BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI
MÜHENDİSLİK VE TEKNOLOJİ YÖNETİMİ TEZLİ YÜKSEK LİSANS PROGRAMI

EŞİT OLMAYAN ALANLI TESİS YERLEŞİMİNDE EVRİMSEL ALGORİTMA YAKLAŞIMI

HAZIRLAYAN

EKİN GÜLÜM

YÜKSEK LİSANS TEZİ

ANKARA - 2022

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI
MÜHENDİSLİK VE TEKNOLOJİ YÖNETİMİ TEZLİ YÜKSEK LİSANS PROGRAMI

EŞİT OLMAYAN ALANLI TESİS YERLEŞİMİNDE EVRİMSEL ALGORİTMA YAKLAŞIMI

HAZIRLAYAN

EKİN GÜLÜM

YÜKSEK LİSANS TEZİ

TEZ DANIŞMANI

DR. ÖĞR. ÜYESİ MEHMET GÜLŞEN

ANKARA - 2022

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Endüstri Mühendisliği Anabilim Dalı Mühendislik ve Teknoloji Yönetimi Tezli Yüksek Programı çerçevesinde Ekin Gülüm tarafından hazırlanan bu çalışma, aşağıdaki jüri tarafından Yüksek Lisans olarak kabul edilmiştir.

Tez Savunma Tarihi: 27 / 05 / 2022

Tez Adı: Eşit Olmayan Alanlı Tesis Yerleşimi Probleminde Evrimsel Algoritma Yaklaşımı
(Tez konusunun başlığı, kelimelerin baş harfleri büyük olacak şekilde "bold" yapılmadan yazılacaktır.)

Tez Jüri Üyeleri (Unvanı, Adı - Soyadı, Kurumu) **İmza**
(Jüri üyelerinin Unvanı, Adı-Soyadı ve Kurumları "bold" yapılmadan yazılacaktır.)

Dr. Öğr. Üyesi Fatma Pınar Göksal Aksaray Üniversitesi ……..

Dr. Öğr. Üyesi Mehmet Gülsen Başkent Üniversitesi ……..

Dr. Öğr. Üyesi Burak Yıldız Başkent Üniversitesi ……..

ONAY

Prof Dr. Ömer Faruk Elaldı
Fen Bilimleri Enstitüsü Müdürü
Tarih : … / … / ….……

BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLER ENSTİTÜSÜ

YÜKSEK LİSANS TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: … / … / 20…

Öğrencinin Adı, Soyadı : EKİN GÜLÜM

Öğrencinin Numarası : 21920278

Anabilim Dalı : Endüstri Mühendisliği

Programı : Teknoloji ve Mühendislik Yönetimi

Danışmanın Unvanı/Adı, Soyadı : Dr. Öğr. Üyesi Mehmet GÜLŞEN

Tez Başlığı : EŞİT OLMAYAN ALANLI TESİS YERLEŞİMİNDE EVRİMSEL ALGORİTMA YAKLAŞIMI

Yukarıda başlığı belirtilen Yüksek Lisans/Doktora tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam ..... sayfalık kısmına ilişkin, ….. / … / 20 …. tarihinde şahsım/tez danışmanım tarafından .....adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı %..... 'dır. Uygulanan filtrelemeler:

1. Kaynakça hariç
2. Alıntılar hariç
3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası:…….

ONAY

Tarih: … / … / 20…

Öğrenci Danışmanı Unvan, Adı, Soyadı, İmza:

Dr. Öğr. Üyesi Mehmet Gülşen

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren, kıymetli tecrübelerinden faydalandığım, her hafta bana her koşulda vakit ayıran tez danışmanım ve değerli Hocam Dr. Öğr. Üyesi Mehmet GÜLŞEN'e en içten duygularla minnet ve şükranlarımı sunarım.

Yaşantım boyunca en zor anlarımda bana destek olan, güç veren ve her koşulda ana okuldan yüksek lisansa kadar eğitim hayatımı fedakârca destekleyen annem Arzu GÜLÜM ve babam İlker GÜLÜM'e, bu süreçte yanımda olan, desteklerini esirgemeyen ve hiçbir zaman beni yalnız bırakmayan Estel BAYSAL'a teşekkürü borç bilirim. Sizler yanımda olmasaydınız belki de bu tez hiç yazılmazdı.

ÖZET

Ekin GÜLÜM

EŞİT OLMAYAN ALANLI TESİS YERLEŞİMİNDE EVRİMSEL ALGORİTMA YAKLAŞIMI
Başkent Üniversitesi Fen Bilimleri Enstitüsü

Endüstri Mühendisliği Anabilim Dalı

2022

Yerleşme problemleri departman, makine, cihaz veya bir elektrik devresinde parçaları bazı kriterlere ve kısıtlara bağlı olarak yerleştirme problemleridir. Tesis yerleşim planlaması NP-Zor bir problem olmasından dolayı çözümü en zor problemlerden birisidir. Genel olarak amacı belirli sayıda departmanın en ideal yerleşimini yapmaktır. Bunu yaparken çoğunlukla gerçek uygulamada departmanların alanının birbirine eşit olmadığı varsayılmaktadır. Bu tez kapsamında, tesis yerleşim problemi alanının en önemli ve en çok ele alınan konularından olan eşit olmayan alanlı tesis yerleşimi (EOATY) problemi ele alınacaktır. Bu problemde departmanlar arası akışların zamanla değişmediğini yani "statik" olduğu varsayılmaktadır. Öncelikle literatürde matematiksel model kullanılarak çözülen doğrusal ve doğrusal olmayan problemler incelenmiştir. Bu modellemelerin yüksek sayıdaki departman sayılarını çözme yeterliliği olmadığından dolayı sezgisel ve meta-sezgisel algoritmalar incelenmiştir. Bunun sonucunda yeni bir evrimsel algoritma geliştirilmiştir. Problem parametreleri Taguchi deney metodu kullanılarak seçilmiştir. Daha sonrasında geliştirilen evrimsel algoritma literatürde de sıklıkla kullanılan belirli problem setlerinde denenip algoritmanın güçlülüğü raporlanmıştır. Yapılan çalışma bu literatürde daha önce farklı açılardan ele alınmış problemi, bir bütünlük içinde ele alarak alternatif konfigürasyonlarda çözüm üretilmesini sağlamıştır. Evrimsel algoritma yaklaşımı kullanılarak, yatay ve dikey FBS (Flexible Bay Stucture) ve STS (Slicing Tree Structure) konfigürasyonları için tesis yerleşim planı bulunmuştur.

ANAHTAR KELİMELER: Eşit olmayan alanlı tesis yerleşimi, Evrimsel Algoritma, FBS, STS, Taguchi metodu

ABSTRACT

Ekin GÜLÜM

EVOLUTIONARY ALGORITHM APPROACH IN UNEQUAL AREA LAYOUT PLANNING PROBLEM

Başkent University, Institute of Science and Engineering

Department of Industrial Engineering

2022

Placement of components on a layout based on a priori determined criteria and limitations is a well-known problem in facility layout planning, circuit design, or machine placement areas. Most layout problems are NP-Hard, and they cannot be solved optimally. The main objective in layout problems is to determine the relative positions of departments that minimize the flow cost among them. This study focuses on Unequal Area Facility Layout planning (UA-FLP), one of the most challenging areas in facility layout research. The problem is solved as a static flow problem, assuming that flows between departments are not changing over time. As a first step of the study, mathematical models used previously to solve linear and nonlinear problems are thoroughly examined. Secondly, heuristic and meta-heuristic algorithms have been investigated since mathematical models are not capable of solving problems with a large number of departments. As a result, a new evolutionary algorithm has been developed. Parameters of the algorithm were selected using the Taguchi experimental method. The algorithm was tested on problem sets that are also frequently used in the literature, and its performance was analyzed. The proposed approach provides alternative layout configurations under a single framework. Contrary to the previous works that focus on a single configuration, the proposed evolutionary algorithm approach generates three alternative solutions, each with a separate layout configuration. Three alternatives considered include horizontal and vertical FBSs (Flexible Bay Structure) and STS (Slicing Tree Structure).

KEYWORDS: Unequal Area Facility Layout Problem, Evolutionary Algorithm, FBS, STS, Taguchi design

İÇİNDEKİLER

TABLOLAR LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

SİMGELER VE KISALTMALAR LİSTESİ

a_i	i'inci departmanın en-boy oranı
AR	Ayarlama oranı
CR	Önem oranı
DM	Yer değiştirme mutasyonu
EA	Evrimsel Algoritma
FBS	Flexible Bay Structure (Esnek Kuşak Yapısı)
FLP	Facility Layout Problem (Tesis yerleşim problemi)
GA	Genetik algoritma
l_i^x	i'inci departmanın x koordinatı uzunluğu
l^y	i'inci departmanın y koordinatı uzunluğu
MIP	Mixed Integer Programming (Karışık Tamsayılı Programlama)
OFV	Objective Function Value (Amaç Fonksiyonu Değeri)
OX	Sıralı çaprazlama
UA-FLP	Unequal Area Facility Layout Problem (Eşit olmayan alanlı tesis yerleşim problemi)
PF	Penalty function (Ceza katsayısı)
PMX	Kısmen eşleştirilmiş çaprazlama
RAR	Yeniden ayarlama oranı
RCX	Azaltılmış vekil çaprazlama
SA	Simulated Annealing (Benzetilmiş Tavlama)
SIM	Basit ters çevirme mutasyon operatörü
SM	Karıştırma mutasyonu
STS	Slicing Tree Structure (Kesim Ağacı Yapısı)
S/N	Signal to Noise (Sinyal-Gürültü)
TS	Tabu Search (Tabu araması)
QAP	Quadratic Assignment Problem (Karesel Atama Problemi)

1. GİRİŞ

Yerleşme problemleri departman, makine, cihaz veya bir elektrik devresinde parçaları bazı kriterlere ve kısıtlara bağlı olarak yerleştirme problemleridir [53]. Etkili ve iyi tasarlanmış bir yerleşim planı, üretim sürelerini azaltabilir, kullanıldığı alanda verimliliği arttırabilir ve çıkan üretim miktarını arttırabilir. Bu nedenle etkili yerleşim planı oldukça önemlidir. Tesis yerleşim planı sadece belli sayıdaki departmanı belli bir alana yerleştirmek için kullanılmayıp, devredeki elektrik elemanlarının en efektif şekilde yerleştirilmesi için dahi kullanılabilir. Tesis yerleşim planlaması üretim tesisi planlaması, hastanelerin departman tasarımı ve hizmet merkezlerinin tasarlamasında kullanılmaktadır.

Tesis yerleşim planlaması matematiksel doğası gereği olarak NP-Zor bir problem olmasından dolayı en önemli ve çözümü en zor problemlerden birisidir [54]. Genel olarak amacı belirli sayıda departmanın en ideal yerleşimini yapmaktır. Bunu yaparken çoğunlukla gerçek uygulamada departmanların alanının birbirine eşit olmamasını varsaymaktadır. Teknoloji ve yeni gelişmeler oldukça çok az firma eski tesisleri kullanmaya devam etmektedir. Bu gelişmelere ve yeni teknolojilere ayak uydurmak için çoğu fabrika ve tesiste değişikler yapılması gerekmektedir. Günümüzde piyasa koşullarından ve rekabetinden dolayı üretim firmaları çoğunlukla zaman, maliyet ve kalite konularında öne geçmeye çalışmaktadır. Tesis yerleşim planlaması da aslında bu konulardan maliyetle direkt olarak ilgilenmekte olup bu maliyeti azaltmaya çalışmaktadır.

Tesis yerleşim planının sonucu bize departmanların birbirine göre yerlerini gösteren bir yerleşim planıdır. İdeal bir blok yerleşimi elde edildikten sonra bu yerleşim üzerinde geliştirmeler veya iyileştirmeler yapılabilmektedir. Her problemin kendi isterleri olduğu gibi bazı problemlere özel olarak tesis yerleşimi elde edildikten sonra koridor düzenlemeleri yapılabilinir, departmanların giriş çıkış noktaları belirlenebilir [53].

Özellikle üretim tesisi planlanırken aslında tesis problemi incelenirken bazı varsayımlar yapılmaktadır. Hem tesis yerleşim planlaması kendi içinde hem de üretim sistemleri ve ham malzeme gereklilikleri kendi içinde karışık olduğundan üretim sistemleri için bazı varsayımlar yapılmaktadır. Bu tip tesis yerleşim planlamasındaki amaç materyal taşıma maliyetlerinin en küçüklenmesidir. Planlamada varsayılanlardan belki de en önemlisi her departman arasında malzemeleri taşımayı sağlayacak ve her zaman kullanılabilir durumda olan ekipmanların bulunmasıdır. Bu durumda her gerektiğinde vakit kaybetmeden

bu malzemeler hızlıca ve öncesinde belirtilen sabit bir aletle taşınacak olması varsayılmaktadır. Bu durumda başka bir varsayım ise önceden belirtilen departmanlar arasındaki ilişki miktarının zamanla değişmeyeceğidir. Malzeme taşınmasında herhangi bir problemle karşılaşılmayacağı ve malzemeyi taşıyacak ekipmanların kurulmasının maliyeti göze alınmamaktadır.

Yerleşim probleminin tipi de problemin çözümüne yaklaşmak için önemli bir kriterdir. Burada iki farklı yerleşim planı düşünülebilir: Makine yerleşimi ve tesis yerleşimi. Aslında bir üretim merkezinde ya da herhangi bir yerde makine yerleşim problemi belli sayıda departmanı belli büyüklükte bir tesise yerleşim probleminden daha fazla varsayım istemektedir. Makine yerleşiminde her makine arasında o makinenin erişilmesi için belirli bir boşluk bırakılması, makineler arasında koridorların bulunması ve bu makinelerin giriş çıkış noktalarının nerede bulunacağı gibi karar tipleri bulunmaktadır. Departmanların yerleştirilmesinde ise daha kolay varsayımlar yapılabilmektedir. Departmanlar makinelere göre daha kolay yerleştirilebilir ve daha kolay şekillere sokulabileceği için bu yerleşim problemi daha az varsayım içermektedir. Burada önemli olan departmanların üst üste çakışmaması ve bir departmanın uzun tarafı ve kısa tarafının oranının belli bir oranı geçmemesine dikkat etmek olacaktır. Genel olarak araştırmaların çoğu departmanların şekillerinin dikdörtgensel olacağını varsaymaktadır [53],[55],[56],[57]. Bu varsayımda aslında farklı şekillerde departman şekillerinin ortaya çıkmasını engellemektedir.

Yerleşim problemlerinin aslında fabrikaya ve işyerine birçok faydası bulunmaktadır. Bu faydalar arasında müşteri taleplerine hızlı cevap verebilmek, maliyetleri düşürmek ve tedarik zincirinin karlılığını arttırmak, işyerinin büyümesini geliştirilmiş malzeme taşınması ve malzeme kontrolüyle sağlamak, yatırımın geri dönüşünü maliyetleri azaltarak maksimize etmek, kolay erişilebilir olmak ve bakımların kolayca yapılabilmesi ve çalışanlar için güvenliği sağlaması, enerji verimli olması ve işyerinin olması gereken çevreye sorumluluğunun erine getirilmesi gibi bir kuruma sağlayabileceği oldukça faydası bulunmaktadır. Bu nedenle iyi bir tesis yerleşimi planlaması aslında oldukça önemlidir.

Tesis yerleşiminde en önemli konseptlerden birisi departmanların birbirleri arasındaki mesafedir. Mesafenin önemli olması aslında direkt olarak materyal taşınmasının maliyetine etki etmesidir. Burada iki farklı yaklaşım kullanılmaktadır. Bunlardan bir tanesi mesafe bazlı bir tanesi ise bitişikliktir. Genel olarak literatürde mesafe bazlı yaklaşımlar problemlerde kullanılmaktadır. Bu problemlerde belli departmanların yan yana olması gibi

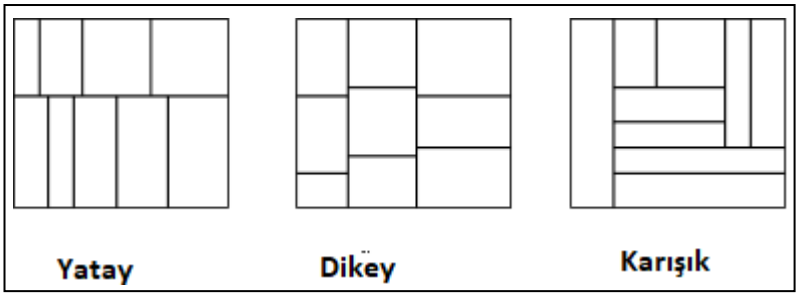
belli yaklaşımlar ve varsayımlar da olabilmektedir. Bu durumlarda optimal çözümlerde böyle varsayımlar da dikkate alınmalıdır.

