBox,'String');
numResults1 = size(OptimalResultsStr,1);
currentVal2 = get(handles.resultsLBox,'Value');
OptimalResultsStr2 = get(handles.resultsOptimumBox,'String');
numResults2 = size(OptimalResultsStr2,1);
% Remove the data and list entry for the selected value
OptimalResultsStr =[];
handles.OptimalResultsStr=[];
OptimalResultsStr2 =[];
handles.OptimalResultsStr2=[];
% If there are no other entries, change the list sting to <empty>
if isequal(numResults1,length(currentVal1)),
   resultsStr1 = {''};
   currentVal1 = 1;
end
if isequal(numResults2,length(currentVal2)),
   resultsStr2 = {''};
```

```
    currentVal2 = 1;
end
% Ensure that list box Value is valid, then reset Value and String
currentVal1 = min(currentVal1,size(OptimalResultsStr,1));
set(handles.resultsLBox,'Value',currentVal1,'String',OptimalResultsStr)
currentVal2 = min(currentVal2,size(OptimalResultsStr2,1));
set(handles.resultsOptimumBox,'Value',currentVal2,'String',OptimalResultsStr2)
% Store the new ResultsData
guidata(hObject,handles)
%----END DELETE LIST BOXES-----
handles.EmitterPoly =0;
delete(handles.gptE);
delete(handles.ployEmit_plot);
delete(handles.handle_Frame_line);
if (handles.cellified==0)
    delete(handles.handle_X_line);
    delete(handles.handle_Y_line);
    delete(handles.handle_Stars);
    if (handles.runGA==1)
        numOfInPoints = length(handles.inPoints);
        initialPop = str2num(get(handles.initialPop,'string'));
        numberOfRDF = length(handles.ResultsData);
        for k=1:numOfInPoints
            if (handles.clearRDF ==0) && ~isempty(handles.cand_plot_rdf_handle)
                for j = 1:numberOfRDF
                    delete(handles.lob_plot_handle(k,j));
                    if (k<2)
                        delete(handles.cand_plot_rdf_handle(j));
                        delete(handles.rand_plot_handle(j));
                    end
```

```
                if (handles.clearRDF==0)&&(handles.ellipse_plot_handle(k,j)~=0)
                    delete(handles.ellipse_plot_handle(k,j));
                    if (handles.showCEPVal==1)
                        delete(handles.cepText_handle(k,j));
                    end
                end
            end
        end
        handles.ellipse_plot_handle=[];
        if (handles.showCEPVal==1)
            handles.cepText_handle=[];
        end
    end
    handles.cand_plot_rdf_handle=[];
    handles.lob_plot_handle=[];
    handles.rand_plot_handle=[];
    handles.runGA = 0;
    handles.clearEmit = 1;
    handles.EmitterPoly =0;
    guidata(hObject,handles);
else if(handles.cellified==1)
        delete(handles.handle_X_Cell_line);
        delete(handles.handle_Y_Cell_line);
        delete(handles.handle_Stars_Cell);
        handles.cellified=0;
        if (handles.runGA==1)
            numOfInPoints = length(handles.inPoints);
            initialPop = str2num(get(handles.initialPop,'string'));
            numberOfRDF = length(handles.ResultsData);
```

```
            for k=1:numOfInPoints
                for j = 1:numberOfRDF
                    if (handles.ellipse_plot_handle(k,j)~=0)
                        delete(handles.ellipse_plot_handle(k,j));
                        if (handles.showCEPVal==1)
                            delete(handles.cepText_handle(k,j));
                        end
                    end
                end
            end
            handles.ellipse_plot_handle = [];
            if (handles.showCEPVal==1)
                handles.cepText_handle = [];
            end
        end
        handles.cand_plot_rdf_handle=[];
        handles.lob_plot_handle=[];
        handles.rand_plot_handle=[];
        handles.runGA = 0;
        handles.clearEmit = 1;
        handles.EmitterPoly =0;
        guidata(hObject,handles);
    end
end
% --- Executes on button press in cellRDFPoly.
function cellRDFPoly_Callback(hObject, eventdata, handles)
% hObject    handle to cellRDFPoly (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% global X_minimum Y_minimum X_maximum Y_maximum X_array Y_array
```

```
% global  X_minimum Y_minimum X_maximum Y_maximum X_array Y_array
% --- Executes on button press in cellEmitterPoly.
function cellEmitterPoly_Callback(hObject, eventdata, handles)
% hObject    handle to cellEmitterPoly (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.handle_X_line);
delete(handles.handle_Y_line);
delete(handles.handle_Stars);
Emit_SplitNumX =  str2num(get(handles.eSplitX,'string'));
Emit_SplitNumY =  str2num(get(handles.eSplitY,'string'));
 [handle_X_Cell_line handle_Y_Cell_line handle_Stars_Cell inPoints]=...
    cellifyRegion(Emit_SplitNumX,Emit_SplitNumY)
handles.handle_X_Cell_line=handle_X_Cell_line;
handles.handle_Y_Cell_line=handle_Y_Cell_line;
handles.handle_Stars_Cell=handle_Stars_Cell;
handles.cellified=1;
handles.inPoints=inPoints;
guidata(hObject,handles);
% --- Executes on button press in storeRDFDev.
function varargout = storeRDFDev_Callback(hObject, eventdata, handles)
% hObject    handle to storeRDFDev (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Retrieve old results data structure
if isfield(handles,'ResultsData') & ~isempty(handles.ResultsData)
    ResultsData = handles.ResultsData;
    % Determine the maximum run number currently used.
    maxNum = ResultsData(length(ResultsData)).RunNumber;
    ResultNum = maxNum+1;
```

```
else, % Set up the results data structure
   ResultsData = struct('RDF',[],'RunNumber',[],'Std',[]);
   ResultNum = 1;
end
if isequal(ResultNum,1) || (ResultNum>1)
   %--Enable the Remove button
   set(handles.removeListitem,'Enable','on')
end
% Get std values to store with the data and put in the results list.
std = str2num(get(handles.rdfAngleDev,'String'));
ResultsData(ResultNum).RunName = ['RDF',num2str(ResultNum)];
ResultsData(ResultNum).RunNumber = ResultNum;
ResultsData(ResultNum).stdValue = std;
% Build the new results list string for the listbox
ResultsStr = get(handles.angleDevListBox,'String');
if isequal(ResultNum,1)
   ResultsStr = {['RDF1               ',num2str(std)]};
else
   ResultsStr = [ResultsStr; {['RDF',num2str(ResultNum),'               ',num2str(std)]}];
end
set(handles.angleDevListBox,'String',ResultsStr);
% Store the new ResultsData
handles.ResultsData = ResultsData;
guidata(hObject,handles);
% --- Executes on button press in runGA.
function runGA_Callback(hObject, eventdata, handles)
% hObject    handle to runGA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%numberOfRDF = length(handles.ResultsData);
```

```
global RunCounter
persistent bestAvgCEP
persistent bestRDFPoints
timeStart = cputime;
%----DELETE LIST BOXES-----
currentVal1 = get(handles.resultsLBox,'Value');
OptimalResultsStr = get(handles.resultsLBox,'String');
numResults1 = size(OptimalResultsStr,1);
currentVal2 = get(handles.resultsLBox,'Value');
OptimalResultsStr2 = get(handles.resultsOptimumBox,'String');
numResults2 = size(OptimalResultsStr2,1);
 % Remove the data and list entry for the selected value
OptimalResultsStr =[];
handles.OptimalResultsStr=[];
OptimalResultsStr2 =[];
handles.OptimalResultsStr2=[];
% If there are no other entries, change the list sting to <empty>
if isequal(numResults1,length(currentVal1)),
   resultsStr1 = {''};
   currentVal1 = 1;
end
if isequal(numResults2,length(currentVal2)),
   resultsStr2 = {''};
   currentVal2 = 1;
end
% Ensure that list box Value is valid, then reset Value and String
currentVal1 = min(currentVal1,size(OptimalResultsStr,1));
set(handles.resultsLBox,'Value',currentVal1,'String',OptimalResultsStr)
currentVal2 = min(currentVal2,size(OptimalResultsStr2,1));
set(handles.resultsOptimumBox,'Value',currentVal2,'String',OptimalResultsStr2)
```

```
% Store the new ResultsData
guidata(hObject,handles)
%----END DELETE LIST BOXES-----
if (handles.runGA == 1)
   if (handles.PreviousChromLength~=length(handles.ResultsData))
      load CEPValues bestAvgCEP bestRDFPoints;
      if (length(bestAvgCEP)>0) || (length(bestRDFPoints)>0)
         bestAvgCEP =[];
         bestRDFPoints=[];
         save CEPValues bestAvgCEP bestRDFPoints
         RunCounter = 0;
         save runGA_Counter RunCounter;
      end
   end
end
set(gcf, 'Pointer', 'watch') ;pause(.5);
if ((handles.clearRDF==1) || (handles.clearEmit==1))
   load runGA_Counter RunCounter;
   RunCounter = 0;
   save runGA_Counter RunCounter;
   bestAvgCEP =[];
   %bestAvgCEP(RunCounter) = min(handles.BestCEP);
   bestRDFPoints=[];
   save CEPValues bestAvgCEP bestRDFPoints
   %RunCounter = RunCounter+1;
elseif (handles.runGA == 1) &&(handles.PreviousChromLength~=length(handles.ResultsData))
      RunCounter = 0;
      load CEPValues bestAvgCEP bestRDFPoints;
      if (length(bestAvgCEP)>0) || (length(bestRDFPoints)>0)
          bestAvgCEP =[];
```

```
            bestRDFPoints=[];
            save CEPValues bestAvgCEP bestRDFPoints;
        end
        save runGA_Counter RunCounter;
elseif  (handles.runGA == 1)
        load runGA_Counter RunCounter;
        RunCounter = RunCounter+1;
        save runGA_Counter RunCounter;
end
if ((handles.clearRDF==1) && (handles.clearEmit==1)) &&...
        ((handles.RDFPoly == 0) || (handles.EmitterPoly == 0))
    warndlg('You did not determine the RDF and Emitter Regions!','!! ERROR !!');
    pause(.30);
    set(gcf, 'Pointer', 'arrow');
    return;
end
try
    numberOfRDF = length(handles.ResultsData);
catch
    errmsg = lasterr;
    if(strfind(errmsg, 'Reference to non-existent field'))
        warndlg('Please enter RDF Parameters first','!! Warning !!');
        numberOfRDF=0;
        set(gcf, 'Pointer', 'arrow');pause(.10);
        return;
    end
end
initialPop=str2num(get(handles.initialPop,'string'));
lengthOfChromosome = numberOfRDF;
numOfIteration=str2num(get(handles.numOfGen,'string'));
```

```
xoverRate =str2num(get(handles.xoverRate,'string'));
mutRate=str2num(get(handles.mutRate,'string'));
selectRate=str2num(get(handles.selectRate,'string'));
if (isfield(handles,'lob_plot_handle') && ~isempty(handles.lob_plot_handle)) ...
        &&(isfield(handles,'ellipse_plot_handle') && ~isempty(handles.ellipse_plot_handle))
    %if (ishandle(handles.lob_plot_handle))
    handles.cellifiedGA = handles.cellified;
    if (handles.cellifiedGA)
        numOfInPoints = handles.PreviousInPointsLength;
        handles.cellifiedGA =0;
    else
        numOfInPoints = length(handles.inPoints)
    end
    if (handles.PreviousInitPop~=initialPop)
        initialPop = handles.PreviousInitPop;
    else
        initialPop = initialPop;
    end
    if (handles.PreviousChromLength~=lengthOfChromosome)
        numberOfRDF = handles.PreviousChromLength;
    else
        numberOfRDF = lengthOfChromosome;
    end
    if (~handles.RDFPoly) || (handles.runGA)
        if (~handles.clearEmit) && (~handles.clearRDF)
            delete(handles.rand_plot_handle);
            delete(handles.cand_plot_rdf_handle);
            handles.cand_plot_rdf_handle=[];
            handles.rand_plot_handle=[];
            for k=1:numOfInPoints
```

```
            if (handles.clearRDF==0)
                for j = 1:numberOfRDF
                    delete(handles.lob_plot_handle(k,j));
                end
            end
            if (~handles.clearEmit)
                for j = 1:numberOfRDF
                    if (handles.ellipse_plot_handle(k,j)~=0)
                        delete(handles.ellipse_plot_handle(k,j));
                        if (handles.showCEPVal==1)
                            delete(handles.cepText_handle(k,j));
                        end
                    end
                end
            end
        end
        handles.lob_plot_handle = [];
        handles.ellipse_plot_handle = [];
        if (handles.showCEPVal==1)
            handles.cepText_handle = [];
        end

    end
  end
  guidata(hObject,handles);
end
pause(.10);
%handles.showCEPVal = get(handles.showCepVal,'Value');
handles.showCEPVal = 1;
handles.showRDFVal = 1;
```

```
isItConvex = get(handles.convexButton,'Value');

%[rand_plot_handle AvgCEP minCEP ellipse_plot_h lob_plot_handle
cand_plot_rdf_handle]=RDF_GA(initialPop,lengthOfChromosome,numOfIteration,xoverRate,mutRate,
selectRate,handles.RDFPolyX_array,handles.RDFPolyY_array,handles.inPoints,handles.ResultsData)