Tesis yerleşim problemini incelenmesinin en önemli nedenlerinden birisi optimal olarak çözülmesinin aslında departman sayısı arttıkça olanaksız hale gelmesidir [54]. Bunun nedeni Np-Zor bir problem olmasıdır çünkü bu problem minimum doğrusal düzenleme probleminin bir genellemesi olmasıdır. Bu nedenle literatürde ilk çıkış noktası optimal tesis yerleşimini bulmak olsa da daha sonrasında literatür de yaklaşık sonuçlar bulmaya odaklanılmıştır. Bu probleme yaklaşım aslında iki alanda incelenebilir: Kesin yaklaşımlar ve kesin olmayan yaklaşımlar. Kesin yaklaşımlar tesis yerleşim problemine optimal sonuçları bulmayı hedeflemiştir ancak henüz literatürde 11 departmandan fazla sayıda tesisyerleşim planında optimal sonuç veren bir yaklaşım bulunamamıştır [10]. Bu nedenle departman sayısı gerçek hayatta oldukça fazla olabildiğinden sezgisel ve meta sezgisel yaklaşımlar uygulanmaya başlanmıştır. Kesin yaklaşımlar matematiksel modelleme ve dal- sınır tekniklerini kullanarak optimal olarak çözmeye çalışmaktadır [53]. Sezgisel yöntemlerise oluşturma ve geliştirmeye odaklanmaktadır. Son olarak meta sezgisel yöntemlerde aslında sezgisel yöntemlerin birlikte kullanılmasıyla ortaya çıkmıştır. Meta sezgisel yöntemlere örnek olarak Benzetimli Tavlama, Genetik Algoritması ve Tabu Araması örnekverilebilir. Aslında meta sezgisel yöntemlerde kendi içinde ikiye ayrılmaktadır: Evrensel arama metotları (Tabu Araması ve Benzetimli Tavlama) ve Evrimsel Algoritmalar (Genetik Algoritma ve Karınca Kolonisi Algoritması). Bu tip yöntemlerin amacı aslında çözüm kümesinde çeşitli tiplerde arama yaparak tesis yerleşiminde optimal sonuçları bulmaktır. Burada verilmesi gereken en önemli kararlardan birisi de aday tesis yerleşim planlarının nasıl gösterileceğidir. Gösterim şekilleri aslında bu yöntemlerin gidiş yollarında da önemli bir yere sahip olmaktadır. Tesis yerleşimi birçok şekilde kâğıt üzerinde gösterilecek şekilde ifade edilebilmektedir.

Bu çalışmada Tesis Yerleşim Probleminin (Facility Layout Planning) araştırılan kısmı Eşit Olmayan Alanlı Tesis Yerleşim Problemi (UA-FLP)'dir. Bu çalışmada ele alınan problem olan UA-FLP'nin amacı, bölünemez ve üst üste çakışmayan ve değişik alandaki departmanların arasındaki materyal akışının en küçüklenecek şekilde tesis içine yerleştirilmesinin sağlanmasıdır. Sütun FLP'lerinden farkı olarak her departmanın eni ve boyu geometrik kurallarına uyarak en iyilenmeye çalışılmaktadır. Sütun FLP çalışmalarında departmanların boyları sütun boyuna eşit olduğundan sabittir ancak bu kısıt UA-FLP

probleminde bulunmamaktadır. Çözülmeye çalışılan problem statik talepli bir problemdir yani departmanlar arası veri akışı zamanla değişmeyip, süreç boyunca sabit kalacağı varsayımı yapılmıştır. Bu problemi çözerken 3 farklı yerleşim türü olan yatay FBS, dikey FBS ve karışık FBS (Slicing Tree Structure) gösterimleri kullanılacaktır. Şu ana kadar kullanılan matematiksel modellerle ulaşılan en iyi çözümler gerçek hayatta kullanılacak departman sayılarına oranla az olduğundan UA-FLP problemini çözerken evrimsel algoritma yaklaşımı kullanılacak ve 3 gösterim tipinin farklı özelliklerinden yararlanılacaktır.

Bu tez literatürde yer verilmemiş olan genel bir bakış açısı kazandırmak adına önemlidir. Kesin çözümler 11 departmana kadar sonuç verebildiği için literatürde sezgisel algoritmaların önemi oldukça fazladır [10]. FBS problemleri üç ana kategoride ele alınabilinir: "Bay" olarak adlandırılan kuşakların yatay, dikey ve karmaşık bir şekilde konfigure edildiği sistemlerdir (Şekil 1.1) Şu ana kadar literatürde yer alan çalışmalar tek bir gösterim şekli üzerinde durup geliştirilen algoritmalar tek bir gösterim şeklini kullanarak sonuç bulmaya odaklanmaktadır. Bu üç farklı konfigürasyon birbiri ile alakalı olup, yapılan alan planlamasının (layout planning) etkili olması için her bir konfigürasyonun hesaplanıp değerlendirilmeye alınması gerekmektedir. Ancak bu çalışmada literatürde sıklıkla kullanılan FBS ve Karışık FBS (STS) gösterimleri bir arada kullanılmakta olup iki gösterim şeklinin güçlü ve zayıf yönleri aynı anda karşılaştırılabilmektedir. Bunun yanında FBS tipi gösterimde "bay"lerin yönü de iki farklı şekilde incelenmiş olup aynı problemler üzerinde yatay ve dikey "bay"lerin vereceği sonuçların farklılığı gösterilmektedir. Kullanılan evrimsel algoritma ise literatürde bulunmamaktadır. Çalışmada hem genetik algoritmanın hem de literatürde son zamanlarda incelenmeye başlanan harmonik algoritmanın özellikleri alınarak yeni bir arama geliştirilmiştir. Bu algoritmanın gücü ve gösterimdeki farklılıkların önemi, önceki çalışmaların sonuçlarına bakarak kıyaslanacaktır.



Şekil 1.1 Yatay, dikey ve karışık FBS gösterimleri

Bu problemler ve koşullar eşliğinde bir sonraki bölümde tesis yerleşim planlamasının literatürde ele alınan varsayımları ve yaklaşımları belirtilecektir. Bu yaklaşımların incelenmesi aslında tesis yerleşim problemine genel bir bakış sağlayacak ve geliştirilmeye açık yerler hakkında fikir verecektir. Burada genel bakış tek zamanlı(statik) tesis yerleşim planlaması konusunda olacaktır. Tesis yerleşiminde optimal sonucu bulmanın departman sayısı arttıkça olanaksız hale gelmesinden dolayı tez genel olarak tesis yerleşiminde meta sezgisel yöntem kullanılarak oluşturulmasını hedeflemektedir. Ele alınacak meta sezgisel yaklaşım ise genetik algoritma olacaktır. Departmanlarının şekillerinin dikdörtgen olması gerektiği, zamanın sabit olduğu yani herhangi bir maliyetin ya da departmanın yerinin tesis yerleşimi bulunduktan sonra değiştirilmesi gerekliliğinin olmadığı ve departmanlar arası uzaklık matrisinin departmanlarının merkezlerinin arasındaki mesafenin ölçülmesiyle bulunacağı varsayılmaktadır. Ayrıca departmanlarının uzun ve kısa taraflarının oranında belli bir oranı geçmemesi hedeflenmektedir.

2. EŞİT OLMAYAN ALANLI YERLEŞİM PROBLEMİ(UA-FLP) LİTERATÜR TARAMASI

Bu bölümde eşit olmayan alanlı tesis yerleşimi için kısa bir özet verilecektir. Sonrasında bu problemin modellenmesinde kullanılan farklı yöntemler incelenip en sonda ise karma bir MIP modeli kodu verilecektir.

Bu problemin amacı belirli bir alana "n" tane departmanın en efektif şekilde yerleştirilmesidir. Bu problem fabrika, okul, hastane ve havaalanı yerleşimi, depolar ve devre elemanları yerleştirilmesinde kullanılabilmektedir. En sık kullanın amaç fonksiyonu, departmanlar arasında materyal alışverişinin en küçüklenmesidir. Bunun yanında nitel olduğu gibi çeşitli araştırmalarda nicel amaç fonksiyonları da belirtilmiştir [53]. Departmanlar arası materyal akışının en küçüklenmiş olması daha az materyal akışına ve daha az work-in-process (yarı mamul) stoğuna yol açmaktadır. Bu problemin çıktısı blokbir yerleşim çıktısıdır. Bu yerleşimde departmanların birbirlerine göre yerleri belirtilmektedir.

Tesis yerleşim problemleri optimal olarak çözülmesi zor bir problemdir. Bunun nedeni bu problemin NP-Zor olmasıdır. Bu nedenle en güçlü bilgisayarlar bile deparmant sayısı artıkça problemi optimal olarak çözememektedir. Problemin NP-Zor özelliğinden dolayı çeşitli buluşsal metodlar da bu problemin çözülmesi için kullanılmaktadır.

Karma tam sayılı modelleme birçok makaleye ve araştırmaya konu olmuştur. Bu problem NP-Zor olduğundan işlem zamanını düşürmeye çalışacak farklı türde varsayımlar ve modellemeler yapılmıştır. Örneğin, sadece departmanların üst üste çakışmaması için kullanılan ikili karar değişkenleri bile basit bir matematiksel model çözümünün süresini ciddi bir biçimde uzatmaktadır. Literatürde kullanılan terimlerden aşağıda bahsedilecektir.

a) Literatürde Modelleme
b) Yerleşim Gösterimleri
c) Kullanılan Amaç Fonskiyonları
d) Kullanılan Kısıtlar
e) Kullanılan veri setleri
f) Çözüm Teknikleri

2.1. Literatürde Modelleme

2.1.1. QAP modeli

Tesis yerleşim planlamasında en çok kullanılan modellerden birisi Karesel atama Problemi modellerdir. Bu model ilk defa Koopmans et al [1] ekonomik aktivitelerde kullanılmak için geliştirilmiştir. Tesis yerleşimi olarak tanımlanan ilk problem olup ana konusu doğrusal atama problemidir. Bu modelin dezavantajı ise problem büyüklüğü arttıkça, problemin çözümünün oldukça uzun sürmesidir. Karesel atama Problemi modelinde amaç fonksiyonu maliyeti, uzaklığı ya da akışı en küçüklemek olabilir. Aşağıda Koopmans and Beckman'ın ilk defa kullandığı model bulunmaktadır.

$$\min \quad \sum_{i,j=1}^{n} \sum_{k,p=1}^{n} f_{ij} d_{kp} x_{ik} x_{jp} \tag{1.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1 \quad 1 \leqslant j \leqslant n, \tag{1.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad 1 \leqslant i \leqslant n, \tag{1.3}$$

$$x_{ij} \in \{0,1\} \quad 1 \leqslant i,j \leqslant n. \tag{1.4}$$

x_{iJ}= 1 tesis i yerleşke j'ye atandıysa

x_{ij}= 0 tesis i yerleşke j'ye atanmadıysa

f_{ij} : tesis i ile tesis k arasındaki materyal akışı

d_{kp} : yerleşke k ve yerleşke p arasındaki mesafe

Burada 1.2deki kısıtlama sadece bir tesisin bir yerleşkeye atanmasını sağlamaktadır ve 1.3'deki kısıtlamada her yerleşkenin sadece bir tesise atanmasını sağlamaktadır. Buradaki amaç toplam materyal akışını n departman için en küçüklemektir. 1.4'deki kısıtlama ise x_{ij}leri ikili değişken olduğunu belirtmektedir.

2.1.2. Graf teorisi modeli

Tesis yerleşim probleminin optimal sonucunun alınmasının zor olması ve çeşitli sezgisel metotlar kullanarak yarı optimal sonuçların kabul edilmesi gerektiğinde ve özellikle ilişki tablosu kullanıldığında iki görev tamamlanmalıdır. Bunlardan birincisi optimal sonuç için bir üst sınır oluşturulmalıdır. Böylelikle yaklaşık sonuçlarla ilgili bir kıstas olmalıdır.

İkincisi ise yaklaşık optimal sonucu iyileştirmek için oldukça fazla bitişiklik ilişkisini(özellikle güçlü olanları) sağlamalıdır [2].

Bu modellemede iki tesisin yan yana yerleştirilmesinin istenirliğinin bilindiği varsayılmaktadır. Başlangıçta departmanların alanları ve şekilleri göze alınmamaktadır ve her departman grafta bir nod olarak tanımlanmaktadır. Departmanların bitişiklik ilişkileri iki bitişik departman(nod) arasındaki arklar ile gösterilir. QAP modelindeki gibi Graf teorisi modellerinin de küçük sayıdaki departmanlarla dahi en küçük sonucu verememektedir. Formülasyonu aşağıdaki gibidir.

$$\text{En büyükleme } Z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_{ij}$$

Graf düzlemseldir (2.1)

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij} = 3n-6 \quad (2.2)$$

x_{ij} {0,1} for all i,j (2.3)

w_{ij} tesis i ve tesis j'nin ilişki ağırlığı (2.4)

x_{ij}= 1 eğer tesis i ve tesis j bitişikse. Öbür türlü 0 (2.5)

2.1.3. MIP modeli

Tesis yerleşim planlamasında MIP'i ilk Montreuil kullanmıştır. Burada modeli oluştururken uzaklık temelli bir amaç fonksiyonu kullanmış olup sürekli bir yerleşim gösterimi kullanmıştır. Aslında bu modelleme ayrık QAP'nin uzantısıdır. Bu modellemenin MIP olarak formulüze edilebilmesi için bütün departman şekillerinin kare olduğu varsayılmalıdır. Burada başlangıç yerleşim planına gerek yoktur. Matematiksel MIP zaten kendiliğinden oluşturacaktır. MIP modelinin girdileri Montreuil [3] halinde aşağıdaki gibidir.

Parametreler,

B_x : Bina uzunluğu

B_y: Bina genişliği

A_i : Departman i'nin alanı

L_i^l: Departman i'nin uzunluğunun alt limiti

L_i^u : Departman i'nin uzunluğunun üst limiti

W_i^l : Departman i'nin genişliğinin alt limiti

W_i^u : Departman i'nin genişliğinin üst limiti

M : Büyük bir sayı

Karar değişkenleri,

α_i : Departman i'nin merkezinin x koordinatı

β_i : Departman i'nin merkezinin y koordinatı

x_i': Departman i'nin sol tarafının x koordinatı

x_i'' : Departman i'nin sağ tarafının x koordinatı

y_i' : Departman i'nin üst tarafının y koordinatı

y_i'' : Departman i'nin alt tarafının y koordinatı

z_{ij}^x=1 eğer departman i departman j'nin doğusunda ise. Öbür türlü 0

z_{ij}^y=1 eğer departman i departman j'nin kuzeyinde ise. Öbür türlü 0

Üstte belirtilen karar değişkenleri ve parametreleri kullanarak modelkısıtları ve amaç fonksiyonu aşağıdaki gibidir:

$$\text{En küçükleme } Z = \sum_i \sum_j f_{ij} c_{ij} \left(|\alpha_i - \alpha_j| + |\beta_i - \beta_j| \right)$$

Kısıtlar altında

$L_i^l <= (x_i'' - x_i') <= L_i^u$	for all i	(3.1)
$W_i^l <= (y_i'' - y_i') <= W_i^u$	for all i	(3.2)
$(x_i'' - x_i') * (y_i'' - y_i') = A_i$	for all i	(3.3)
$0 <= x_i' <= x_i'' <= B_x$	for all i	(3.4)
$0 <= y_i' <= y_i'' <= B_y$	for all i	(3.5)
$\alpha_i = 0.5*x_i' + 0.5* x_i''$	for all i	(3.6)
$\beta_i = 0.5*y_i' + 0.5* y_i''$	for all i	(3.7)
$x_j'' <= x_i' + M (1- z_{ij}^x)$	for all i and j, i≠j	(3.8)
$y_j'' <= y_i' + M (1- z_{ij}^y)$	for all i and j, i≠j	(3.9)
$z_{ij}^x + z_{ji}^x + z_{ij}^y + z_{ji}^y >= 1$	for all i and j, i<j	(3.10)
$\alpha_i, \beta_i >= 0$	for all i	(3.11)
$x_i', x_i'', y_i', y_i'' >=0$	for all i	(3.12)
z_{ij}^x, z_{ij}^y 0/1 Integer	for all i and j, i≠j	(3.13)

Amaç fonksiyonu uzaklığa bağlı, departmanlar arası akışa ve bu akışın maliyetine bağlı bir fonksiyondur. (3.1) ve (3.2) numaralı kısıtlar departmanları binanın uzunluğuna

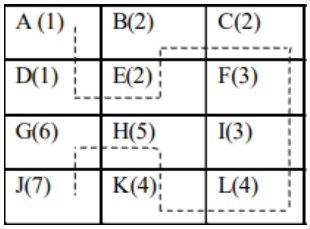
ve genişliğine göre yerleştirmektedir. (3.3) numaralı kısıt departmanların alan kısıtıdır ancak bu kısıt bu problemin doğrusallığını bozmaktadır. Kısıtlar (3.4) ve (3.5) departmanların köşelerini bina uzunluğunu ve genişliğini geçmeyecek şekilde yerleştirilmesini sağlamaktadır. (3.6) ve (3.7) kısıtları, sırasıyla departmanların x ve y merkezlerinin koordinatlarını göstermektedirler. (3.8) numaralı kısıtlama ise $z^{x}_{\bar{i}\bar{j}}$=1 olduğunda x''_{J} <= x'_{i}(departman i departman j'nin doğusunda) olmasını zorlamaktadır. Bu kısıt sadece $z^{x}_{\bar{i}\bar{j}}$=1 olduğunda çalışmaktadır. Kısıt (3.9) kısıt (3.8) ile aynı mantıkta ancak y doğrultusunda çalışmaktadır. (3.10) numaralı kısıt ise departmanların üst üste çakışmama koşulunu sağlamaktadır. Bunu her departman ikilisinin en azından doğu-batı ya da kuzey-güney ayrımı yaparak sağlamaktadır. Son olarak (3.11) ve (3.12) numaralı kısıtlar pozitiflik kısıtları olup (3.13) numaralı kısıt ise ikili karar değişkenlerini temsil etmektedir.