[Solution rand_plot_handle AvgCEP minCEP ellipse_plot_h cepText_h lob_plot_handle
cand_plot_rdf_handle bestCEPVal AvgCEPForBestRDFSet
RDFCoordinates]=RDF_GA(initialPop,lengthOfChromosome,numOfIteration,xoverRate,mutRate,selec
tRate,handles.RDFPolyX_array,handles.RDFPolyY_array,handles.inPoints,handles.ResultsData,handl
es.showCEPVal,handles.showRDFVal,isItConvex)

handles.rand_plot_handle = rand_plot_handle;

handles.cand_plot_rdf_handle = cand_plot_rdf_handle;

handles.RDFCoordinates = RDFCoordinates;

for k=1:length(handles.inPoints)

   for j = 1:lengthOfChromosome

      handles.lob_plot_handle(k,j) = lob_plot_handle(k,j);

      if (ellipse_plot_h(k,j)~=0)

         handles.ellipse_plot_handle(k,j) = ellipse_plot_h(k,j);

         if (handles.showCEPVal==1)

            handles.cepText_handle(k,j) = cepText_h(k,j);

         end

      end

   end

end

handles.PreviousInPointsLength = length(handles.inPoints);

handles.PreviousInitPop = initialPop;

handles.PreviousChromLength =  lengthOfChromosome;

handles.AvgCEP=AvgCEP;

handles.BestCEP=minCEP;

handles.runGA = 1;

handles.clearEmit = 0;

handles.clearRDF = 0;

 set(gcf, 'Pointer', 'arrow')

elapsed_time = cputime-timeStart;
```

```
disp ('RUNGA Function Elapsed Time: ')
elapsed_time
load runGA_Counter RunCounter;
bestAvgCEP(RunCounter+1) = min(handles.BestCEP);
bestRDFPoints{RunCounter+1} =  Solution;
save -append  CEPValues bestAvgCEP bestRDFPoints
%save CEPValues -append
%runGA_counter =runGA_counter+1;
guidata(hObject,handles);
% --- Executes on button press in plotAvgCEP.
function plotAvgCEP_Callback(hObject, eventdata, handles)
% hObject    handle to plotAvgCEP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
figure;
%CEP=handles.AvgCEP;
bestCEP = handles.BestCEP;
%plot(CEP,'LineWidth',2); hold on;
plot(bestCEP,'-.','Color','red','LineWidth',2);
% legend('Average CEP','Best CEP',2);
xlabel('Number Of Iterations');
ylabel('AvrgCEP');
title('Graph of Average CEP');
grid;
% --- Executes on button press in removeListitem.
function removeListitem_Callback(hObject, eventdata, handles)
% hObject    handle to removeListitem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Callback of the uicontrol handles.RemoveButton.
```

```
currentVal = get(handles.angleDevListBox,'Value');
resultsStr = get(handles.angleDevListBox,'String');
numResults = size(resultsStr,1);
% Remove the data and list entry for the selected value
resultsStr(currentVal) =[];
handles.ResultsData(currentVal)=[];
% If there are no other entries, disable the Remove and Plot button
% and change the list sting to <empty>
if isequal(numResults,length(currentVal)),
   resultsStr = {''};
   currentVal = 1;
   set(handles.removeListitem,'Enable','off')
end
% Ensure that list box Value is valid, then reset Value and String
currentVal = min(currentVal,size(resultsStr,1));
set(handles.angleDevListBox,'Value',currentVal,'String',resultsStr)
% Store the new ResultsData
guidata(hObject,handles)
function initialPop_Callback(hObject, eventdata, handles)
% hObject    handle to initialPop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of initialPop as text
%        str2double(get(hObject,'String')) returns contents of initialPop as a double
if (mod(str2double(get(hObject,'String')),2)~=0)
   %warndlg('Please enter Initial Population even ','!! Warning !!');
end
%--------------------------EXTRA GUI CONTROLS-----------------------
function rdfAngleDev_Callback(hObject, eventdata, handles)
% hObject    handle to rdfAngleDev (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of rdfAngleDev as text
%        str2double(get(hObject,'String')) returns contents of rdfAngleDev as a double
% --- Executes during object creation, after setting all properties.
function rdfAngleDev_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rdfAngleDev (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on selection change in angleDevListBox.
function angleDevListBox_Callback(hObject, eventdata, handles)
% hObject    handle to angleDevListBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns angleDevListBox contents as cell array
%        contents{get(hObject,'Value')} returns selected item from angleDevListBox
% --- Executes during object creation, after setting all properties.
function angleDevListBox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to angleDevListBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end
function numOfGen_Callback(hObject, eventdata, handles)
% hObject    handle to numOfGen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of numOfGen as text
%        str2double(get(hObject,'String')) returns contents of numOfGen as a double
% --- Executes during object creation, after setting all properties.
function numOfGen_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numOfGen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function selectRate_Callback(hObject, eventdata, handles)
% hObject    handle to selectRate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of selectRate as text
%        str2double(get(hObject,'String')) returns contents of selectRate as a double
% --- Executes during object creation, after setting all properties.
function selectRate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to selectRate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in calcAvgGdop.
function calcAvgGdop_Callback(hObject, eventdata, handles)
% hObject    handle to calcAvgGdop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
function xoverRate_Callback(hObject, eventdata, handles)
% hObject    handle to xoverRate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of xoverRate as text
```

```
%        str2double(get(hObject,'String')) returns contents of xoverRate as a double
% --- Executes during object creation, after setting all properties.
function xoverRate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xoverRate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function rdfSplitX_Callback(hObject, eventdata, handles)
% hObject    handle to rdfSplitX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rdfSplitX as text
%        str2double(get(hObject,'String')) returns contents of rdfSplitX as a double
% --- Executes during object creation, after setting all properties.
function rdfSplitX_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rdfSplitX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function rdfSplitY_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to rdfSplitY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of rdfSplitY as text
%        str2double(get(hObject,'String')) returns contents of rdfSplitY as a double
% --- Executes during object creation, after setting all properties.
function rdfSplitY_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rdfSplitY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function eSplitX_Callback(hObject, eventdata, handles)
% hObject    handle to eSplitX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of eSplitX as text
%        str2double(get(hObject,'String')) returns contents of eSplitX as a double
% --- Executes during object creation, after setting all properties.
function eSplitX_CreateFcn(hObject, eventdata, handles)
% hObject    handle to eSplitX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end
function eSplitY_Callback(hObject, eventdata, handles)
% hObject    handle to eSplitY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of eSplitY as text
%        str2double(get(hObject,'String')) returns contents of eSplitY as a double
% --- Executes during object creation, after setting all properties.
function eSplitY_CreateFcn(hObject, eventdata, handles)
% hObject    handle to eSplitY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes during object creation, after setting all properties.
function initialPop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to initialPop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in rdfCoords.
function rdfCoords_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to rdfCoords (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (get(hObject,'Value') == get(hObject,'Max'))
    % then checkbox is checked-take approriate action
    %handles.showCepVal = 1;
    %handles.showCepVal=str2num(get(handles.showCepVal,'string'));
    if (handles.runGA == 1)
        set(handles.cand_plot_rdf_handle,'Visible','on')
        if (handles.showRDFVal==1)
          set(handles.cand_plot_rdf_handle,'Visible','on')
        end
    end
else
    % checkbox is not checked-take approriate action
    % handles.showCepVal = 0;
   if (handles.runGA == 1)
      if (handles.showRDFVal==1)
        set(handles.cand_plot_rdf_handle,'Visible','off')
      end
   end
end
% --- Executes on button press in showCepVal.
function showCepVal_Callback(hObject, eventdata, handles)
% hObject    handle to showCepVal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of showCepVal
if (get(hObject,'Value') == get(hObject,'Max'))
```

```
    % then checkbox is checked-take approriate action
    %handles.showCepVal = 1;
    %handles.showCepVal=str2num(get(handles.showCepVal,'string'));
    if (handles.runGA == 1)
        set(handles.cepText_handle,'Visible','on')
        if (handles.showCEPVal==1)
          set(handles.cepText_handle,'Visible','on')
        end
    end
else
    % checkbox is not checked-take approriate action
    % handles.showCepVal = 0;
   if (handles.runGA == 1)
%    for k=1:length(handles.inPoints)
 %       for j = 1:numberOfRDF
            if (handles.showCEPVal==1)
                %handles.cepText_handle(k,j) = cepText_h(k,j);
                set(handles.cepText_handle,'Visible','off')
            end
  %      end
   % end
   end
end
% --- Executes on button press in showResults.
function showResults_Callback(hObject, eventdata, handles)
% hObject    handle to showResults (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%----DELETE LIST BOXES-----
```

```
currentVal1 = get(handles.resultsLBox,'Value');
OptimalResultsStr = get(handles.resultsLBox,'String');
numResults1 = size(OptimalResultsStr,1);
currentVal2 = get(handles.resultsLBox,'Value');
OptimalResultsStr2 = get(handles.resultsOptimumBox,'String');
numResults2 = size(OptimalResultsStr2,1);
% Remove the data and list entry for the selected value
OptimalResultsStr =[];
handles.OptimalResultsStr=[];
OptimalResultsStr2 =[];
handles.OptimalResultsStr2=[];
% If there are no other entries, change the list sting to <empty>
if isequal(numResults1,length(currentVal1)),
   resultsStr1 = {''};
   currentVal1 = 1;
end
if isequal(numResults2,length(currentVal2)),
   resultsStr2 = {''};
   currentVal2 = 1;
end
% Ensure that list box Value is valid, then reset Value and String
currentVal1 = min(currentVal1,size(OptimalResultsStr,1));
set(handles.resultsLBox,'Value',currentVal1,'String',OptimalResultsStr)
currentVal2 = min(currentVal2,size(OptimalResultsStr2,1));
set(handles.resultsOptimumBox,'Value',currentVal2,'String',OptimalResultsStr2)
 % Store the new ResultsData
guidata(hObject,handles)
%----END DELETE LIST BOXES-----
set(gcf, 'Pointer', 'watch') ;pause(.5);
load CEPValues bestAvgCEP bestRDFPoints;
```

```
for i=1:length(bestRDFPoints)
    for k =1:length(bestRDFPoints{1,1})
        for z=1:length(handles.inPoints)
            memberRDF(i,:)=bestRDFPoints(1,i);
            memberRDFArray= memberRDF{i,1};
            memberRDFArrayXY = memberRDFArray{1,k};
            memberRDFArrayX = memberRDFArrayXY(1,1);
            memberRDFArrayY = memberRDFArrayXY(1,2);
            %range calculation
            memberInPoints(z,:)=handles.inPoints{1,z};
            R(z,k,i) = sqrt((memberInPoints(z,1)-memberRDFArrayX)^2+(memberInPoints(z,2)-
memberRDFArrayY)^2);
        end

    end
end
MinRangeAvg = mean(mean(R));
 [minRange,InMinR] = min(MinRangeAvg)
 [ResultCEP, Index]= min(bestAvgCEP);
mu=mean(bestAvgCEP);
sigma=std(bestAvgCEP);
compare =sigma/mu;
if (compare >0.05)
    ResultRDFs = bestRDFPoints(Index);
else if (compare<0.05)
    ResultRDFs = bestRDFPoints(InMinR);
    end
end
% OptimalRDFsResult = struct('RDF',[],'RDFNumber',[],'X',[],'Y',[],'SolutionNum',[]);
showResult = ResultRDFs{1,1};
```

```
OptimalRDFsResult = struct('RDF',[],'RDFNumber',[],'SolutionNum',[],'X',[],'Y',[],'CEP',[]);
set(handles.resultsLBox,'Value',1);
for i=1:length(bestRDFPoints)
   for k =1:length(bestRDFPoints{1,1})
      memberRDF(i,:)=bestRDFPoints(1,i);
      memberRDFComp= memberRDF{i,1};
      memberRDFArrayXY= memberRDFComp{1,k};
      %memberRDFArrayXY = memberRDFArray(1,1);
      memberRDFArrayX = memberRDFArrayXY(1,1);
      memberRDFArrayY = memberRDFArrayXY(1,2);
      OptimalRDFsResult(i,k).RDFNumber = k;
      OptimalRDFsResult(i,k).SolNum = ['',num2str(i)];
      OptimalRDFsResult(i,k).Name = ['RDF',num2str(k)];
      OptimalRDFsResult(i,k).XVal = memberRDFArrayX;
      OptimalRDFsResult(i,k).YVal = memberRDFArrayY;
      OptimalRDFsResult(i,k).CepVal = bestAvgCEP(i);
      % Build the new results list string for the listbox
      OptimalResultsStr = get(handles.resultsLBox,'String');
      if isequal(k,1)&& isequal(i,1)
         OptimalResultsStr = {['RDF1          ','SolNum1      ' ...
         num2str(OptimalRDFsResult(k).XVal),'      ',num2str(OptimalRDFsResult(k).YVal),...
         '   ',num2str(OptimalRDFsResult(k).CepVal)]};
      else
         if (i==length(bestRDFPoints)-1)
            OptimalResultsStr = [OptimalResultsStr;{['RDF',num2str(k),'
','SolNum',num2str(OptimalRDFsResult(i).SolNum),'      ', ...
            num2str(OptimalRDFsResult(i,k).XVal),'      ',...
            num2str(OptimalRDFsResult(i,k).YVal),'   ',num2str(OptimalRDFsResult(i).CepVal)]}];
           %OptimalResultsStr = [OptimalResultsStr;{['Optimum Baseline AvgCEP:
',num2str(ResultCEP)]}];
         else
```