Üstteki modelde geçen (3.3) numaralı kısıttan dolayı bu model yukarıda da belirtildiği gibi doğrusal bir model değildir. Eşit olmayan alanlı tesis yerleşim problemini optimal olarak çözmek için değişik yollar bulunmaktadır. Örnek olarak dışbükey olmayan formunda hali hazırda optimal kısıtları çözebilecek programlarda çözülebilir. Bu optimizasyon metodlarından bazıları AlphaBB ve BARON gibi dönüşüm metotlarıdır [3]. Bu problem doğrusal olmayan çözücülerle çözülebileceği gibi modeli ve kısıtları doğrusallaştırmak için de çalışmalar gerçekleştirilmiştir. Yapılan çalışmalarda bu kısıtı doğrusallaştırmak için birçok çalışma ve araştırma yapılmıştır. İleride bahsedilecek kısıtlardaki farklılıklar bölümünde bu çalışmalar ve kullanılan tekniklerden bahsedilecektir.

2.2. Yerleşim Gösterimleri

2.2.1. Şebeke gösterimi

Buradaki şebeke yaklaşımı aslında QAP modelin bir kısıtlamasının geliştirilmesi için ortaya çıkmıştır. QAP modelde departmanların eşit alanlarda olduğu varsayımı yapılmaktadır. Ancak bildiğimiz gibi bu varsayım gerçek hayata uyumlu değildir. Bu nedenle eşit olmayan alanlı yerleşim planı sorununu çözebilmek için QAP modeli her biri eşit alandaki şebekelere bölünmektedir. Her departman kendi alanı kadar şebeke kaplamaktadır. Eğer boş bir alan var ise de oraya sahte boşluklar atanmaktadır [4].



Şekil 2.. Şebeke gösterimi Örneği [4]

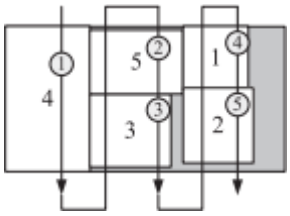
Şebeke gösteriminin ana faydalarından bir tanesi dikdörtgensel şekle sahip olmayan departmanlara yerleşimde yer vermesidir. Gerçek hayatta birçok L-şekilli, T-şekilli ve V-şekilli departman olmasından dolayı ilk başta görülen ve en önemli yararlarından birisi de gerçek departman şekillerinin modellenmesine izin vermesidir. Bir diğer yararı ise görüleceği üzere diğer gösterimler sadece dikdörtgensel şekilde departmanlara izin verdiğinden dolayı oluşturulabilecek aday çözümler azalacaktır ve bu da yüksek ihtimalle amaç fonksiyonun artmasına neden olacaktır. Bir diğer avantajı ise yerleşim yapıldıktan sonra oluşabilecek olan boş alanların daha efektif bir şekilde oluşturulabilecektir.

Buradaki temel problem modeldeki departman yerleşiminde aynı departmandaki şebekelerin bitişik olma zorunluluğudur. Bitişikliğin garantilenmesi için Space Filling Curve (SFC) adı verilen bir method uygulanmaktadır. Yerleşimin oluşturulması için bütün şebekeleri dolaşan bir SFC kullanılması öngörülmüştür. Aynı departmanların şebekesini bitirmeden başka bir departmanın şebekesine geçmediğinden bitişikliği sağlamış olur. Bu gösterimdeki başka bir problem ise önceden bir departmanı göstermek için kaç tane dikdörtgensel alanın gerekli olduğu bilinmemesidir. Hatta kaç tane departmanın dikdörtgensel şekle uymayacağı bilinmemektedir [5].

2.2.2. Flexible Bay Structure (FBS)

İlk defa Tong tarafından tanımlanmıştır. Burada tanımlarken ana tesis belirli sayıda üretilen koylara bölünmektedir. Bundan sonra ise belirli sayıdaki koylara departmanlar sırasıyla yerleştirilmektedir. Her koyun alanı da içindeki departmanların alanlarının toplamına eşittir. FBS'deki ana amaç tesisi yatay veya dikey kolonlara ayırmaktır. Farklı genişliklerde bölmeler vardır ve her bir departman sadece bir bölmede bulunabilmektedir. Departmanlar başka bölmelere taşamamaktadır. Ayrıca her departman bir bölmede bulunabileceğinden, bölümün genişliği aslında içinde bulunan departmanlarının genişliğiyle

aynı olmalıdır. Burada değişen departmanların boylarıdır. Departmanların boyları istenilen alanı sağlamak için değişmektedir. FBS ile bulunan yerleşim planları genelde optimal sonuçtan uzak sonuçlar vermektedir. Çünkü FBS ile kısıtlı sayıda yerleşim planı üretilebilmektedir. Bu gösterim şeklinin bir diğer kısıtı ise bölüme yerleştirilen departmanların o bölümü tamamen kapatması gerekmektedir. Bu nedenle departmanların bir kısıtı olan en-boy oranı her departman için sağlanamama ihtimali ortaya çıkmaktadır. Bu nedenle de aslında her bölüme departmanlar en-boy oranını bozmayacak şekilde yerleştirilmelidir. Bir tesis yerleşiminde, departmanların dikdörtgen şeklinde olduğu varsayılırsa FBS muhtemel her yerleşimi verememektedir [6].

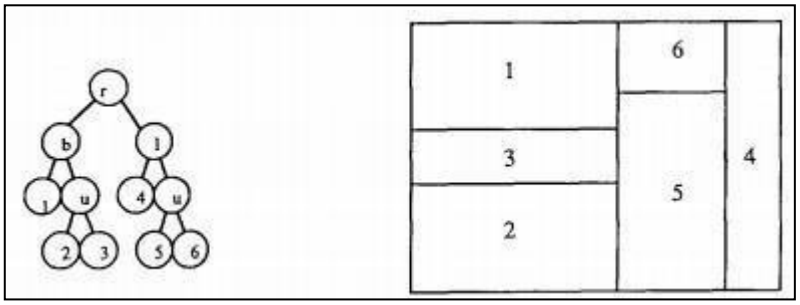


Şekil 2.2. Örnek FBS gösterimi [6]

Şekil 2.2'deki örnekten de görüleceği gibi burada tesis 3 farklı bölmeye bölünmüştür ve departmanlar sadece bir bölmeye ait olacak şekilde yerleştirilmiştir.

2.2.3. Slicing Tree Structure (STS)

İlk tanımı Tam tarafından 1992 yılında yapılmıştır [7]. Burada yerleşimi ilk olarak dilimleme yapısında göstermiştir. Dilimleme yapısının bir farklı gösterimi ise dilimleme ağacıdır (STS). Devamlı bölme işlemlerini gösteren bir ikili ağaçtır. İçteki her nod kesimin nasıl yapılacağını gösterir. Her içteki noda nasıl kesileceğinin anlaşılması için bir harf atanır (u,b gibi). STS'de kesimler önemlidir. Genelde ya 2 (aşağı, yukarı) ya da 4 (aşağı,yukarı,sağ,sol) dilimleme şekilleri mevcuttur. Her yaprağa ise özel bir yerleşke numarası atanır. Bu durumda her yaprak bir yerleşkeyi ifade etmektedir.



Şekil 2.3. Örnek STS (Karışık FBS) gösterimi [7]

Şekil 2.3'deki örnekte sol tarafta dilimleme ağacının kodlanmasıdır. Sağ tarafta ise soldaki kodlamanın ortaya çıkan yerleşimi görünmektedir. Bu durum FBS'e göre daha istenilir ve daha farklı şekiller verebilmektedir. Ancak gene STS'de oluşabilecek tesis yerleşimlerinin her departmanın en boy kısıtına uygun sonuçlar vermeme ihtimalleri vardır. Bu yerleşim biçimi bir önceki tesis yerleşim gösterimi olan FBS gibi departmanları herhangi bir bölmeye sıkıştırma kısıtı olmamasından dolayı FBS'ye göre daha farklı şekillerde sonuç verip böylelikle de yüksek ihtimalle daha düşük amaç fonksiyonu değerleri verebilmektedir. STS'de tesis yerleşimi oluştururken her departmanın yeri kromozomlarla gösterilmektedir. Kumarbazın çöküşü adı verilen ve 5 kuraldan oluşan bir yapı kullanılarak oluşturulan yerleşim departmanların en boy kısıtını bozmayacak şekilde verilebilir. Böylelikle verilen kodlamanın sonucunda ortaya çıkan tesis yerleşimindeki departmanlar her seferinde en-boy kısıtına uyan sonuçlar verebilmektedir [8].

2.3. Kullanılan Amaç Fonksiyonları

Eşit olmayan alanlı tesis yerleşim modelleri hakkındaki makalelerde, hepsi aynı amaç fonksiyonunu kullanmamaktadır. Meller et al [9], Tam [7] ve Meller [10] makalelerinde materyallerin akış toplamını en küçüklemek amaç fonksiyonu olarak belirlemişlerdir. Hassan [2] 'de graf çözümü yapılırken amaç fonksiyonu olarak bitişiklik ele alınmıştır. Bu makaledeki amaç ilişki tablosundaki kullanarak departmanların arasındaki ilişkinin en iyilenmesi olarak ele alınmıştır.

2.4. Kullanılan Kısıtlar

Montreuil [3] 'in ilk MIP modelinden sonra modelde geçen kısıtlarla ilgili birçok makale daha yazılmıştır. Bu makalelerin genel konusu bu ilk modelde geçen doğrusal olmayan kısıtlamalardır. Bu alanda en çok uğraşılan durum ise departmanların alan kısıtlaması olmuştur. Montreuil modellerken alan kısıtlamasını doğrusallaştırmak için "sınırlandırılmış çevre kısıtlaması"nı kullanmıştır. Formül şu şekildedir:

$$p_i \leqslant 4(l_i^x + l_i^y) \leqslant P_i, \quad \forall i.$$

Burada

$p_i = 4\sqrt{a_i}$

$P_i = 2\sqrt{a_i}(1+\alpha_i)/\sqrt{\alpha_i}$: göstermektedir.

a_i departman i'nin alanını, α_i ise her departman için izin verilen en-boy oranını göstermektedir.

Meller[10] yazdığı makalede Montreuil'in "sınırlandırılmış çevre kısıtlamasının gerçek alana kısıtlamada pek de başarılı olmadığını belirtmiştir. Hatta α_i: 2,3,4 ve 5 değerleri için alanın %11, %25, %36 ve %44 daha düşük çıktığının gözlemlendiği belirtilmiştir.

Bu nedenle Meller yeni bir alan kısıtlaması olarak "vekil alan kısıtlaması" önermiştir. Bu kısıtlama modele sadece n tane yeni değişken eklemektedir. Ve kısıtlama aşağıdaki gibidir:

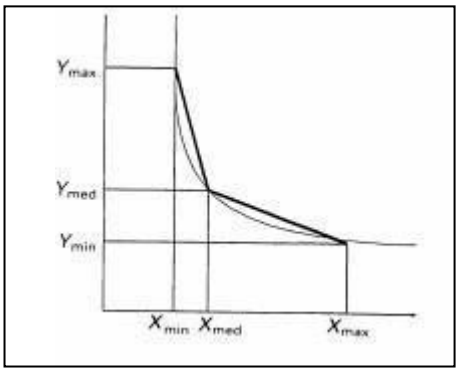
$$4(l_i^x + l_i^y) \geqslant 3\sqrt{a_i} + f \times 2l_i^{\max}, \quad \forall i.$$

Sol taraf departmanın gerçek çevresini belirtirken sağ taraf ise kare departmanların en düşük çevreye sahip olduğunu modellemektedir. Kısıtlamada geçen f değeri 0.95 alındığı durumda ise bu kısıtlamanın sonuçta çıkan departman alanlarında çok düşük bir hataya sahip olacağı belirtilmiştir.

Lacksonen [11] alan kısıtlamasını parçalı doğrusallaştırma ile çözmeyi denemiştir. Bu kısıtlamayı kullanırken 4 farklı değişken tanımlamıştır.

$$(XU_i - XL_i)\,(YU_i - YL_i) = A_i \quad \forall i$$

Buradaki kısıtlama alan kısıtlamasıdır ancak bu haliyle doğrusal bir kısıt değildir. Bu durumda Lacksonen parçalı doğrusallaştırma tekniğini kullanmıştır ve alanı aşağıdaki figürdeki gibi doğrusallaştırmıştır.



Şekil 2.4. Alan doğrusallaştırması [11]

Öncelikle segmentlerin bitiş noktasını tanımlamıştır ve en-boy oranını R olarak almıştır.

$$Y_{max} = X_{max} = (AR)^{0.5}$$
$$Y_{med} = X_{med} = A^{0.5}$$
$$Y_{min} = X_{min} = (A/R)^{0.5}.$$

Burada iki yeni değişken eklemiştir (X_{1i} ve X_{2i}). Burada sırasıyla soldaki ve sağdaki X segmentinin uzunlukları belirtilmiştir. Bu durumda ise yeni uzunluk ve boy ölçüleri aşağıdaki gibi olmuştur.

$$YU_i - YL_i = Y_{max} + a_1 X_{1i} + a_2 X_{2i}$$
$$XU_i - XL_i = X_{min} + X_{1i} + X_{2i}$$
$$X_{1i} \leqslant X_{med} - X_{min}$$
$$X_{2i} \leqslant X_{max} - X_{med}.$$

a_1 ve a_2 eğimleri R cinsinden yazılabilirler.

$$a_1 = -\sqrt{R}$$
$$a_2 = -1/\sqrt{R}.$$

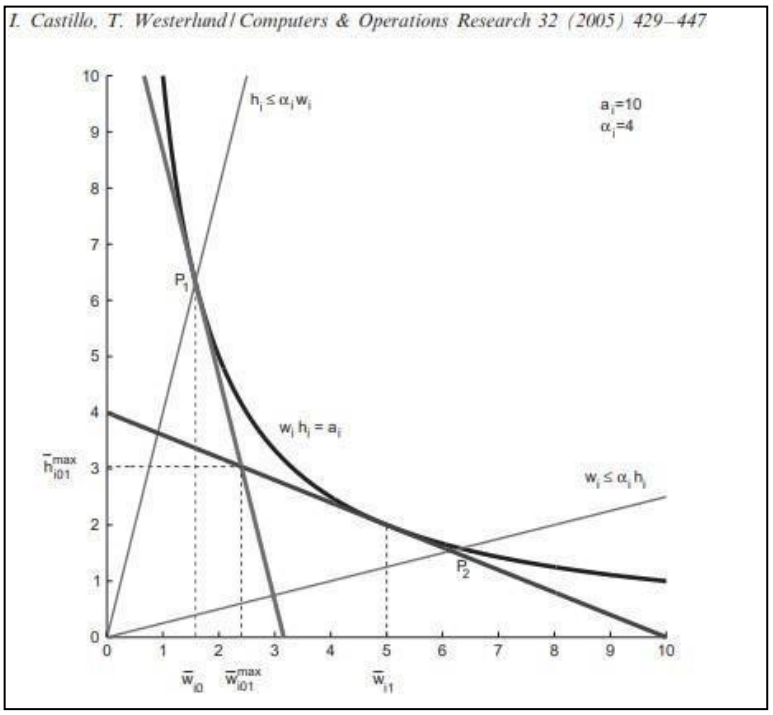
Son durumda ise A(alan) ve R(en-boy oranı) kullanıcı tarafından girildiğinde alan kısıtlamaları aşağıdaki gibi olmaktadır.

$$YU_i - YL_i + R^{0.5} X_{1i} + R^{-0.5} X_{2i} = (A_i R)^{0.5}$$
$$XU_i - XL_i - X_{1i} - X_{2i} = (A_i/R)^{0.5}$$
$$X_{1i} \leqslant A_i^{0.5}(1 - R^{-0.5})$$
$$X_{2i} \leqslant A_i^{0.5}(R^{0.5} - 1).$$

Lacksonen bu kısıtların R=2 olduğunda sadece %0 ve %3 aralığında farklı sonuç verdiğini belirtmiştir. Eğer %3 hata aralığı yeterli değil ise 4 segmentli doğrusallaştırma yapılabileceği ancak bu doğrusallaştırma 2 tane daha değişken eklenerek yapılması gerektiği belirtmiştir.

Bu yaklaşımlardan sonra Castillo et al [12] bir ε doğruluklu doğrusallaştırma yaklaşımı gerçekleştirmişlerdir. Bunun nedeni ise alan kısıtının hiperbolik ve konkav olması ve daha önce yapılan çalışmaların bu kısıtı yeterli kadar yaklaşamamasıdır. Ayrıca mevcut olan doğrusal olmayan çözücüler ise küresel optimalliği garanti edememektedir. Bu durumda alan için kesim düzlemi yaklaşımı önermektedirler ve bu yaklaşım Bulunan alan

en-boy oranından bağımsız olarak optimallikte sadece %ε kadar yanlışlıkla alanı vermektedir.



Şekil 2.5. Oluşturulan örnek kesim düzlemleri [12]

Aşağıda son sonuçtaki alanların % ε kadar hatalı olacak bir gösterimi verilmektedir. Buradaki ε sayısı kullanıcı tarafından belirlenmektedir.

$$1 - \frac{w_i h_i}{a_i} \leqslant \varepsilon, \quad i = 1, \ldots, N.$$

$$w_i h_i = a_i, \quad \frac{w_i}{h_i} \leqslant \alpha_i, \quad \frac{h_i}{w_i} \leqslant \alpha_i, \quad w_i^{\text{low}} \leqslant w_i \leqslant w_i^{\text{up}}, \quad h_i^{\text{low}} \leqslant h_i \leqslant h_i^{\text{up}},$$

Burada, $w_i^{\text{up}} = \min\{\sqrt{a_i \alpha_i}, w_{\text{F}}\}$, $h_i^{\text{up}} = \min\{\sqrt{a_i \alpha_i}, h_{\text{F}}\}$, $w_i^{\text{low}} = a_i / h_i^{\text{up}}$, and $h_i^{\text{low}} = a_i / w_i^{\text{up}}$

Konveks alt sınır olan $-h_i + a_i / w_i \leqslant 0$, 'dan $\bar{w}_{i0}$ noktasında konveks alt sınır olan

$$-h_i - \frac{a_i}{\bar{w}_{i0}^2} w_i \leqslant -2 \frac{a_i}{\bar{w}_{i0}},$$ oluşturulabilir.

$\bar{w}_{i0} = w_i^{\text{low}}$ noktasında başlayan kesim düzlemi ve onu takip eden $\bar{w}_{i0} < \bar{w}_{i1} \leqslant w_i^{\text{up}}$, noktasında maksimum alan kısıtının ihlali kesişim noktaları olan $(\bar{w}_{i01}^{\max}, \bar{h}_{i01}^{\max})$ noktada gerçekleşir ve bu

$$\bar{w}_{i01}^{\max} = 2\,\frac{1/\bar{w}_{i1} - 1/\bar{w}_{i0}}{1/\bar{w}_{i1}^2 - 1/\bar{w}_{i0}^2}, \tag{4.1}$$

$$\bar{h}_{i01}^{\max} = -\frac{a_i}{\bar{w}_{i0}^2}\,\bar{w}_{i01}^{\max} + 2\,\frac{a_i}{\bar{w}_{i0}} \tag{4.2}$$

İle gösterilmektedir.

$$\varepsilon_{i01}^{\max} = 1 - \bar{w}_{i01}^{\max}\bar{h}_{i01}^{\max} \Big/ a_i = \left(\frac{\bar{w}_{i1} - \bar{w}_{i0}}{\bar{w}_{i1} + \bar{w}_{i0}}\right)^2, \tag{4.3}$$

Alan kısıt ihlali $\bar{w}_{io}$'dan $\bar{w}_{i01}^{max}$'a kadar monoton olarak artar ve tam tersi durumda da monoton olarak azalır.

Bu durumda eğer maksimum alan kısıtlaması olan ε belirtilirse, (4.3) numaralı kısıt $\bar{w}_{i1}$ ve $\bar{w}_{io}$ arasında kesim düzlemi oluşturmak için yeterli bilgiyi vermektedir. (4.3) numaralı kısıtı kullanarak aşağıdaki eşitliği elde edebiliriz.

$$\frac{\bar{w}_{i1}}{\bar{w}_{i0}} = \frac{1 + \sqrt{\varepsilon}}{1 - \sqrt{\varepsilon}}. \tag{4.4}$$

$\bar{w}_{i1}$ ve $\bar{w}_{io}$'ın sahip olduğu maksimum hata £'yi bulduktan sonra $\bar{w}_{i2}$'den $\bar{w}_{i1}$'e maksimum hatası £ olan diğer noktaları da bulmak mümkündür. C_i'nin w_{ilow}'dan w_{iup}'a kadar kadar toplam alt-aralık sayısını belirttiğini varsayalım. C_{i+1} bir departmanın oluşturabileceği toplam kesim düzlemi sayısıdır.

$$\frac{\bar{w}_{iC_i}}{\bar{w}_{i0}} = \frac{w_{i1}}{w_{i0}}\,\frac{w_{i2}}{w_{i1}}\cdots\frac{w_{iC_i}}{w_{i,C_i-1}} = \left(\frac{\bar{w}_{i1}}{\bar{w}_{i0}}\right)^{C_i} = \left(\frac{1 + \sqrt{\varepsilon}}{1 - \sqrt{\varepsilon}}\right)^{C_i}. \tag{4.5}$$

En-boy oranından bağımsız olarak departmanın son alanının %£ hata ile tanımlanması için her departman için gereken kesim düzlemi sayısını önceden bulabiliriz.
Kısıt (4.5)'den alttakini elde edebiliriz.

$$C_i = \left\lceil \frac{\ln(w_i^{\text{up}}/w_i^{\text{low}})}{\ln((1 + \sqrt{\varepsilon})/(1 - \sqrt{\varepsilon}))} \right\rceil, \tag{4.6}$$

Burada parantez içindeki kısım ifadenin sayısal değerinin üst tamsayısıdır. Kesim düzlemlerinin oluşturulma noktaları aşağıdaki gibi hesaplanabilir.

$$-h_i - \frac{a_i}{\bar{w}_{ik}^2} w_i \leqslant -2\frac{a_i}{\bar{w}_{ik}}, \quad k = 0, \ldots, C_i. \tag{4.7}$$

$$w_{i0} = w_i^{low} \tag{4.8}$$

Son olarak ise her departman için önceden oluşturulacak Ci+1 sayıda kesim düzlemi, yani Konveks olmayan ve hiperbolik alan kısıtının ε-doğruluklu gösterimi aşağıdaki gibidir:

$$\bar{w}_{ik} = \bar{w}_{i,k-1}\left(\frac{w_i^{up}}{w_i^{low}}\right)^{1/C_i}, \quad k = 1, \ldots, C_i, \tag{4.9}$$

(4.9) numaralı kısıt tam olarak doğrusaldır ve CPLEX gibi standart algoritma kullanılan ve her yerde ulaşılabilir olan uygulamalar kullanılabilir. Ve Lacksonen [11]'in kullandığı parçalı yaklaşım aksine fazladan değişkenlere ihtiyaç bulunmamaktadır.

2.5. Kullanılan Veri Setleri

Bu problemdeki zorluklardan bir tanesi de kıstas olarak kullanılabilecek veri setlerinin az olmasıdır. Aşağıdaki tabloda literatürde en çok kullanılan 9 adet veri setinin tablosu bulunmaktadır. Buradaki akış sayıları çoğunlukla Nugent [13]'den alınmış olup buradaki değerler, alanlar ve departman sayıları değiştirilerek literatürde daha kolay kullanılabilir hale getirilmişlerdir. Tabi ki bu problemler belirli kısıtları değiştirerek değiştirebilir.

Tablo 2. 1. Sıklıkla kullanılan veri setleri

			Departman boyutları			
Numara	**Problem Veri seti**	**Departman Sayısı**	**En**	**Boy**	**Şekil kısıtı**	**Mesafe gösterimi**
1	O7	7	8,54	13	α = 4	Doğrusal
2	O8	8	11,31	13	α = 4	Doğrusal
3	O9	9	12	13	α = 4	Doğrusal
4	VC10E-a	10	25	51	α = 5	Eüklidyen
5	VC10R-a	10	25	51	α = 5	Doğrusal
6	Ba12	12	6	10	Lmin=1	Doğrusal
7	Ba14	14	7	9	Lmin=1	Doğrusal
8	AB20-ar5	20	2	3	α = 5	Doğrusal
9	AB20-ar4	20	2	3	α = 4	Doğrusal

2.6. Çözüm Yöntemleri

Bazı makaleler eşit olmayan alanlı departmanların yerleşim problemini kesin sonuç olarak çözmeyi denemiştir. İlk olarak Lawler [14] çalışmasında ikinci dereceden atama problemini fazladan kısıtlarla formulüze edilmesini göstermiştir. Daha sonrasında Bazaara et al [15] ikinci dereceden atama problemine yeni bir formülasyon sunmuştur. Ortaya çıkan problem 0-1 doğrusal tamsayılı bir problemdir. Daha sonrasında Montreuil [3] alan kısıtını çevresine kısıt koyarak doğrusallaştırmaya çalışmıştır. Bundan sonra Lacksonen [11] parçalı doğrusallaştırma yaparken fazladan 2 ikili değişken ekleyerek bu hataları minimize etmeye çalışmıştır. Al-Khayyal et al. [16] ise gerektiğinde satır üreten dallandırma-kesme işlemi ile dinamik olarak gerçek alan kısıtında tanjantal destekler oluşturan bir teknik geliştirmişlerdir. Meller et al ise [9]'da yazdığı makalesinde her departman için birer gerçek değişken eklemiştir. Kesin çözüm çalışmalarının en sonunda ise Westerlund [12] ise optimal olarak son departman alanlarının yüzde ε -farklı olacak şekilde sonuç verecek bir doğrusal ε-kesinlikli gösterim geliştirmişlerdir. Şu anda geliştirilen Karmaşık tamsayılı doğrusal modellerin sadece doğrusal olmayan alanları belirli bir hata oranında doğru verdiği belirtilebilir.

Bazı makalelerde yaklaşık sonuç bulmak için yaklaşımlar kullanılmıştır. Aşağıdaki makalelerde sezgisel yöntemlerden örnekler verilmektedir. İnşa etme algoritmaları tesis yerleşim problemleri için uygulama ve konsept açısından en basit sezgisel yöntemlerdendir. Bu algoritma sırasıyla departmanların seçilmesi ve yerleştirilmesiyle tam bir tesis yerleşiminin elde edilmesini amaçlamaktadır. En çok bilinen algoritmalardan bazıları: HC66 [17], ALDEP [18], CORELAP [19] ve SHAPE [20] olarak belirtilebilir. Bu algoritmalardan HC66 en çok alıntılanan ve herhangi bir inşa algoritmasının alt sınırı belirlemek için kullanılan algoritmadır. Bu algoritmanın basitliğinden dolayı belirtilen sonuçlarda genelde kalite olarak başarısız sonuçlar vermektedir. Ancak bu algoritmadan elden edilen tesis yerleşim örneklerini geliştirme algoritmalarının başlama kısmında kullanılabilmektedir.

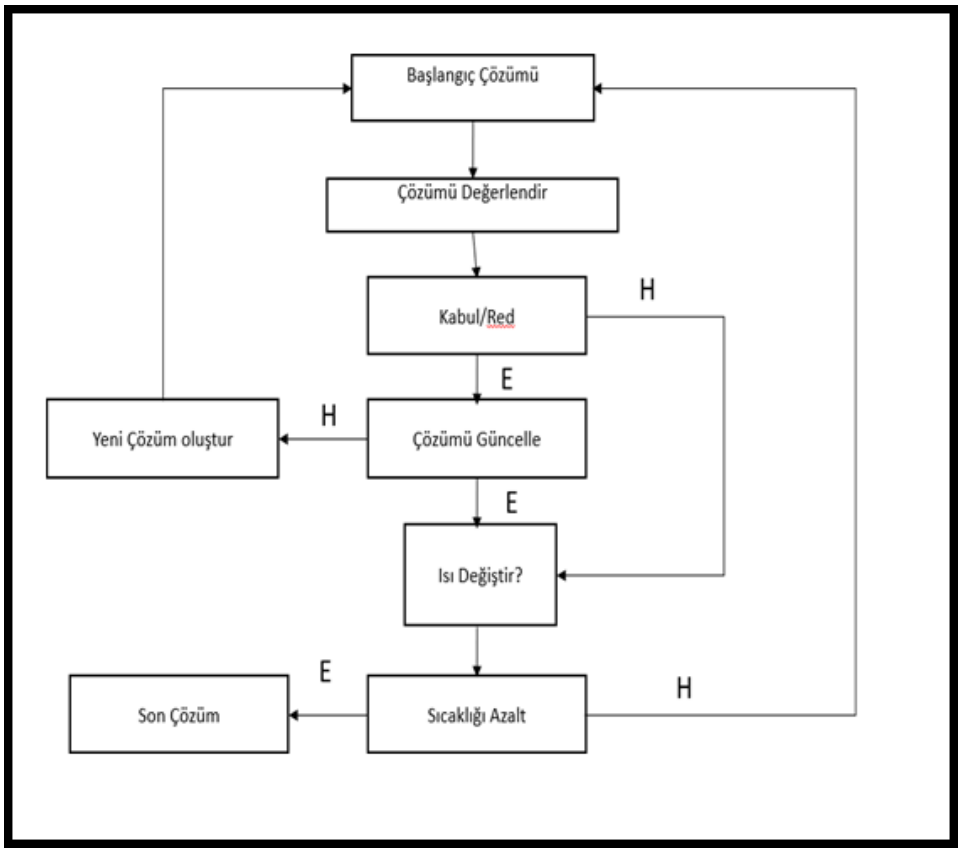
Geliştirme algoritmaları başlangıç tesis yerleşim planı ile başlamaktadır. Sonrasında halihazırdaki yerleşim planı ikili departmanların yerlerinin değiştirilmesiyle farkı planlar elde edilir. En iyi sonucu veren değişim saklanır ve bu prosedür daha iyi bir sonuç vermeyene kadar veya durma kriteri gerçekleşene kadar devam eder. Bu durumda geliştirme algoritmalarını genellikle ilk elde edilen tesis yerleşim planına bağlıdır. CRAFT Armour et al [21] tarafından geliştirilmiştir. CRAFT aynı zamanda bilgisayarlaştırılmış ilk algoritma olarak bilinmektedir. H63 [22], COFAD [23], LOGİC [7] ve MULTIPLE [24] bunlardan bazılarıdır. Bu algoritmanın negatif yanı ise genellikle bir yerel optimal noktada durmasıdır.

Sezgi ötesi algoritmalarının geliştirilmesi geliştirme algoritmalarının performansını ciddi bir şekilde arttırmıştır. Tesis yerleşim problemlerinde genellikle kullanılan algoritmalar Benzetilmiş tavlama (SA), Tabu Araması(TS) ve Genetik algoritmadır(GA).

2.6.1 Benzetilmiş tavlama

Verilen bir fonksiyonun küresel en iyi noktasını bulabilmek için kullanılan bir olasılıksal tekniktir. Kirkpatrick [25] araştırmasında istatistiksel mekanik ile çok değişkenli eniyileme arasında anlamlı bir ilişki olduğundan bahsetmektedir. Benzetilmiş tavlamayı Gezen satıcı probleminde kullanmıştır. Burkard [26] araştırmasında sadece değişim esnasında amaç fonksiyonunun azalmasını değil aynı zamanda belli koşullarda artmasına rağmen değişimi gerçekleştirmişlerdir. Oldukça kısa zamanda yarı optimal sonuçlar elde etmişlerdir. Matai et al [27] araştırmalarında modifiye Benzetilmiş tavlama algoritmasını kullanmışlardır. Bu algoritmayı çok amaç fonksiyonlu tesis yerleşiminde kullanmış ve daha önceki araştırmalardan daha verimli sonuçlar elde etmişlerdir. Benzetilmiş Tavlama tekniği

aslında son yıllarda popülerliğini kaybetmiş ve başka evrimsel algoritma türleri literatürde sıklıkla kullanılmaya başlamıştır.

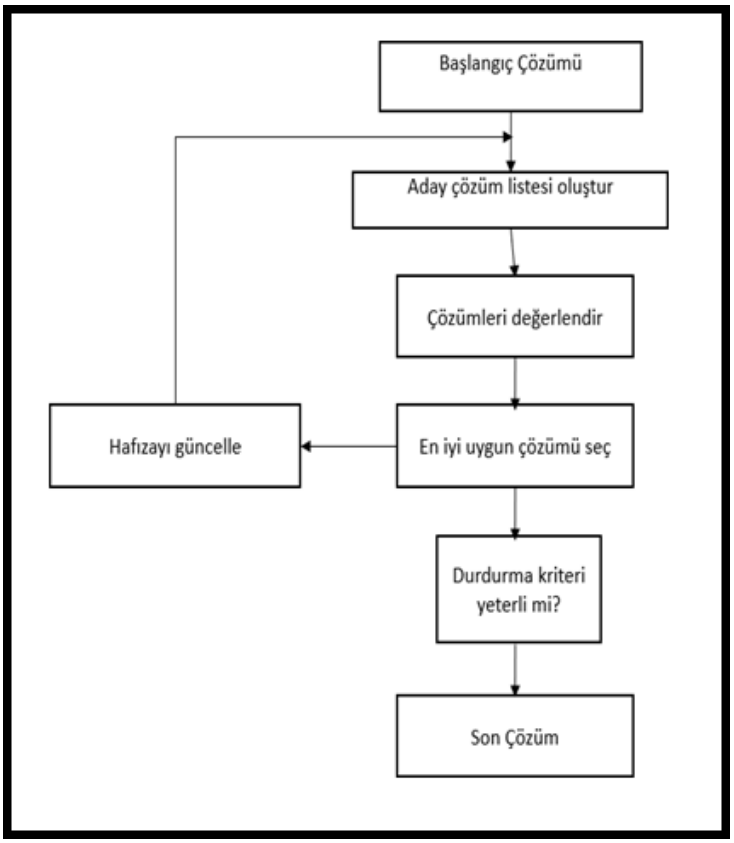


Şekil 2.6. Benzetilmiş tavlama akış şeması

2.6.2 Tabu Araması

Yerel arama metodunu kullanarak matematiksel en iyileme için kullanılan bir arama türüdür. Burada Tabu kelimesi önceden bakılan yerlere tekrar bakmamaya cesaretlendirme anlamına gelmektedir. Glover [28] ilk olarak Tabu aramasını kullanmıştır. Bu aramada amaç yerel en iyi noktadan bazı hareketleri kısıtlayarak çıkmaktır. Skorin-Kapov [29] Tabu aramasının bir varyasyonunu ikinci dereceden atama problemini çözmek için kullanmıştır. Problemdeki değişken sayısı arttıkça doğru sayılabilecek tabu arama listesinin de artacağını belirtmiştir. Bundan sonra Chiang et al [30] araştırmalarında Tabu aramasını komşu bazlı departmanların yer değiştirilmesi tekniği kullanarak yapmıştır. Araştırmalarında uzun süreli hatıra yapısı, dinamik tabu listesi büyüklüğü ve ayrıştırma stratejilerinden bahsetmişlerdir.