```
            OptimalResultsStr = [OptimalResultsStr;{['RDF',num2str(k),'
','SolNum',num2str(OptimalRDFsResult(i).SolNum),'      ', ...

            num2str(OptimalRDFsResult(i,k).XVal),'      ',...

            num2str(OptimalRDFsResult(i,k).YVal),'   ',num2str(OptimalRDFsResult(i).CepVal)]}];

          end

       end

       set(handles.resultsLBox,'String',OptimalResultsStr);

   end

   OptimalResultsStr = [OptimalResultsStr;{['-------------------------------------------------------------------------']}];

   set(handles.resultsLBox,'String',OptimalResultsStr);

end

OptimalRDFsResult2 = struct('RDF',[],'RDFNumber',[],'X',[],'Y',[],'CEP',[]);

set(handles.resultsOptimumBox,'Value',1);

%startCount = (length(bestRDFPoints)*length(bestRDFPoints{1,1})+length(bestRDFPoints))

for t=1 :length(showResult)

   OptimalRDFs(t,:)= showResult{1,t};

   %OptimalRDFsX_Val= OptimalRDFs(t-startCount+1,1);

   OptimalRDFsX_Val= OptimalRDFs(t,1);

   OptimalRDFsY_Val= OptimalRDFs(t,2);

   OptimalRDFsResult2(t).Name = ['RDF',num2str(t)];

   OptimalRDFsResult2(t).Number = t;

   OptimalRDFsResult2(t).XVal = OptimalRDFsX_Val;

   OptimalRDFsResult2(t).YVal = OptimalRDFsY_Val;

   % Build the new results list string for the listbox

   OptimalResultsStr2 = get(handles.resultsOptimumBox,'String');

   if isequal(t,1)

      OptimalResultsStr2 = {['RDF1              ',num2str(OptimalRDFsResult2(t).XVal),'
',num2str(OptimalRDFsResult2(t).YVal)]};

   else

      OptimalResultsStr2 = [OptimalResultsStr2;{['RDF',num2str(t),'
',num2str(OptimalRDFsResult2(t).XVal),'            ',num2str(OptimalRDFsResult2(t).YVal)]}];
```

```
    end

    if (t==length(showResult))

        OptimalResultsStr2 = [OptimalResultsStr2;{['Optimum Baseline AvgCEP:
',num2str(ResultCEP)]}];

    end

    set(handles.resultsOptimumBox,'String',OptimalResultsStr2);

end

set(gcf, 'Pointer', 'arrow');

handles.OptimalRDFsResult = OptimalRDFsResult;

guidata(hObject,handles);

% --- Executes on selection change in resultsLBox.

function resultsLBox_Callback(hObject, eventdata, handles)

% hObject    handle to resultsLBox (see GCBO)

% eventdata  reserved - to be defined in a future version of MATLAB

% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns resultsLBox contents as cell array

%        contents{get(hObject,'Value')} returns selected item from resultsLBox

% --- Executes during object creation, after setting all properties.

function resultsLBox_CreateFcn(hObject, eventdata, handles)

% hObject    handle to resultsLBox (see GCBO)

% eventdata  reserved - to be defined in a future version of MATLAB

% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.

%       See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

    set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in convexButton.

function convexButton_Callback(hObject, eventdata, handles)

% hObject    handle to convexButton (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of convexButton
% % --- Executes on button press in nonConvexButton.
% function nonConvexButton_Callback(hObject, eventdata, handles)
% % hObject    handle to nonConvexButton (see GCBO)
% % eventdata  reserved - to be defined in a future version of MATLAB
% % handles    structure with handles and user data (see GUIDATA)
% % Hint: get(hObject,'Value') returns toggle state of nonConvexButton
% --- Executes on selection change in resultsOptimumBox.
function resultsOptimumBox_Callback(hObject, eventdata, handles)
% hObject    handle to resultsOptimumBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns resultsOptimumBox contents as cell array
%        contents{get(hObject,'Value')} returns selected item from resultsOptimumBox
% --- Executes during object creation, after setting all properties.
function resultsOptimumBox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to resultsOptimumBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

EK- B RDF_GA.m

```
function [Solution rand_plot_handle CEP minCEP ellipse_plot_h cepText_handle lob_plot_h
cand_plot_rdf_handle bestCEPVal AvgCEPForBestRDFSet RDFCoordinates]=
RDF_GA(initialPop,lengthOfChromosome,numOfIteration,xoverRate,mutRate,selectRate,RDFPolyX_a
rray,RDFPolyY_array,inPoints,ResultsData,showCEPVal,showRDFVal,isItConvex)

grid_size = 10000;

% Creating initial population

for i = 1:initialPop

   for j = 1:lengthOfChromosome

      pop= [rand(1)*grid_size rand(1)*grid_size];

      in = inpolygon(pop(1,1),pop(1,2),RDFPolyX_array(1,:),RDFPolyY_array(1,:));

      while (in~= 1)

         pop=[rand(1)*grid_size rand(1)*grid_size];

         in = inpolygon(pop(1,1),pop(1,2),RDFPolyX_array(1,:),RDFPolyY_array(1,:));

      end

      if (in==1)

         colorSet= [rand(1) rand(1) rand(1)];

         rand_plot_handle(i,j)=plot(pop(1),pop(2),'+','Color',colorSet);

         candidate(i,j) ={pop};

      end

   end

end

% Find fitness values

for m=1:numOfIteration

   if (m==numOfIteration)

      lastCall=1;

   else lastCall=0;

   end

   [AvgCEPForEachRDFSet avgCEP AvgBestCEP] =
fitnessFunc(initialPop,lengthOfChromosome,inPoints,candidate,ResultsData,lastCall)

   CEP(m) =avgCEP;
```

```
minCEP(m) = AvgBestCEP;
%Intermediate Population
[SortedCEP,Cand_index]=sort(AvgCEPForEachRDFSet);
%Elitist selection
NumOfKeptChorom = ceil(length(Cand_index)*selectRate);
for th=1:NumOfKeptChorom
   for jj=1:length(candidate(1,:))
      MatePool(th,jj) = [candidate(Cand_index(th),jj)];
   end
end
%Roulette Wheel Selection
totalFitness =0;
for a=1:length(AvgCEPForEachRDFSet)
   totalFitness=totalFitness+1/SortedCEP(a);
end
for k = 1:length(Cand_index)
   Pi(k) = ((1/SortedCEP(k))/totalFitness);
end
wheel = cumsum(Pi);
t=NumOfKeptChorom+1;
while t<=initialPop
   roulette_ball = rand(1);
   for j=1:length(SortedCEP)
      if (wheel(j)>=roulette_ball)
         MatePool(t,:)= candidate(j,:,:);
         t=t+1;
         if (t>=initialPop)
            break;
         end
         break;
```

```
        end
    end
end
Int_MatePool={[] []};
Int_selectRate=0.5;
[Int_AvgCEPForEachRDFSet Int_AvgCEP] =
fitnessFunc_Int(initialPop,lengthOfChromosome,inPoints,MatePool,ResultsData)
Int_CEP(m) =Int_AvgCEP;
%Intermediate Population
[SortedInt_CEP,Cand_Int_index]=sort(Int_AvgCEPForEachRDFSet);
%Elitist selection
NumOfKeptChorom = ceil(length(Cand_Int_index)*Int_selectRate);
for th=1:NumOfKeptChorom
    for jj=1:length(MatePool(1,:))
        Int_MatePool(th,jj) = [MatePool(Cand_index(th),jj)];
    end
end
%Perform xOver and Mutation on to intermediate population
i=0;
par1_History = [];
diversityCounter = 0;
while (i<initialPop-NumOfKeptChorom)
    par1=ceil(length(Int_MatePool)*rand(1));
    par2=ceil(length(Int_MatePool)*rand(1));
    if (par1==par2)
        while (par1==par2)
            par1=ceil(length(Int_MatePool)*rand(1));
            if (par1~=par2)
                continue;
            end
```

```
        par2=ceil(length(Int_MatePool)*rand(1));
    end
end %end of par1==par2
par1_History(i+1) = par1;
par2_History(i+1) = par2;
%Cross-over operation
Pc=rand(1);
XoverPoint1=ceil(lengthOfChromosome*rand(1));
if (xoverRate>Pc)
    selected1 = Int_MatePool(par1,XoverPoint1);
    selected2 = Int_MatePool(par2,XoverPoint1);
    ma= selected1{1,1};
    fa= selected2{1,1};
    %---------------------Keep Population Diversity----------------
    %Are drawn chromosoms very similar to each other?
    %If yes discard them, and  insert new randomly created
    %individuals instead of them.
    %In order to check diversity use Euclidian distance between
    %individuals
    if (isItConvex==0)
            offSpring1_X= fa(1,1);
            offSpring1_Y= fa(1,2);
            offSpring2_X= ma(1,1);
            offSpring2_Y= ma(1,2);
    else
        beta= rand(1);
        offSpring1_X= (ma(1,1)+fa(1,1))/2 + beta*0.1*((ma(1,1)+fa(1,1))/2);
        offSpring1_Y= (ma(1,2)+fa(1,2))/2 + beta*0.1*((ma(1,2)+fa(1,2))/2);
        offSpring2_X= (ma(1,1)+fa(1,1))/2 - beta*0.1*((ma(1,1)+fa(1,1))/2);
        offSpring2_Y= (ma(1,2)+fa(1,2))/2 - beta*0.1*((ma(1,2)+fa(1,2))/2);
```

```
            inPoint1 =
inpolygons(offSpring1_X,offSpring1_Y,RDFPolyX_array(1,:),RDFPolyY_array(1,:));

            inPoint2 =
inpolygons(offSpring2_X,offSpring2_Y,RDFPolyX_array(1,:),RDFPolyY_array(1,:));

            while ((inPoint1== 0) || (inPoint2== 0))

               disp ('Inpolygons While-Loop Start..');

               beta= rand(1)*0.1;

               offSpring1_X= (ma(1,1)+fa(1,1))/2 + beta*0.1*((ma(1,1)+fa(1,1))/2);

               offSpring1_Y= (ma(1,2)+fa(1,2))/2 + beta*0.1*((ma(1,2)+fa(1,2))/2);

               offSpring2_X= (ma(1,1)+fa(1,1))/2 - beta*0.1*((ma(1,1)+fa(1,1))/2);

               offSpring2_Y= (ma(1,2)+fa(1,2))/2 - beta*0.1*((ma(1,2)+fa(1,2))/2);

               inPoint1 =
inpolygons(offSpring1_X,offSpring1_Y,RDFPolyX_array(1,:),RDFPolyY_array(1,:));

               inPoint2 =
inpolygons(offSpring2_X,offSpring2_Y,RDFPolyX_array(1,:),RDFPolyY_array(1,:));

               counter = counter+1

               if (counter>100)

                  while ((inPoint1== 0) || (inPoint2== 0))

                     par1=ceil(length(Int_MatePool)*rand(1));

                     par2=ceil(length(Int_MatePool)*rand(1));

                     if (par1==par2)

                        while (par1==par2)

                           par1=ceil(length(Int_MatePool)*rand(1));

                           if (par1~=par2)

                              continue;

                           end

                           par2=ceil(length(Int_MatePool)*rand(1));