Scholz et al [31] araştırmasında kesilmiş ağaç yapısını tabu araması ile birlikte kullanmışlardır. Bu ağaç yapısının kullanım artısı ise hem belirli boyutlarda departmanlarda çözüm bulması hem de değişik alanlı departmanlara çözüm bulmasıdır.

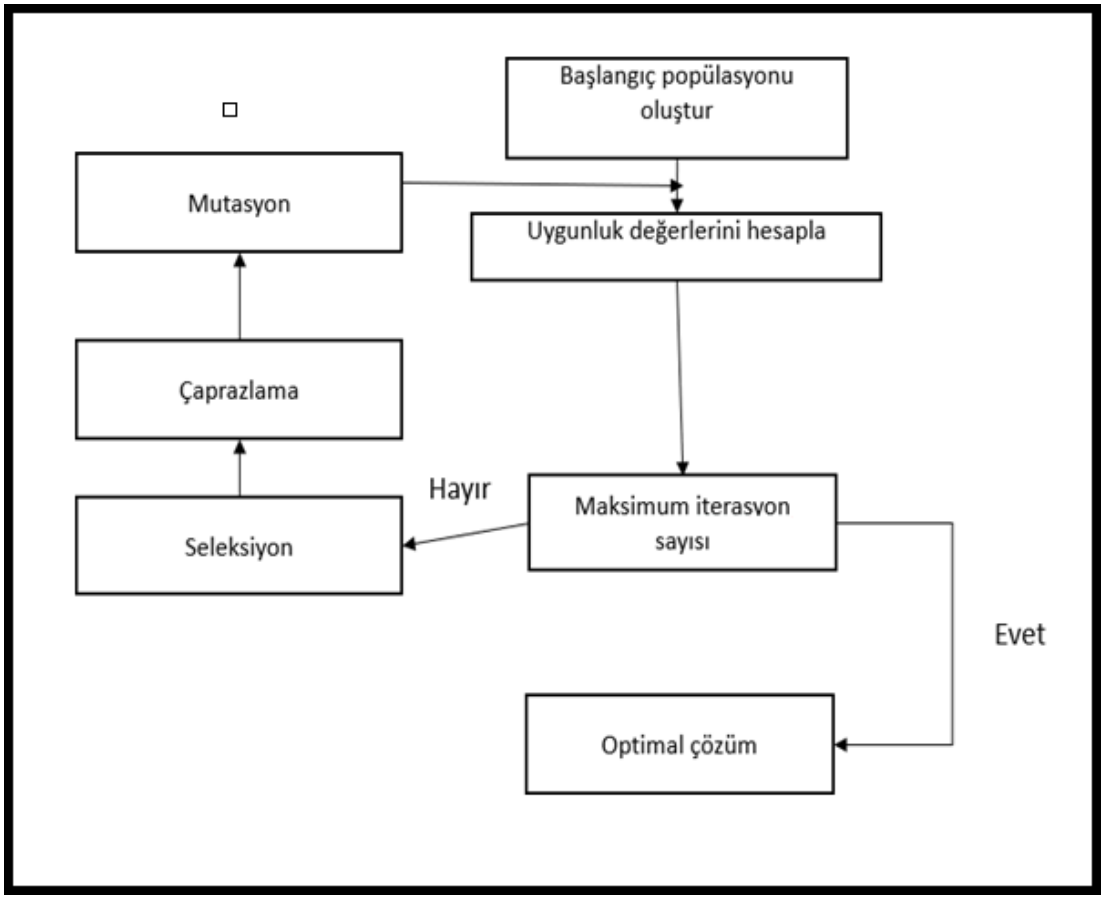


Şekil 2.7. Tabu araması akış şeması

2.6.3. Genetik Algoritması

Darwin'in en sağlıklının hayatta kalmasında belirtilen rekabetçi güçleri kullanarak bir popülasyonda evrim sürecini gerçekleştirmektedir. Evrimsel algoritmalardan birisidir. İlk olarak Tam [7] araştırmasında Benzetilmiş tavlamanın zayıf yönlerinden kurtulmak için genetik algoritmayı kullanmıştır ve 12-30 departmanlı örneklemler için umut verici sonuçlar elde etmiştir. Balakrishnan [4] araştırmasında ilk başta popülasyonu oluştururken iki farklı yöntem kullanmıştır (rastsal method ve Urban'ın prosedürü). Sonrasında ise genetik algoritmada yapılan mutasyon CRAFT metoduyla gerçekleştirilmiştir. Bu araştırmayı dinamik tesis yerleşim probleminde uygulamıştır. Shayan et al [8] araştırmasında alan kısıtlamalarını ve her departmanın en boy oranını sağlayabilecek kromozomların

tanımlanma gerekliliğinin zorluluğundan bahsedip, kesim ağacını, kumarbazın çöküşü yöntemi ile birleştirip her seferinde kısıtlara uygun departman yerleşimi veren bir genetik algoritma kullanmışlardır. Sonuçlar 100 departmanlı problemlerde bile önemli sürede sürmemiştir. Aiello [32] araştırmasında Çok amaçlı genetik algoritma ile eşit olmayan alanlı tesis yerleşim problemini ele almışlardır. Bu çalışmada önemli fark kesilmiş ağaç yapısını kullanmalarıdır. Çoklu karar verme kriterleri ise materyal taşıma maliyeti, en-boy oranı ve yakınlıktır. Gonçalves et al [33] araştırmasında Önyargılı rastsal genetik algoritma kullanmışlardır. Bu algoritmayı 28 farklı mihenk taşı araştırmaya uyarlamışlardır ve bu 28 veri setinden 19'unda daha iyi sonuç elde etmişlerdir. Görüleceği üzere sezgi ötesi algoritmalardan en meşhuru ve üzerinde çalışılanı genetik algoritmadır. Bu tip algoritmayı kullanarak literatürde birçok çalışma gerçekleştirilmiştir.



Şekil 2.8. Genetik algoritma akış şeması

3. TESİS YERLEŞİMİNDE EVRİMSEL ALGORİTMA İNCELENMESİ

3.1. Genel Kavramlar

Son yıllarda, ekonomi, politika, yönetim ve mühendislik gibi farklı alanlardan kaynaklanan gerçek hayattaki karmaşık problemleri çözmek için sezgi ötesi algoritmalar kullanılmaktadır. Yoğunlaştırma ve çeşitlendirme, sezgi ötesi algoritmanın temel unsurlarıdır. Gerçek hayat problemini etkili bir şekilde çözmek için bu unsurlar arasındaki uygun denge gereklidir. Sezgi ötesi algoritmaların çoğu biyolojik evrim sürecinden, sürü davranışından ve fizik yasalarından esinlenmiştir. Bir koloni tarafından çözülen yemek bulma, yuvalarını genişletme ya da inşa etme problemleri iş gücü olarak bireyler arasında eşit olarak dağıtılmıştır. Bu problemlerin birçoğunun mühendislik ve bilgisayar bilimlerinde karşılıkları vardır. Sosyal böceklerin en önemli özelliklerinden birisi ise bu problemleri oldukça esnek ve kuvvetli şekilde çözebilmeleridir: esneklik onlara çevrelerine uyum sağlama gücü verirken güçlülük ise kolonin bazı elemanlarının görevlerini yapamama durumunda dahi işlemesini bahsetmektedir [34]. Bu algoritmalar genel olarak tek çözüm ve popülasyon tabanlı sezgi ötesi algoritma olmak üzere iki kategoriye ayrılır. Tek çözüm tabanlı sezgi ötesi algoritmalar, tek aday çözümü kullanır ve bu çözümü yerel aramayı kullanarak geliştirir. Ancak, tek çözüm tabanlı sezgi ötesi yöntemlerden elde edilen çözüm, yerel optimumda sıkışıp kalabilir. İyi bilinen tek çözüm tabanlı sezgi ötesi yöntemler, benzetilmiş tavlama, tabu arama (TS), mikrokanonik tavlama (MA) ve yönlendirilmiş yerel aramadır (GLS). Popülasyon tabanlı meta-sezgisel arama işlemi sırasında, birden çok aday çözümü kullanır. Bu meta-sezgiseller, popülasyondaki çeşitliliği korur ve çözümlerin yerel optimumlara takılmasını önler. İyi bilinen popülasyon tabanlı sezgi ötesi algoritmalardan bazıları, genetik algoritma (GA), parçacık sürü optimizasyonu (PSO), karınca kolonisi optimizasyonu (ACO)

, benekli sırtlan optimize edici (SHO) , imparator penguen optimize edici (EPO) ve martı optimizasyonudur (SOA) . Sezgi ötesi algoritmaları arasında Genetik algoritma (GA), biyolojik evrim sürecinden esinlenerek yapılan ve tanınmış bir algoritmadır. GA, Darwin'in doğada en güçlü olanın hayatta kalması teorisini taklit eder. Genetik algoritma, J.H. Holland tarafından 1992'de tanıtılmıştır. Genetik algoritmanın temel unsurları kromozom temsili, uygunluk seçimi ve biyolojik-ilham operatörleridir.

3.2. Evrimsel Operatörler

3.2.1. Kodlama şemaları

Hesaplama problemlerinin çoğu için, kodlama şeması (yani, belirli bir biçimde dönüştürmek için) önemli bir rol oynar. Kodlama şemaları, problem alanına göre farklılaştırılır. İyi bilinen kodlama şemaları ikili, sekizli, onaltılı, permütasyon, değer tabanlı ve ağaçtır. İkili kodlama, yaygın olarak kullanılan kodlama şemasıdır. Her gen veya kromozom,1 veya 0 dizisi olarak temsil edilir. İkili kodlamada, her bit çözümün özelliklerini temsil eder. Çaprazlama ve mutasyon operatörlerinin daha hızlı uygulanmasını sağlar. Ancak, ikili forma dönüştürmek için ekstra çaba gerektirir ve algoritmanın doğruluğu ikili dönüştürmeye bağlıdır. Bit akışı soruna göre değiştirilir. İkili kodlama şeması, epistasis ve doğal temsil nedeniyle bazı mühendislik tasarım problemleri için uygun değildir [36]. Sekizli kodlama şemasında, gen veya kromozom sekizli sayılar (0-7) şeklinde temsil edilir. Onaltılık kodlama şemasında, gen veya kromozom onaltılık sayılar (0-9, A-F) biçiminde temsil edilir. Permütasyon kodlama şeması genellikle sıralama problemlerinde kullanılır. Bu kodlama şemasında, gen veya kromozom, bir dizideki konumu temsil eden sayı dizisi ile temsil edilir. Değer kodlama şemasında, gen veya kromozom, bazı değerler dizisi kullanılarak temsil edilir. Bu değerler gerçek, tam sayı veya karakter olabilir [37]. Bu kodlama şeması, daha karmaşık değerlerin kullanıldığı problemlerin çözümünde yardımcı olabilir. Bu tür problemlerde ikili kodlama başarısız olabilir. Çoğunlukla optimal ağırlıkları bulmak için nöral (sınır) ağlarında kullanılır.

Ağaç kodlamasında, gen veya kromozom, bir işlevler veya komutlar ağacıyla temsil edilir. Bu işlevler ve komutlar herhangi bir programlama dili ile ilgili olabilir. Bu, ağaç biçiminde bastırma metodunun temsiline çok benzer. Bu tür kodlama genellikle gelişen programlarda veya ifadelerde kullanılır [38].

3.2.2. Seçim teknikleri

Seçim, belirli bir dizinin üreme sürecine katılıp katılmayacağını belirleyen genetik algoritmalarda önemli bir adımdır. GA'nın yakınsama hızı seçim basıncına bağlıdır. İyi bilinen seçim teknikleri rulet çarkı, sıra, turnuva, boltzmann ve stokastik evrensel örneklemedir.

Rulet çarkı seçimi, olası tüm dizileri, uygunluk değerlerine göre kendilerine tahsis edilen çarkın bir kısmı ile bir çark üzerine eşler. Bu çark daha sonra yeni neslin oluşumuna katılacak belirli çözümleri seçmek için rastgele döndürülür. Burada her i'inci bireye p(i) olasılıkla seçim ihtimali tanınır. Ancak, stokastik doğasının getirdiği hatalar gibi birçok sorundan zarar görmektedir. Burada en önemli dezavantajlarından birisi dominant bir bireyin olma ihtimali ve seçimde onun seçilecek olmasıdır [38]. De Jong et al [39], seçim prosedüründe determinizm kavramını getirerek hataları ortadan kaldırmak için rulet tekerleği seçim yöntemini değiştirdi. Sıra seçimi, Rulet çarkı seçiminin değiştirilmiş şeklidir.Uygunluk değeri yerine sıraları kullanır. Sıralamalar uygunluk değerlerine göre verilir, böylece her bireye kendi sıralarına göre seçilme şansı verilir. Sıra seçim yöntemi, çözümünyerel bir minimuma zamanından önce yaklaşma şansını azaltır

Turnuva seçim tekniği ilk olarak 1983 yılında Brindle [40] tarafından önerilmiştir. Bireyler, çiftler halinde stokastik rulet çarkından uygunluk değerlerine göre seçilir. Seçimden sonra uygunluk değeri daha yüksek olan bireyler yeni nesil havuzuna eklenir

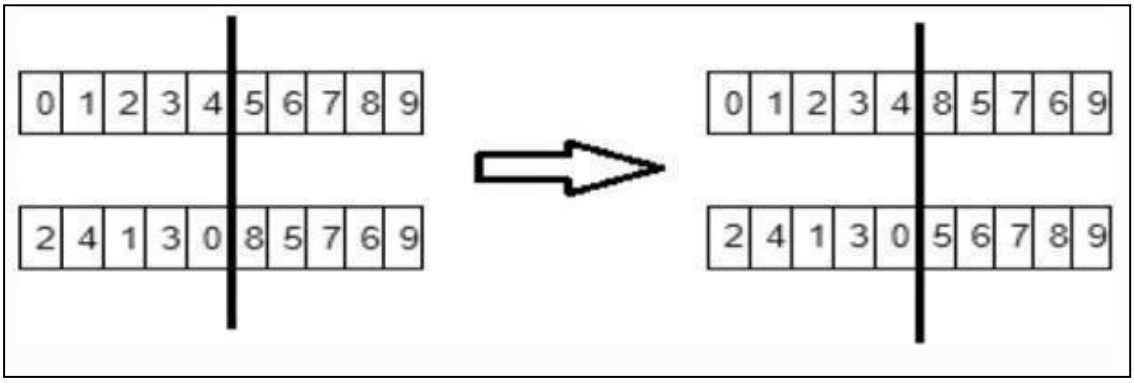
Bu seçim yönteminde, her bir birey, eğer nihai çözüm popülasyonuna ulaşırsa, diğer tüm n-1 bireylerle karşılaştırılır. Stokastik evrensel örnekleme (SUS), mevcut rulet tekerleği seçim yönteminin bir uzantısıdır. Bir nesilden bireyler listesinde rastgele bir başlangıç noktası kullanır ve yeni bireyi eşit aralıklarla seçer. Gelecek nesil için çapraz geçişe katılmak üzere seçilmek için tüm bireylere eşit şans verir [38].

Boltzmann seçimi, Monte Carlo Simülasyonunda kullanılan entropi ve örnekleme yöntemlerine dayanmaktadır. Erken yakınsama sorununun çözülmesine yardımcı olur. Entropi örneklemesi, evrimi daha düşük konfigürasyonların seçilmesine yönlendirir. Bu şekilde, seçilecek nadir konfigürasyonların oranı bol olanlardan daha yüksektir. Çok daha kısa sürede yürütülürken, en iyi diziyi seçme olasılığı çok yüksektir. Bu nedenle, bu iki örnekleme yönteminin aynı anda kullanılmasıyla yavruların kabulünde, kişi sadece en azından yerel minimumlar arasındaki yüksek engelleri aşmayı değil, aynı zamanda küresel veya neredeyse küresel maksimumu aramaya devam etmeye yardımcı olabilir. Ancak, bilgi kaybı olasılığı vardır [41]. Elitizm yoluyla yönetilebilir. Elitizm seçimi, Rulet çarkı seçiminin performansını iyileştirmek için K.D. Jong [38] tarafından önerildi. Bir nesildeki seçkin bireyin her zaman bir sonraki nesile yayılmasını sağlar. Normal seçim prosedüründen sonra en yüksek uygunluk değerine sahip birey bir sonraki nesilde mevcut değilse, o zaman elitist olan da bir sonraki nesile otomatik olarak dahil edilir.

3.2.3. Çaprazlama operatörleri

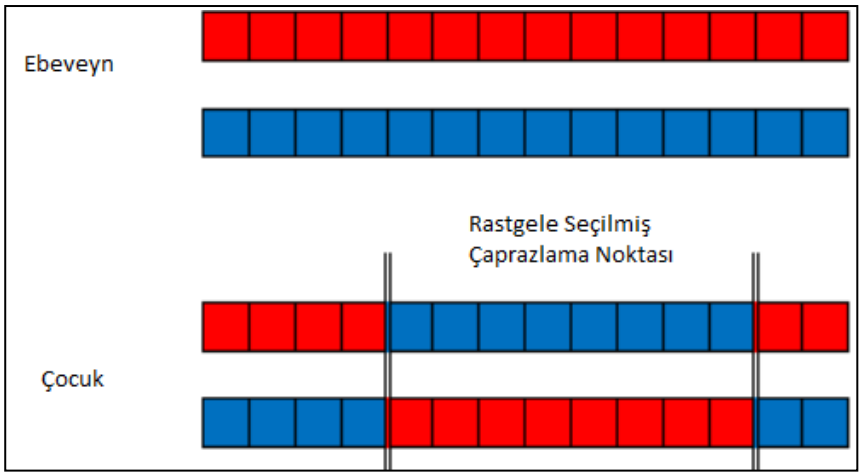
Çaprazlama operatörleri, iki veya daha fazla ebeveynin genetik bilgilerini birleştirerek yavruları oluşturmak için kullanılır. İyi bilinen çaprazlama operatörleri, tek noktalı, iki noktalı, k-noktalı, tek biçimli, kısmen eşleştirilmiş, sıralı, önceliği koruyan çaprazlama, karıştırma, indirgenmiş vekil ve döngüdür.