                        end

                     end %end of par1==par2

                     XoverPoint1=ceil(lengthOfChromosome*rand(1));

                     selected1 = Int_MatePool(par1,XoverPoint1);

                     selected2 = Int_MatePool(par2,XoverPoint1);
```

```
                    ma= selected1{1,1};

                    fa= selected2{1,1};

                    beta=rand(1);

                    offSpring1_X= (ma(1,1)+fa(1,1))/2 + beta*0.1*((ma(1,1)+fa(1,1))/2);

                    offSpring1_Y= (ma(1,2)+fa(1,2))/2 + beta*0.1*((ma(1,2)+fa(1,2))/2);

                    offSpring2_X= (ma(1,1)+fa(1,1))/2 - beta*0.1*((ma(1,1)+fa(1,1))/2);

                    offSpring2_Y= (ma(1,2)+fa(1,2))/2 - beta*0.1*((ma(1,2)+fa(1,2))/2);

                    inPoint1 =
inpolygons(offSpring1_X,offSpring1_Y,RDFPolyX_array(1,:),RDFPolyY_array(1,:));

                    inPoint2 =
inpolygons(offSpring2_X,offSpring2_Y,RDFPolyX_array(1,:),RDFPolyY_array(1,:));

                end

            end

            disp ('Inpolygons While-Loop End..');

            end

        end

        CoordOffSpring1= {[offSpring1_X offSpring1_Y]};

        CoordOffSpring2= {[offSpring2_X offSpring2_Y]};

        tempParent1=Int_MatePool(par1,:);

        tempParent1(1,XoverPoint1)=CoordOffSpring1;

        Int_MatePool(NumOfKeptChorom+1+i,:) = tempParent1;

        i=i+1;

        if (i==(initialPop-NumOfKeptChorom))

            candidate =Int_MatePool;

            %Perform Mutation

            Pm=rand(1);

            MutatedChromos=ceil(length(Int_MatePool)*rand(1));

            MutatedGen=ceil(lengthOfChromosome*rand(1));

            if (mutRate>Pm)

                Mutant= Int_MatePool{MutatedChromos,MutatedGen};

                Mutant= [(Mutant(1,1)+randn) (Mutant(1,2)+randn)];
```

```
            Int_MatePool{MutatedChromos,MutatedGen}=Mutant;
        end
        break;
    end
    tempParent2=Int_MatePool(par2,:);
    tempParent2(1,XoverPoint1)=CoordOffSpring2;
    Int_MatePool(NumOfKeptChorom+1+i,:) = tempParent2;
    i=i+1;
    if (i==(initialPop))
        candidate =Int_MatePool;
        %Perform Mutation
        Pm=rand(1);
        MutatedChromos=ceil(length(Int_MatePool)*rand(1));
        MutatedGen=ceil(lengthOfChromosome*rand(1));
        if (mutRate>Pm)
            Mutant= Int_MatePool{MutatedChromos,MutatedGen};
            Mutant= [(Mutant(1,1)+randn) (Mutant(1,2)+randn)];
            Int_MatePool{MutatedChromos,MutatedGen}=Mutant;
        end
        break;
    end
else
    Int_MatePool(NumOfKeptChorom+1+i,:) = Int_MatePool(par1,:);
    i=i+1;
    if (i==(initialPop-NumOfKeptChorom))
        candidate =Int_MatePool;
        %Perform Mutation
        Pm=rand(1);
        MutatedChromos=ceil(length(Int_MatePool)*rand(1));
        MutatedGen=ceil(lengthOfChromosome*rand(1));
```

```
        if (mutRate>Pm)
            Mutant= Int_MatePool{MutatedChromos,MutatedGen};
            Mutant= [(Mutant(1,1)+randn) (Mutant(1,2)+randn)];
            Int_MatePool{MutatedChromos,MutatedGen}=Mutant;
        end
        break;
    end
    Int_MatePool(NumOfKeptChorom+1+i,:) = Int_MatePool(par2,:);
    if (i==(initialPop))
        candidate =Int_MatePool;
        %Perform Mutation
        Pm=rand(1);
        MutatedChromos=ceil(length(Int_MatePool)*rand(1));
        MutatedGen=ceil(lengthOfChromosome*rand(1));
        if (mutRate>Pm)
            Mutant= Int_MatePool{MutatedChromos,MutatedGen};
            Mutant= [(Mutant(1,1)+randn) (Mutant(1,2)+randn)];
            Int_MatePool{MutatedChromos,MutatedGen}=Mutant;
        end
        break;
    end
    i=i+1;
end %end of (xoverRate>Pc)
%Perform Mutation
Pm=rand(1);
MutatedChromos=ceil(length(Int_MatePool)*rand(1));
MutatedGen=ceil(lengthOfChromosome*rand(1));
if (mutRate>Pm)
    Mutant= Int_MatePool{MutatedChromos,MutatedGen};
    Mutant= [(Mutant(1,1)+randn) (Mutant(1,2)+randn)];
```

```
            Int_MatePool{MutatedChromos,MutatedGen}=Mutant;
        end
    end %end of while
    if(m==numOfIteration)
        delete(rand_plot_handle);
        %Solution= Int_MatePool{1,:};
        rand_plot_handle =[];
        for j = 1:lengthOfChromosome
            candidateArray = Int_MatePool{1,j};
rand_plot_handle(j)=plot(candidateArray(1,1),candidateArray(1,2),'+','Color','r','LineWidth',2);
            if (showRDFVal==1)
               a=sprintf('RDF%g',j);
               cand_plot_rdf_handle(j)=text(candidateArray(1,1)-0.25,candidateArray(1,2)-0.7,[a,'
(',num2str(candidateArray(1,1)),';',num2str(candidateArray(1,2)),')'],'FontSize',8);
            end
            Solution{j}=candidateArray;
        end
        for k=1:length(inPoints)
            for j = 1:lengthOfChromosome
                candidateArray = Int_MatePool{1,j};
                EmitterXY = inPoints{1,k};
                EmitterX = EmitterXY(1,1);
                EmitterY = EmitterXY(1,2);
                lob_x = linspace(candidateArray(1,1),EmitterX,20);
                lob_y = linspace(candidateArray(1,2),EmitterY,20);
                lob_plot_h(k,j) = plot(lob_x,lob_y,'LineWidth',1,'Color','g');
            end
        end
        for p=1:lengthOfChromosome
            RDFCoordinates{p} =Int_MatePool{1,p};
```

```
        end

        [ellipse_plot_h cepText_handle bestCEPVal AvgCEPForBestRDFSet] =
drawElipsRegions(lengthOfChromosome,inPoints,Int_MatePool,ResultsData,showCEPVal)

    end

end % End Of for loop for iteration

function  [AvgCEPForEachRDFSet AvgCEP AvgBestCEP] =
fitnessFunc(initialPop,lengthOfChromosome,inPoints,candidate,ResultsData,lastCall)