Tek noktalı çaprazlamada rastgele bir çaprazlama noktası seçilir. Bu noktayı aşan iki ebeveynin genetik bilgileri birbiriyle değiştirilir. Şekil 3, takastan sonraki genetik bilgiyi gösterir. Yeni yavruları elde etmek için her iki ebeveynin kuyruk dizisinin bilgilerini değiştirmektedir.



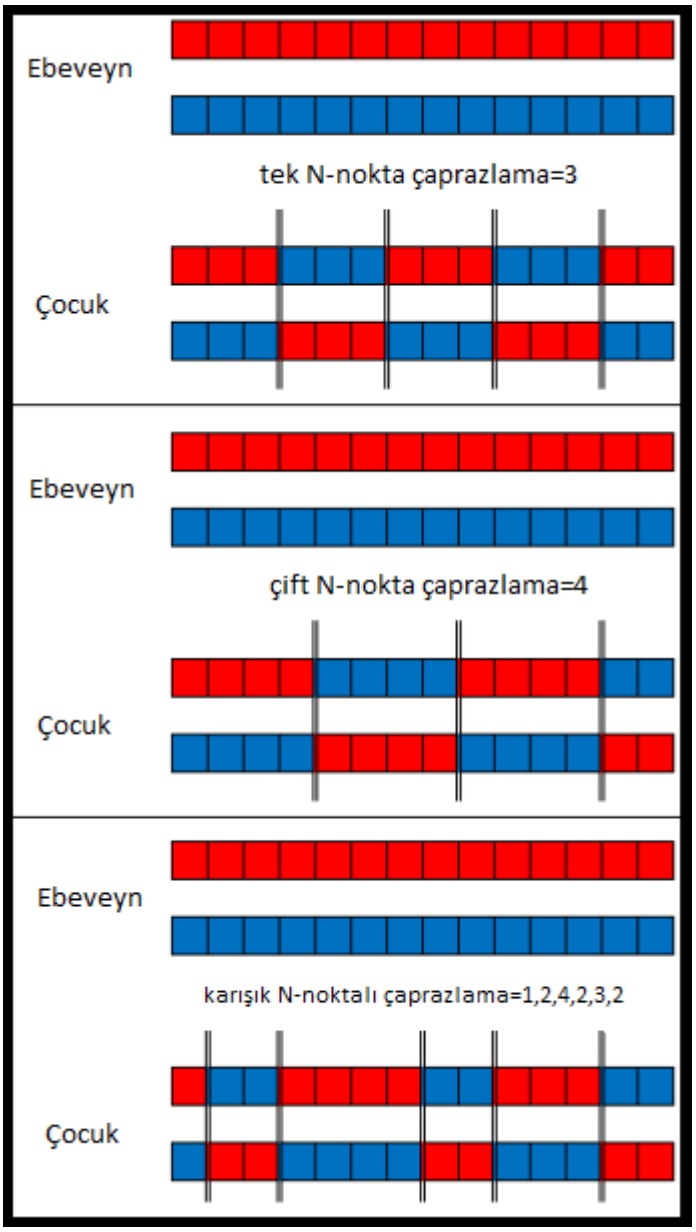
Şekil 3.1. Tek noktalı çaprazlama örneği [58]

İki noktalı veya k noktalı çaprazlamada iki veya daha fazla rastgele çaprazlama noktası seçilir ve oluşturulan segmentlere göre ebeveynlerin genetik bilgileri değiştirilir [42]. Şekil 4, çaprazlama noktaları arasında genetik bilginin değiş tokuşunu göstermektedir. Ebeveynlerin orta segmenti, yeni yavruları oluşturmak için değiştirilir.



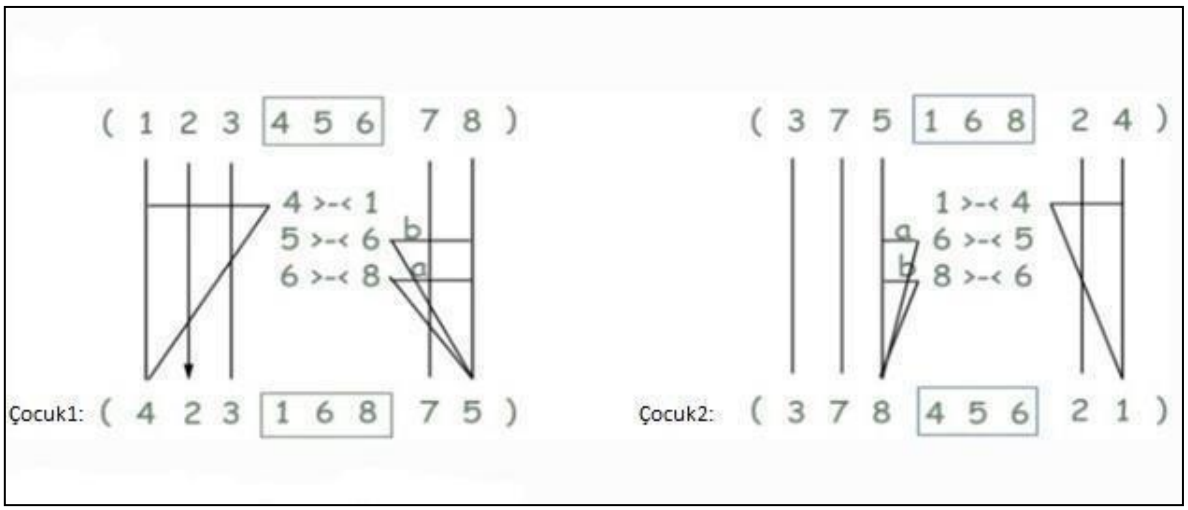
Şekil 3.2. Rastgele seçilmiş çaprazlama örneği [58]

3 çeşit N-tip çaprazlama noktası bulunmaktadır ve bunlar çift, tek ve rastgeledir. Çift N-noktalı çaprazlamada, çift bir rastgele çaprazlama noktası seçilmektedir. Genler buna göre bölünüp ebeveynler arasında geçiş yapmaktadır. Benzer operasyonlar aynı şekilde tek N-noktalı çarpazlamada da gerçekleştirilmektedir. Karışık N-noktalıda ise N tane rastgele nokta seçilip bu noktalar arasındaki genler ebeveynler arasında geçiş yapılarak değiştirilmektedir.



Şekil 3.3. N-nokta çaprazlama örnekleri [58]

Kısmen eşleştirilmiş çaprazlama (PMX) en sık kullanılan çaprazlama operatörüdür. Diğer çaprazlama operatörlerinin çoğundan daha iyi performans gösteren bir operatördür. Kısmen eşleşen (haritalanmış) çaprazlama, D. Goldberg [42] tarafından önerildi. Çiftleşme için iki ebeveyn seçilir. Bir ebeveyn, genetik materyalin bir kısmını bağışlar ve diğer ebeveynin karşılık gelen kısmı çocuğa katılır. Bu işlem tamamlandıktan sonra, kalan aleller ikinci ebeveynden kopyalanır. Şekil 6, PMX örneğini göstermektedir.

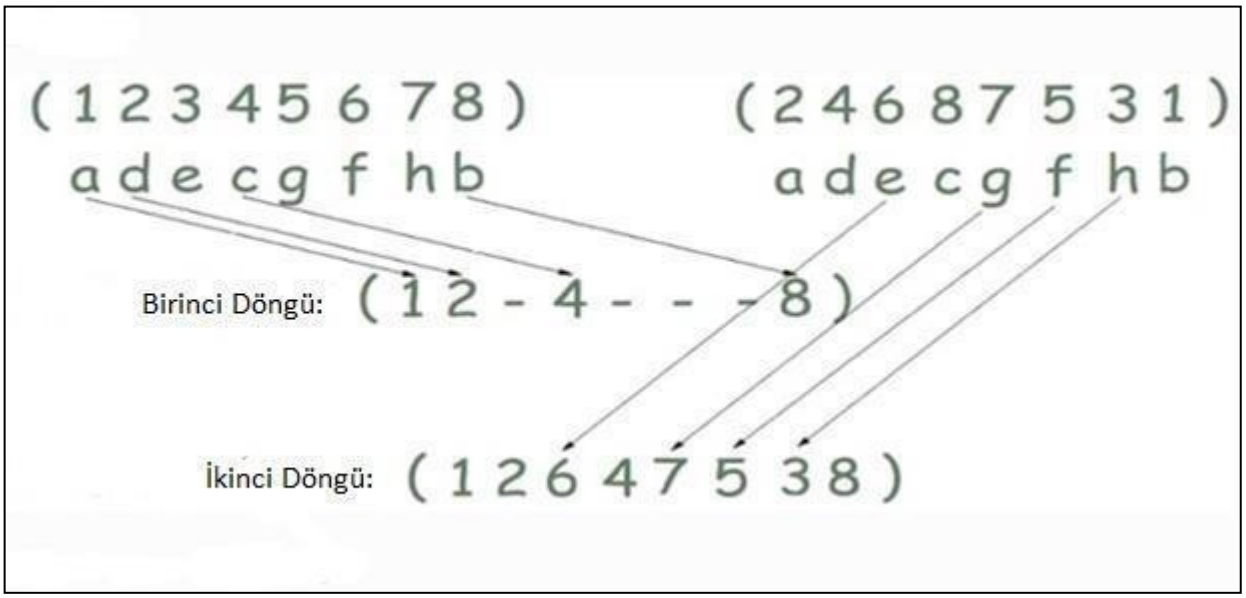


Şekil 3.4.Kısmen eşleşmiş çaprazlama örneği [58]

Sıra çaprazlama (OX) Davis [43] tarafından 1985 yılında önerilmiştir. OX, ebeveynin bir (veya daha fazla) parçasını seçilen kesim noktalarından yavruya kopyalar ve kalan boşluğu kopyalanan bölümde yer alan değerler dışındaki değerlerle doldurur. OX'in varyantları, farklı problem türleri için farklı araştırmacılar tarafından önerilmiştir. OX, sorunları sıralamak için kullanışlıdır. Önceliği koruyan çaprazlama (PPX), çaprazlama uygulamasından önce yavruların ebeveyninde bulunan bireysel çözümlerin sırasını korur.

Karışık çaprazlama, Eshelman ve diğerleri tarafından diğer çaprazlama tekniklerinin getirdiği önyargıyı azaltmak için önerildi [44]. Çaprazlamadan önce tek bir çözümün değerlerini karıştırır ve çaprazlama işlemi gerçekleştirildikten sonra bunları çözer, böylece çaprazlama noktası çaprazlamada herhangi bir sapma oluşturmaz. Ancak, bu çaprazlamanın kullanımı son yıllarda çok sınırlıdır. Ebeveynler çözüm temsilleri için aynı gen dizisine sahipse, azaltılmış vekil çaprazlama (RCX), gereksiz çaprazlamaları azaltır. RCX, ebeveynler genetik bileşimlerinde yeterince çeşitliyse, GA'nın daha iyi bireyler ürettiği varsayımına dayanır. Ancak, RCX aynı bileşime sahip ebeveynler için daha iyi bireyler üretemez. Döngü çaprazlaması Oliver [45] tarafından önerildi. Ebeveynlerin konumuna atıfta bulunarak her öğenin konumu işgal ettiği ebeveynleri kullanarak bir yavru oluşturmaya çalışır [46]. İlk

döngüde, ilk ebeveynden bazı öğeler alır. Sonraki döngüde ise öbür ebeveynden öğeler almaktadır.



Şekil 3.5. Döngü çaprazlama örneği [58]

3.2.4. Mutasyon operatörleri

Mutasyon, bir popülasyondan sonraki popülasyona genetik çeşitliliği koruyan bir operatördür. İyi bilinen mutasyon operatörleri, yer değiştirme, basit ters çevirme ve karıştırmalı mutasyondur. Yer değiştirme mutasyonu (DM) operatörü, belirli bir bireysel çözümün bir alt dizisini kendi içinde değiştirir. Yer, yer değiştirme için verilen alt diziden rastgele seçilir, öyle ki elde edilen çözüm, bir rastgele yer değiştirme mutasyonunun yanı sıra geçerli olur. DM'nin varyantları, değişim mutasyonu ve yerleştirme mutasyonudur. Değişim mutasyonu ve yerleştirme mutasyon operatörlerinde, bireysel bir çözümün bir parçası ya başka bir parça ile değiştirilir ya da başka bir yere eklenir. Buradaki dezavantaj ise erken yakınsamanın gerçekleşmedir.

Basit ters çevirme mutasyon operatörü (SIM), tek bir çözümde belirtilen herhangi iki konum arasındaki alt diziyi tersine çevirir. SIM, rastgele seçilen diziyi tersine çeviren ve onu rastgele bir konuma yerleştiren bir ters çevirme operatörüdür. Karıştırma mutasyonu (SM) operatörü, bireysel çözümün belirli bir aralığındaki öğeleri rastgele bir sırayla yerleştirir ve yakın zamanda oluşturulan çözümün uygunluk değerinin iyileşip iyileşmediğini kontrol eder. Bu mutasyon operatörünün dezavantajı ise popülasyonun bozulma ihtimalidir [38].

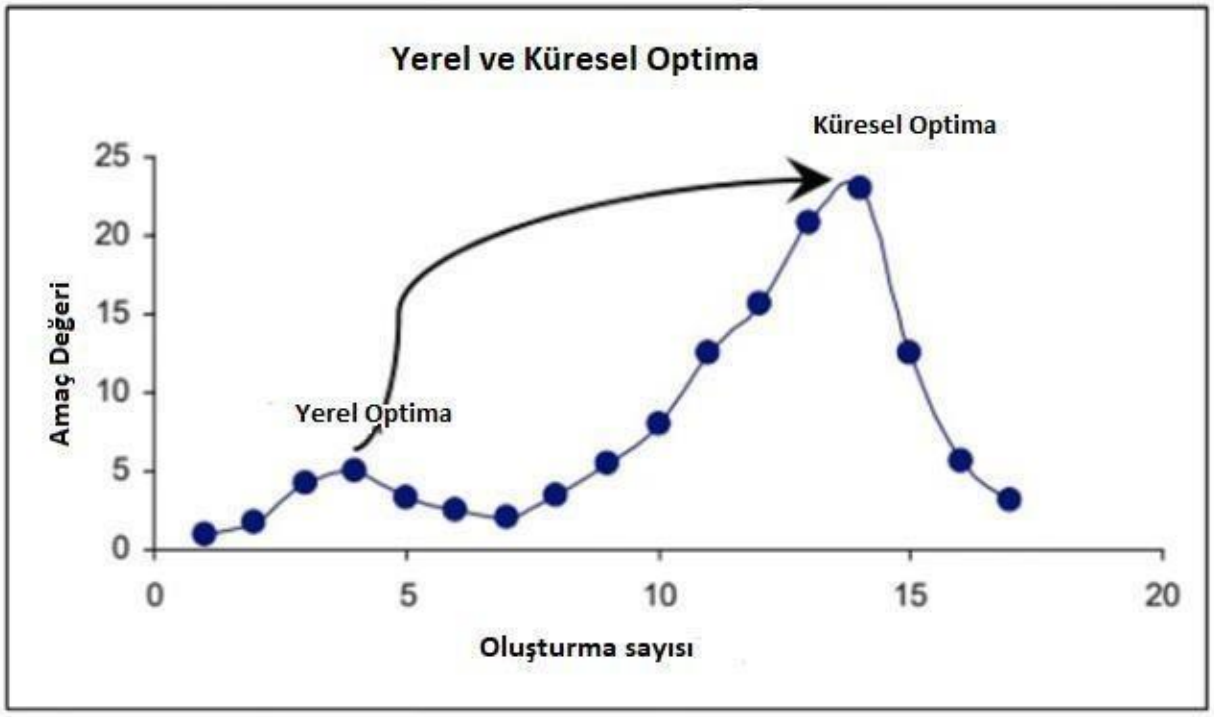
3.3. Evrimsel Algoritmada Karar Verilmesi Gereken Zorluklar

3.3.1. Başlangıç popülasyonunun seçimi

Başlangıç popülasyonu, genetik algoritmaların performansı için her zaman önemli bir faktör olarak kabul edilir. Popülasyonun büyüklüğü de çözümün kalitesini etkiler. Eğer küçük bir popülasyon seçilirse başarılı bir sonuç elde edilemeyebilir veya büyük bir popülasyon seçilirse mantıklı bir sürede iyi bir sonuç elde edilemeyebilir. Başlangıçta problem karmaşıklığı optimal popülasyon büyüklüğünü etkiler mi sorusuna cevap olarak bir problemin karmaşıklığı arttıkça kesin olarak optimal popülasyon büyüklüğünü seçmek de zorlaşmaktadır [47]. Araştırmacılar, büyük bir popülasyon göz önüne alındığında, algoritmanın daha fazla hesaplama süresi aldığını savunuyorlar. Bununla birlikte, küçük popülasyon zayıf bir çözüme yol açabilir [48]. Bu nedenle, uygun popülasyon büyüklüğünü bulmak her zaman zorlu bir konudur. Harik et al [49], kendi kendine adaptasyon yöntemini kullanarak popülasyonu araştırdı. Popülasyon boyutunun aynı kaldığı algoritmanın yürütülmesinden önce kendi kendine uyarlamanın kullanılması ve popülasyon boyutunun etkilendiği algoritma yürütme sırasında kullanılan uygunluk fonksiyonu ile öz uyarlamanın kullanıldığı iki yaklaşım kullandılar.

3.3.2. Erken yakınsama

Erken yakınsama, genetik algoritma için yaygın bir sorundur. Bir geni tanımlamayı zorlaştıran alellerin kaybına yol açabilir [45]. Erken yakınsama, optimizasyon probleminin çok erken çakışması durumunda sonucun optimalin altında olacağını belirtir. Bu sorunu önlemek için, bazı araştırmacılar çeşitliliğin kullanılması gerektiğini öne sürdüler. Çeşitliliği artırmak için seçim baskısı kullanılmalıdır. Seçim baskısı, genetik algoritmaların başlangıç popülasyonundaki daha iyi bireyleri destekleyen bir derecedir. Seçim baskısı (SP1) bazı seçim baskılarından (SP2) daha büyükse, SP1 kullanan popülasyon SP2 kullanan popülasyondan daha büyük olmalıdır. Daha yüksek seçim baskısı, erken yakınsamaya yol açabilecek popülasyon çeşitliliğini azaltabilir. Algoritmanın yerel optimal çözüm yerine global optimal çözümü bulması için yakınsama özelliği düzgün bir şekilde ele alınmalıdır (bkz. Şekil 3.6). Optimal çözüm, uygun olmayan bir çözümün yakınındaysa, Genetik algoritmanın global doğası, tabu arama ve yerel arama gibi diğer algoritmaların yerel doğası ile birleştirilebilir. Genetik algoritmaların küresel doğası ve tabu araştırmasının yerel doğası, yoğunlaştırma ve çeşitlendirme arasında uygun dengeyi sağlar.



Şekil 3.6. Yerel ve küresel optima [58]

3.3.3. Verimli uygunluk fonksiyonlarının seçimi

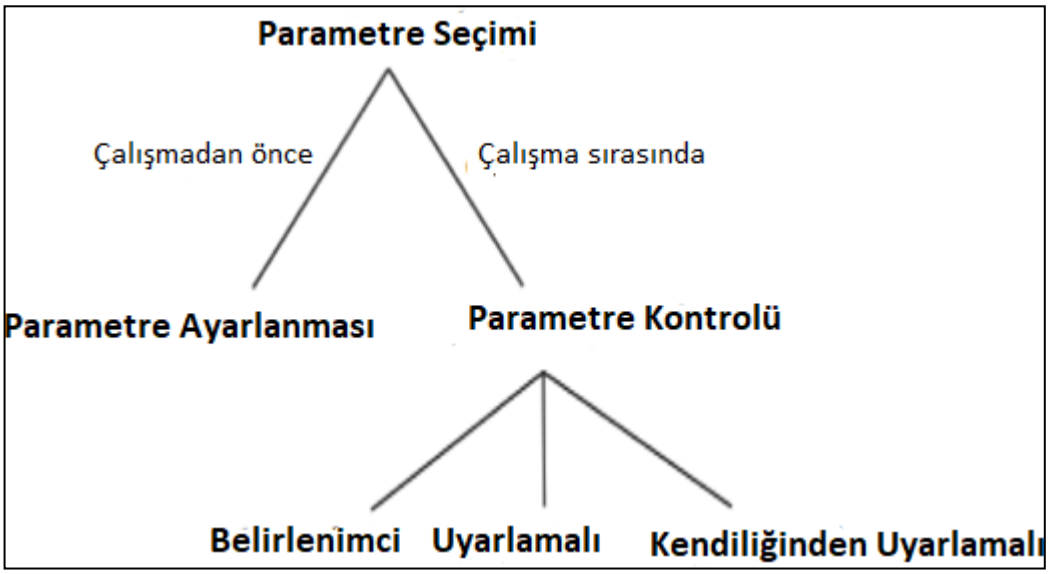
Uygunluk işlevi, bir algoritmanın her yinelemesinde en uygun bireyin seçilmesinde önemli bir rol oynayan itici güçtür. Yineleme sayısı küçükse, maliyetli bir uygunluk fonksiyonu ayarlanabilir. Yineleme sayısındaki artış, hesaplama maliyetini artırabilir. Uygunluk fonksiyonunun seçimi, uygunluklarının yanı sıra hesaplama maliyetine de bağlıdır

3.3.4. Mutasyon ve Çaprazlama Derecesi

Çaprazlama ve mutasyon operatörleri genetik algoritmaların ayrılmaz bir parçasıdır. Evrim sırasında mutasyon dikkate alınmazsa, evrim için yeni bir bilgi olmayacaktır. Evrim sırasında çaprazlama dikkate alınmazsa, algoritma yerel optimuma neden olabilir. Bu operatörlerin derecesi, genetik algoritmaların performansını büyük ölçüde etkiler. EA'ların etkinliği, kontrol parametrelerinin seçimine oldukça bağlıdır. (Popülasyon boyutu, karmaşık bir şekilde etkileşime giren çaprazlama ve mutasyon) Küresel optimumu sağlamak için bu operatörler arasında uygun bir denge gereklidir. Olasılık yapısı, etkili ve optimal bir çözüm için kesin dereceyi belirleyemez. EA'nın her operatörünün özel bir ve etkisi farklıdır. Bu faktörlerin etkisi, olasılıklarından etkilenir, yani %100 çaprazlama olasılığı %50'den tamamen farklı sonuçlar verir. Aynısı mutasyon için de geçerli olasılık. Bu nedenle, EA'nın tasarımı, bu parametreler için bir seçim stratejisi ile sınırlıdır.

Çaprazlama ve mutasyon oranları arasında bir denge sağlamak, EA'daki parametreleri kontrol etme meselesidir. Nitekim değerlerin kontrol edilmesi, EA'daki en önemli araştırma alanlarından biridir.

Parametre ayarı olarak bilinen bir ana konu altında araştırılmıştır. Çalışma sırasında parametre değerleri davranışına göre sınıflandırılan iki ana parametre değeri ayarlama türü vardır: (i) Birincisi, çaprazlama ve mutasyon için farklı değerlerin denenmesine dayanan parametre ayarlamadır (ortak yaklaşım), ve daha sonra algoritmanın son çalışmasından önce en iyi sonuçları verenlerin seçilmesi. Bu işlem, çalışma (sabit değer) sırasında değerde herhangi bir değişiklik olmadan yapılır. (ii) İkinci tip, çalıştırma sürecinde bir şekilde değiştirilen çaprazlama ve mutasyon için başlangıç parametre değerlerine bağlı olan parametre kontrolüdür [50].



Şekil 3.7. Parametre seçim teknikleri

3.4. Kullanılan Yerleşim Gösterimi Şekline göre Evrimsel Algoritma Çalışmaları

3.4.1. FBS kullanılan evrimsel algoritma araştırmaları

Tablo 3.1. FBS tipi gösterim kullanılan makaleler

Ad	An island model genetic algorithm for unequal area facility layout problems[52]	A new relaxed flexible bay structure representation and particle swarm optimization for the unequal area facility layout problem [6]	Unequal-area facility layout by genetic search [5]
Yıl	2017	2011	1994
Çözüm yolu	Paralel Genetik algoritma	Parçacık Sürü Optimizasyonu	Genetik algoritma
Çözüm aşamaları	P sayıda birey oluşturulur ve daha sonrasında adalara dağıtılırlar. Her adada en iyi bireyler üreme için seçilir ve mutasyon ve çaprazlamaya uğrarlar. Daha sonrasında en iyi m tane birey yanındaki adaya geçerler.	Populasyon bazlı bir algoritmadır. Amaç her parçacığın kendi optimal yerini bulmasıdır. Parçacıklar çözüm uzayı üzerinde dolaşırken birbirleriyle haberleşirler, yayın yaparlar. En iyi pozisyonlarının uygunluğu ve diğerlerinin en iyi pozisyonları hakkında bilgi edinirler	Başlangıç populasyonu oluşturuldaktan sonra çaprazlama ve mutasyon uygulanarak ve populasyon değiştirilerek en iyi sonuca ulaşmaya çalışılır. Her üremede bir adet çocuk yaratılır.
Mümkün olmayan kısıtlar	Dx^k ile amaç fonksiyonu çarpılır. Dx geometrik koşulları sağlamayan departmanlar. k ise 3 olan bir katsayıdır.	Geometrik koşullara uymayan departmanlar cezalandırılmış amaç fonksiyonu ile cezalandırılmaktadır.	Ni^k*(Vfeas-Vall). Ni geometrik koşullara uymayan departman sayısı. K 3 olan bir katsayı. Vfeas: En iyi uygun amaç fonksiyonu. Vall: En iyi amaç fonk.
Parametre seçimi	Önerilen algoritma parametreleri empirik olarak ayarlanmıştır. Belirli değerler seçilip tam faktoriyel deney yapılmıştır.	Parametreler farklı ayarlarla test edilmiştir.	Manuel parametre ayarlaması ile karar verilmiştir.

3.4.2. STS kullanılan evrimsel algoritma araştırmaları

Tablo 3.2. STS tipi gösterim kullanılan makaleler

Ad	Harmony search for the layout design of an unequal area facility[51]	STaTS: A Slicing Tree and Tabu Search based heuristic for the unequal area facility layout problem [31]	A Slicing Tree Representation and QCP-Model-Based Heuristic Algorithm for the Unequal-Area Block Facility Layout Problem [55]	A biased random-key genetic algorithm for the unequal area facility layout problem [33]	Applying Ant System for solving Unequal Area Facility Layout Problems [57]
Yıl	2017	2009	2013	2015	2010
Çözüm yolu	Harmonik Arama	Tabu Araması	Kendiliğinden adapte olan harmoni araması	Önyargılı Rastgele Genetik Algoritma (BRKGA)	Karınca kolonisi optimizasyonu
Çözüm aşamaları	İlk olarak belirli bir sayıda çözüm yaratılır. Daha sonra belli bir ihtimalle ya yeni bir çözüm yaratılır ya da çözüm alanındaki her sonuçtan birer nod alarak yeni bir çözüm yaratılır.	Belirli sayıda çözüm üretilir. 4 farklı hareket tanımlanmıştır. Bunların her birisi yerleşim üzerinde gerçekleştirilir ve uygun sonuç çıkartan en iyi yerleşim alınır. Bu durumda bazı hareketler uygun yerleşimi bozacağından yapılmayacaktır.	Parametreler ve harmoni hafıza matriksi tanımlanır. M1=0 ve harmoni sayıcısı 1 tanımlanır. Boş kesim aşaması tanımlanır ve harmoni sayıcısı 1 güncellenir. M1:M1+1.harmoni sayıcısı 2 tanımlanır, M2=0. Gerçek kesim içerikleri yaratılır	BRKGA ile novel yerleştirme stratejisi birleştirilmiştir. BRKGA'da bir ebeveyn elit seçimle seçilecek olup diğeri rastgele seçilmektedir. Eşleşecek ebeveyn seçilirken tekrarlanma mümkün olduğundan birden fazla birey ortaya çıkabilmektedir.	Algoritma feromon bilgisini ve sezgisel bilgileri kullanarak karınca sonuçları bulmaktadır. Gereken 3 kısmı aynı anda gerçekleşmemektedir. Öncelikle kesim sırası ve oryantasyonları belirlenmektedir ve bu bilgi feromon bilgisinden gelmektedir.
	Bu aşamadan sonra yine belli bir ihtimalle çözümün vektörleri değiştirilir. Bundan sonra belirli bir ihtimalle kesim tipi vektörü değiştirilir ve bunların sonucunda yeni bir çözüm çıkar. Elde edilen çözüm populasyonla karşılaştırılır	Her hareket için tabu listesi oluşturulur ve bazı hareketlerin tekrarlanması önlenir. Tabu aramasında bulunan en iyi sonuç saklanır sonrasında ise tepe-tırmanma stratejisi için başlangıç noktası olarak kullanılır.	Oluşturulan harmoni vektörünün QCP modeli ile çözülür. Belli ihtimalle vektörler değiştirilir ve HMCRn ve PARn tanımlanır. Bu iki değer algoritma boyunca öğrenilir ve değişir. Tekrar QCP modeli çözülerek amaç fonksiyonu bulunur.	BRKGA, r-boyutluhiperkübü durmadan ararken böylelikle kombinatoryal optimizasyon probleminin sonuç evrenini de araştırmaktadır.	Kesim sırası ve oryantasyon daha sonrasında sahte yerleşime dönüştürülüp sezgisel bilginin hesaplanmasında kullanıp daha sonrasında ise departman sıraları belirlenmektedir. Bundan sonra yerel arama bilgileri kullanılıp feromon bilgisi güncellenmektedir.

Mümkün olmayan kısıtlar	u^v ile amaç fonksiyonu çarpılır. (u: belli bir katsayı,v: en-boy oranını ihlal eden departman sayısı)	Bağlayıcı eğri sayesinde yerleştirilen her departman halihazırda geometrik koşullarını sağlamaktadır.	QCP modeli kullanıldığından yerleştirilen departmanlar geometrik kurallara uymaktadır.	Maksimum boş alan yöntemi kullanılarak departmanların geometrik koşulları sağlanmıştır	Uyarlanabilir ceza fonksiyonu kullanılmaktadır.
Parametre seçimi	Iterated racing prosedürü kullanılmaktadır.	Tabu araması gerçekleştirirken herhangi bir parametre kullanılmamıştır.	Her problem için Taguchi analizi yapılmıştır. N1, HMS1, HMS2 ve LP değerlerinin farklı kombinasyonları test edilmiş ve sonrasında en iyi kombinasyonları seçilmiştir.	BRKGA parametreleri aynı gelişimsel algoritmaları kullanılan benzer çalışmaları baz alınarak karar verilmiştir.	Manuel parametre ayarlaması ile seçilmişlerdir.

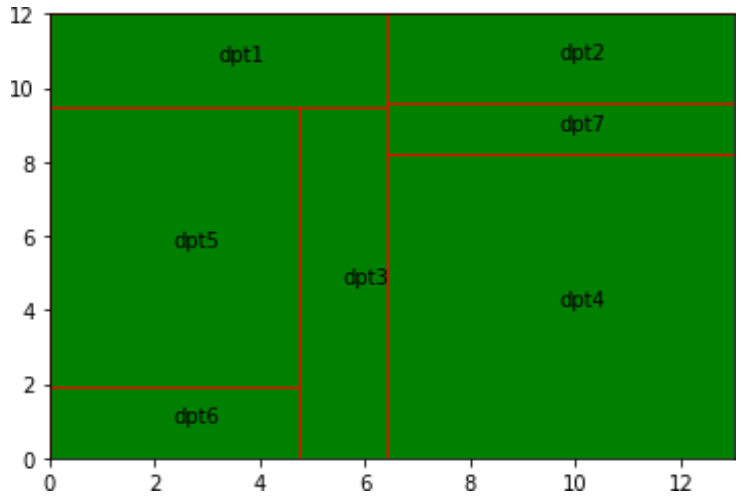
4. ÖNERILEN TESIS YERLEŞIMINDE GENEL GÖSTERIMLI EVRIMSEL ALGORITMA MODELI

4.1. Tesis Yerleşim Gösterimi

Önerilen yaklaşımda hem STS hem de FBS gösterimleri birlikte kullanılmaktadır. Başlangıçta 0.5 ihtimalle STS ya da FBS oluşturulacağı belirlenip, vektörler ona uygun şekillenmektedir. Böylelikle aynı kodlama üzerinde hem STS hem FBS gösterimleri yapılabiliyor olup son tesis yerleşim planları da 2 gösterim şekli özelinde karşılaştırılabilmektedir.

Departman Sırası	**1 5 6 3 2 7 4**
Kesim Sırası	**4 1 5 3 6 2**
Yön	**0 1 1 0 1 1**

(Yerleşim kodlaması)

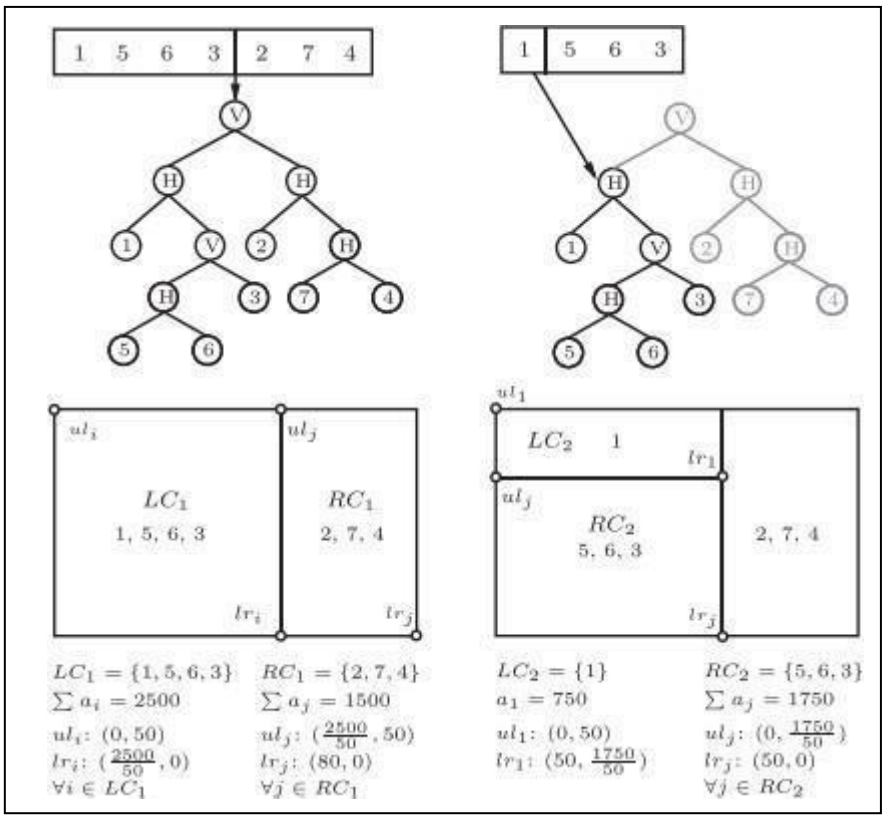


(Ortaya çıkan yerleşim şeması)

Şekil 4.1. Vektör kodlamalarının yerleşim şemasına dönüştürülme örneği

Kesim ağacı bir alanın nasıl ayrılacağını gösterir ancak her bir departmanın tam koordinatlarını göstermemektedir. Bu nedenle 3 vektörden oluşan kesim ağacının yerleşime dönüştürülmesi için metodolojiye ihtiyaç duyulmaktadır.