for i=1:length(ResultsData)

    diagOfCovMatr = ResultsData(1,i).stdValue*(pi/180);

    covariance_phi(i,i) =diagOfCovMatr;

end

% Population Fitness Assignment

for k=1:length(inPoints)

    for i=1:initialPop

        for j=1:lengthOfChromosome

            CandidateXY= candidate{i,j};

            CandidateX= CandidateXY(1,1);

            CandidateY= CandidateXY(1,2);

            EmitterXY=inPoints{1,k};

            EmitterX=EmitterXY(1,1);

            EmitterY=EmitterXY(1,2);

            %Sensor Measurements

            phi_rad(i,j)=atan((EmitterY-CandidateY)/(EmitterX-CandidateX));

            phi_1_deg = phi_rad(i,j)*(180/pi);

            %elips params

            X=0;Y=0;

            if (j==lengthOfChromosome)

                for (t=1:lengthOfChromosome)

                    H_ArraySet(t,:)={[sin(phi_rad(i,t)) -cos(phi_rad(i,t))]};

                end

                for x=1:lengthOfChromosome
```

```
                H_Array= H_ArraySet{x,1};
                H(x,:) = [H_Array];
            end
            Qo=inv(H'*inv(covariance_phi)*H);
            % Plot Eliptical Error Probable
            sigma_x = sqrt(Qo(1,1));
            sigma_y = sqrt(Qo(2,2));
            cor_coff = Qo(2,1)/(sigma_x*sigma_y);
            ro_xy=cor_coff;
            Cx=[sigma_x^2 ro_xy*sigma_x*sigma_y; ro_xy*sigma_x*sigma_y sigma_y^2];
            Pe=0.95;
            C=-2*log(1-Pe);
            % a semi-minor axes of ellipse
            % b semi-major axes of ellipse
            k1=2*(sigma_x^2*sigma_y^2-ro_xy^2*sigma_x^2*sigma_y^2)*C^2;
            m1=sigma_x^2+sigma_y^2+((sigma_y^2-
sigma_x^2)^2+4*ro_xy^2*sigma_x^2*sigma_y^2)^0.5;
            n1=sigma_x^2+sigma_y^2-((sigma_y^2-
sigma_x^2)^2+4*ro_xy^2*sigma_x^2*sigma_y^2)^0.5;
            b=sqrt(2*k1/m1);
            a=sqrt(2*k1/n1);
            % GDOP
            GDOP(i,k) = sqrt(a^2+b^2);
            % CEP Calculation
            CEP(i,k) = 0.75*GDOP(i,k);
        end % end of  if(j==lengthOfChromosome)
        %ellipse_plot_h(k,i,j)=plot (X,Y,'LineWidth',3, 'Color','b');
    end % end of for j=1:lengthOfChromosome
    %counter = counter+1;
  end % end of for i=1:initialPop
end % end of for k=1:length(inPoints)
```

```
AvgCEPForEachRDFSet=mean(CEP');

BestCEPValues =min(CEP');

TotalAvgCEP=mean(AvgCEPForEachRDFSet);

AvgCEP=TotalAvgCEP;

AvgBestCEP=mean(BestCEPValues);

function  [ellipse_plot_h cepText_handle bestCEPVal AvgCEPForBestRDFSet] =
drawElipsRegions(lengthOfChromosome,inPoints,candidate,ResultsData,showCEPVal)

for i=1:length(ResultsData)

   diagOfCovMatr = ResultsData(1,i).stdValue*(pi/180);

   covariance_phi(i,i) =diagOfCovMatr;

end

for k=1:length(inPoints)

   for j=1:lengthOfChromosome

      candidateArray = candidate{1,j};

      CandidateX= candidateArray(1,1);

      CandidateY= candidateArray(1,2);

      EmitterXY=inPoints{1,k};

      EmitterX=EmitterXY(1,1);

      EmitterY=EmitterXY(1,2);

      %Sensor Measurements

      phi_rad(i,j)=atan((EmitterY-CandidateY)/(EmitterX-CandidateX));

      phi_1_deg = phi_rad(i,j)*(180/pi);

      %elips params

      X=0;Y=0;

      if (j==lengthOfChromosome)

         for (t=1:lengthOfChromosome)

            H_ArraySet(t,:)={[sin(phi_rad(i,t)) -cos(phi_rad(i,t))]};

         end

         for x=1:lengthOfChromosome

            H_Array= H_ArraySet{x,1};
```

```
    H(x,:) = [H_Array];
end
Qo=inv(H'*inv(covariance_phi)*H);
% Plot Eliptical Error Probable
sigma_x = sqrt(Qo(1,1));
sigma_y = sqrt(Qo(2,2));
cor_coff = Qo(2,1)/(sigma_x*sigma_y);
ro_xy=cor_coff;
Cx=[sigma_x^2 ro_xy*sigma_x*sigma_y; ro_xy*sigma_x*sigma_y sigma_y^2];
Pe=0.95;
C=-2*log(1-Pe);
% a semi-minor axes of ellipse
% b semi-major axes of ellipse
k1=2*(sigma_x^2*sigma_y^2-ro_xy^2*sigma_x^2*sigma_y^2)*C^2;
m1=sigma_x^2+sigma_y^2+((sigma_y^2-
sigma_x^2)^2+4*ro_xy^2*sigma_x^2*sigma_y^2)^0.5;
n1=sigma_x^2+sigma_y^2-((sigma_y^2-sigma_x^2)^2+4*ro_xy^2*sigma_x^2*sigma_y^2)^0.5;
b=sqrt(2*k1/m1);
a=sqrt(2*k1/n1);
%Draw Elips Regions
x0 = EmitterX;
y0 = EmitterY;
%Ellipse Start
theta = [-0.03:0.01:2*pi];
% Parametric equation of the ellipse
x = a*cos(theta);
y = b*sin(theta);
%Inclination Angle of the major axis measured from counter-clockwise from
%the x axis
phi=0.5*atan((2*ro_xy*sigma_x*sigma_y)/(sigma_y^2-sigma_x^2));
```

```
            % Coordinate transform
            X = cos(phi)*x - sin(phi)*y;
            Y = sin(phi)*x + cos(phi)*y;
            X = X + x0;
            Y = Y + y0;
            ellipse_plot_h(k,j)=plot (X,Y,'LineWidth',3, 'Color','b');
            if (showCEPVal==1)
               % CEP Calculation
               CEP(k,j) = 0.75*sqrt(a^2+b^2);
               cepText=sprintf('CEP%g=',k);
               cepText_handle(k,j)=text(x0+0.05,y0-0.05,[cepText,' ',num2str(CEP(k,j))],'FontSize',8);
            else
               cepText_handle =1e4;
            end
         end
      end
end
bestCEPVal =0;
for g=1:length(inPoints)
        bestCEPVal =bestCEPVal+ CEP(g,lengthOfChromosome);
end
 AvgCEPForBestRDFSet=bestCEPVal /length(inPoints);
```

EK- C ÖZGEÇMİŞ

Adı Soyadı : Nebi Vuran

Doğum Yeri : Manisa

Doğum Yılı : 1976

Medeni Hali : Evli

Eğitim ve Akademik Durumu:

Lise 1990-1993 Salihli Lisesi

Lisans 1994-2002 Orta Doğu Teknik Üniversitesi Elektrik-Elektronik Mühendisliği

Yabancı Dil:

İngilizce

İş Tecrübesi:

1997 Ağustos –1998 Mayıs	SEBİT / Yazılım Mühendisi
1999 Ağustos –2000 Haziran	Meteksan Sistem / Yazılım Mühendisi
2000 Haziran –2001 Kasım	Bildem / Yazılım Mühendisi
2002 Ekim – 2004 Eylül	TÜBİTAK-BİLTEN / Araştırmacı
2004 Eylül -.....	HAVELSAN / Uzman Mühendis