Başlangıç olarak bir yerleşim oluşturulurken departmanların alanlarının toplamının mevcut tesis alanına eşit olması gerekmektedir. Eğer yerleştirilecek departmanlar ve yerleşilecek tesis toplam alanı bu koşulu sağlıyor ise metodoloji uygulanabilir hale gelmektedir. Tesis eni w_f ve tesis boyu h_f olarak tanımlanmaktadır. Sol üst koordinat ul_i ve sağ üst köşeler lr_i her departman için (0, h_f) ve (w_f,0) olarak tanımlanmaktadır. İlk nod kesimin tipini belirtmektedir (dikey, yatay). Bu durumda kesimin sol tarafında kalandepartmanlar LC1 ve sağ tarafında kalandepartmanlar RC1 kümelerine ayrılır. Her kümeiçin gerekli alan dikkate alınarak kesimin lokasyonu belirlenir. Her LC1'deki i değeri için lr_i ve RC1'deki her i değeri için ul_i'nin x koordinatları sol kümenin alanlarının toplamının, toplam tesis alanına bölünmesiyle yeniden hesaplanır. Eğer nod bir yaprak nodu ise departmanın yerleşeceği lokasyonun koordinatları tam olarak hesaplanır. Bu durum, her karşılanacak küme için aynı işlemler yapılmalıdır. Bu işlemlerin sonucunda her departmanın koordinatlarının bilindiği bir yerleşim ortaya çıkmaktadır.



Şekil 4.2. Kodlama şemasına göre yerleşim oluşturmasının örnek 2 adımı [58]

Yukarıdaki örnekte 3 vektör bulunmaktadır ve bu vektörler sonucunda yukarıdaki yerleşim şeması elde edilmektedir. Bu araştırmada kesim noktası n-1 departman sayısı kadar, kesim kodlaması ise 2 çeşittir. 0 yatay kesim ve 1 dikey kesim yapılacağını belirtmektedir. Departman vektörü ise n sayıdan oluşmaktadır. Oluşturulan yerleşim şemaları FBS ya da karışık FBS olarak tanımlanıp, algoritma içinde gerçekleştirilecek değişimlerde bu tip yerleşim tiplerini bozmayacak şekilde gerçekleştirilmektedir.

4.2. Evrimsel Algoritma

4.2.1. Tanımlama aşaması

Bu aşamada her farklı departman için 3 farklı vektör tanımlanmaktadır. Kaç farklı yerleşim şeması yaratılacağı belirlenmektedir. Buna Hafıza büyüklüğü denmektedir. Bundan sonra algoritmanın temelini oluşturan 3 farklı olasılık tanımlanmaktadır. Bunlar CR (consideration rate) önem oranı, AR (adjustment rate) ayarlama oranı ve RAR (re-adjustment rate) yeniden ayarlama oranıdır. Bu 3 oran önceden algoritma için karar verilmektedir. En son olarak da tanımlama aşamasında algoritmanın ne zaman duracağı belirlenmektedir. Genelde çalışmalarda popülasyonda bulunan en iyi sonuçta gelişme olmadığında veya belirli bir toplamhareket sayısına geldiğinde durma gibi kriterler kullanılmaktadır. Bu çalışmada durma kriteri popülasyonda bulunan en iyi sonuçta gelişme olmadığında gerçekleşecek şekilde tanımlanmıştır.

4.2.2. Geliştirme aşaması

Bu aşama ana algoritmanın çalıştığı aşamadır. Populasyona yeni bir çocuk eklemek için 3 seçenekten birisi gerçekleştirilmektedir. Bunlardan birinci Rastgele seçimdir. Burada 1-CR ihtimalle rastgele yeni bir yerleşim üretilip popülasyona katılmaktadır. CR ihtimalle ise popülasyondan iki yerleşim seçilip bu ikisinden yeni bir yerleşim oluşturulmaktadır. CR ihtimalle popülasyondan yerleşim seçilmesitamamen rastgeledir. Her bir yerleşimin eşit seçilme şansıbulunmaktadır.

CR ile çocuk oluşturulurken iki ebeveynin kesim kodu, kesim sırası ve departman vektörleri alınır. Her bir vektör için bir kesim noktası belirlenir ve oraya kadarki sayılar birinci ebeveynden ve geri kalan öğeler de ikinci ebeveynden alınır. Oluşan çocukta her kesim kodlaması dışında herhangi bir tekrarlama meydana gelirse, tekrarlar kaldırılacak şekilde düzenleme yapılır. Böylelikle iki ebeveynden bir adet yerleşimelde edilmiş olup popülasyona katılır.

Son aşama ise ayarlama aşamasıdır. AR ihtimalle oluşturulan yerleşim değiştirilebilir. Bu aşama algoritmanın yerelaraştırması içindir.

Burada 4 farklı ayarlama tipi vardır ve hepsi eşit ihtimallerle gerçekleştirilebilir. Her bir yerleşim en fazla bir defa ayarlanabilir.

1. 2 departman değerialınır ve birbirleriyle yer değiştirilir.
2. İkirastgele karar verilen kesim noktasıbirbirleriyle yer değiştirilir.
3. 1 departman ve bir yer seçilir. Seçilen departman seçilen sıraya koyulur ve sağındaki departmanlar birer adet sağa kayarlar.

4. 1 kesim noktası ve bir yer seçilir. Seçilen kesim noktası seçilen sıraya koyulur ve sağındakikesim noktaları birer adet sağa kayarlar.

Bu 4 değişimin de gerçekleşme ihtimali0.25'tir.

4.2.3. Yeniden ayarlama aşaması

Ayarlama aşamasından sonra ise yeniden ayarlama aşaması bulunmaktadır. Bundan önceki aşamalarda kesimkoduyla ilgili bir değişiklik yapılmamıştır. Burada ise kesimkodları RAR olasılıkla değiştirilmektedir. Bu yerel aramayı daha da efektif gerçekleştirmek için yapılmaktadır. Ancak bu değerin yüksek olması sonucu rastsallığa götüreceği için genellikle küçük değerler seçilmektedir. RAR ihtimali FBS tipi yerleşimde gerçekleştirilmemektedir çünkü FBS yapısını bozma ihtimali vardır. Bu ihtimalsadece STStipi yerleşimlerde gerçekleştirilmektedir.

Sonolarak yerleşim oluşturulurken en-boy oranı hesaba katılmadığından dolayı Ceza şeması uygulanmaktadır. Amaç fonksiyonunu hesaplarken OFV*u^v olarak hesaplanmaktadır. Burada u 1'den büyük bir sayı, v ise en-boy oranını ihlal eden departman sayısıdır. Bu şemanın amacı en-boy oranına uymayan yerleşim şemalarını yüzde yüz göz ardıetmeden popülasyona katmaktır. Böylelikle ileride popülasyona olumlu etkiedecek ancak departmanların bazılarının geometrik kısıtlara uymayan yerleşimleri popülasyona dahil edilmiş olur.

4.3. Parametrelerin Optimal Değerlerinin Bulunması

Evrimsel algoritmanın başarısı genel olarak kullanılan parametrelerin doğruluğuna bağlıdır. Bu çalışmada kullanılacak olan parametrelere karar verebilmek için Taguchi deney tasarımı yapılmıştır. Butasarım AB20 probleminin verilerikullanılarak gerçekleştirilmiştir.

Öncelikle algoritma verimine etkieden 5 faktörekarar verilmiştir. Bunlar CR, AR, RAR, ceza katsayısı ve iterasyon sayısıdır. Ve bu 5 faktöründe 4 seviyesine karar verilmiştir.

Tablo 4.1. Seçilen parametreler ve değerleri

Parametre	Değer			
CR	0,9	0,8	0,7	0,6
AR	0,7	0,6	0,5	0,4
RAR	0,3	0,2	0,1	0,1
Ceza katsayısı(pf)	1,2	1,5	2,0	3,0
İterasyon sayısı(run)	100.000,0	400.000,0	700.000,0	1.200.000,0

Yukarıdaki tabloda 5 faktör ve bu faktörlerin 4'er seviyesi verilmiştir. İlk aşamadaki deney tasarımı aşağıdaki tabloda bulunmaktadır. Taguchi deney tasarımı Minitab 14 programında gerçekleştirilmiştir. Parametre kombinasyonları 16 deney tasarımı için Minitab tarafından oluşturulmuştur.

Tablo 4.2. Başlangıç Taguchi deney seti

Deney no	cr	ar	rar	run	pf			Parametreler		
1	1	1	1	1	1	0,9	0,4	0,1	200.000	1,25
2	1	2	2	2	2	0,9	0,5	0,2	500.000	1,5
3	1	3	3	3	3	0,9	0,6	0,3	800.000	2
4	1	4	4	4	4	0,9	0,7	0,4	1.200.000	3
5	2	1	2	3	4	0,8	0,4	0,2	800.000	3
6	2	2	1	4	3	0,8	0,5	0,1	1.200.000	2
7	2	3	4	1	2	0,8	0,6	0,4	200.000	1,5
8	2	4	3	2	1	0,8	0,7	0,3	500.000	1,25
9	3	1	3	4	2	0,7	0,4	0,3	1.200.000	1,5
10	3	2	4	3	1	0,7	0,5	0,4	800.000	1,25
11	3	3	1	2	4	0,7	0,6	0,1	500.000	3
12	3	4	2	1	3	0,7	0,7	0,2	200.000	2
13	4	1	4	2	3	0,6	0,4	0,4	500.000	2
14	4	2	3	1	4	0,6	0,5	0,3	200.000	3
15	4	3	2	4	1	0,6	0,6	0,2	1.200.000	1,25
16	4	4	1	3	2	0,6	0,7	0,1	800.000	1,5

Her bir deney numarası için algoritma 5 defa çalıştırılmıştır ve aşağıdaki tablodaki sonuçlar elde edilmiştir.

Tablo 4.3. Başlangıç Taguchi analizinin deneysel sonuçları

CR	AR	RAR	Run	Pf	Sonuç1	Sonuç 2	Sonuç 3	Sonuç 4	Sonuç 5
0,9	0,4	0,1	200.000	1,25	6831	7285	7566	6410	7140
0,9	0,5	0,2	500.000	1,50	5726	5640	5730	5766	5987
0,9	0,6	0,3	800.000	2,00	5561	5711	5619	5811	5661
0,9	0,7	0,4	1.200.000	3,00	5601	6217	5440	5998	5886
0,8	0,4	0,2	800.000	3,00	5894	5930	6376	6360	5588
0,8	0,5	0,1	1.200.000	2,00	5877	6535	6389	5980	6081
0,8	0,6	0,4	200.000	1,50	7938	8455	7659	7710	7360
0,8	0,7	0,3	500.000	1,25	6025	6062	5954	6639	6136
0,7	0,4	0,3	1.200.000	1,50	6011	6740	5998	6115	6038
0,7	0,5	0,4	800.000	1,25	6104	5531	5603	6110	5779
0,7	0,6	0,1	500.000	3,00	6092	6316	6121	6516	6329
0,7	0,7	0,2	200.000	2,00	8386	8599	8426	8409	8342
0,6	0,4	0,4	500.000	2,00	7230	6501	6038	6244	6089
0,6	0,5	0,3	200.000	3,00	8389	7317	8542	8446	8518
0,6	0,6	0,2	1.200.000	1,25	5674	5440	5737	5308	5884
0,6	0,7	0,1	800.000	1,50	6259	6605	6287	6444	5979

Source	DF	Seq SS	Adj SS	Adj MS	F	P
CR	3	4637943	4637943	1545981	15,63	0,000
AR	3	1281099	1281099	427033	4,32	0,008
RAR	3	162412	162412	54137	0,55	0,652
Run	3	52707909	52707909	17569303	177,66	0,000
Pf	3	3082630	3082630	1027543	10,39	0,000
Error	64	6328998	6328998	98891		
Total	79	68200991				

S = 314,469 R-Sq = 90,72% R-Sq(adj) = 88,55%

Şekil 4.3. Taguchi deney tasarımının ANOVA analiz sonuçları

Response Table for Signal to Noise Ratios
Smaller is better

Level	CR	AR	RAR	Run	Pf
1	-76,38	-76,15	-76,19	-77,93	-75,77
2	-76,40	-76,17	-76,09	-75,79	-76,24
3	-76,28	-75,97	-76,26	-75,50	-76,40
4	-75,65	-76,41	-76,17	-75,49	-76,31
Delta	0,76	0,44	0,17	2,44	0,63
Rank	2	4	5	1	3

Şekil 4.4. Taguchi deneyinin S/N oranı sonuçları

Şekil 4.3'te görüleceği üzere öncelikle deneye ANOVA analizi yapılmıştır. Burada etki yüzdesi en fazla olan parametre %77 ile iterasyon parametresi olduğu gözlemlenmektedir. Daha sonrasında %7 ile CR, %4,5 ile ceza katsayısı, %2 ile AR ve %0.02 ile RAR gelmektedir. Seçilen parametreler ve faktörleri ile iterasyon sayısı oldukça baskın çıkmıştır. Seçilen parametreler ve faktörler bu deney tasarımında sonuçların %90,72'sini açıklayabilmektedir. Deneyde S/N oranlarına bakarak parametrelerin optimal değerleri CR, AR, RAR, run ve pf olmak üzere sırasıyla 0.9, 0.5, 0.1, 1.200.000 ve 1.2 çıkmıştır. Ancak run parametresinin oldukça baskın olmasından dolayı optimal bulunan bu değerlerden yola çıkarak yeni bir Taguchi deneytasarımı gerçekleştirilmesine karar verilmiştir.

Tekrarlanan deney tasarımında bu sefer 4 faktör ve 4 seviye bulunmaktadır. Bu faktör ve seviyeleri baz alınarak 16 deney tasarımlı bir Taguchi tasarımının yeterli olacağına karar verilmiştir. Tablo 3.4'te bir önceki deney tasarımında bulunan optimal değerlerin çevresinde deney seti verilmiştir.

Tablo 4.4. Taguchi deney seti ve sonuçları

Deney no	cr	ar	rar	pf	cr	ar	rar	pf	Sonuç
1	1	1	1	1	0,95	0,65	0,10	1,2	5776
1	1	1	1	1	0,95	0,65	0,10	1,2	5823
1	1	1	1	1	0,95	0,65	0,10	1,2	5841
1	1	1	1	1	0,95	0,65	0,10	1,2	5711
1	1	1	1	1	0,95	0,65	0,10	1,2	5813
2	1	2	2	2	0,95	0,60	0,15	1,4	5794
2	1	2	2	2	0,95	0,60	0,15	1,4	5731
2	1	2	2	2	0,95	0,60	0,15	1,4	5591
2	1	2	2	2	0,95	0,60	0,15	1,4	5922
2	1	2	2	2	0,95	0,60	0,15	1,4	5685
3	1	3	3	3	0,95	0,55	0,20	1,6	5576
3	1	3	3	3	0,95	0,55	0,20	1,6	6078
3	1	3	3	3	0,95	0,55	0,20	1,6	6039
3	1	3	3	3	0,95	0,55	0,20	1,6	5877
3	1	3	3	3	0,95	0,55	0,20	1,6	6039
4	1	4	4	4	0,95	0,50	0,25	1,8	5682
4	1	4	4	4	0,95	0,50	0,25	1,8	5815
4	1	4	4	4	0,95	0,50	0,25	1,8	6089
4	1	4	4	4	0,95	0,50	0,25	1,8	5681
4	1	4	4	4	0,95	0,50	0,25	1,8	5619
5	2	1	3	2	0,90	0,65	0,20	1,4	5857
5	2	1	3	2	0,90	0,65	0,20	1,4	6093
5	2	1	3	2	0,90	0,65	0,20	1,4	5705
5	2	1	3	2	0,90	0,65	0,20	1,4	5525
5	2	1	3	2	0,90	0,65	0,20	1,4	5706
6	2	2	4	1	0,90	0,60	0,25	1,2	5849
6	2	2	4	1	0,90	0,60	0,25	1,2	5706
6	2	2	4	1	0,90	0,60	0,25	1,2	5264
6	2	2	4	1	0,90	0,60	0,25	1,2	5836
6	2	2	4	1	0,90	0,60	0,25	1,2	5526
7	2	3	1	4	0,90	0,55	0,10	1,8	6491
7	2	3	1	4	0,90	0,55	0,10	1,8	6045
7	2	3	1	4	0,90	0,55	0,10	1,8	5663
7	2	3	1	4	0,90	0,55	0,10	1,8	5728
7	2	3	1	4	0,90	0,55	0,10	1,8	5825
8	2	4	2	3	0,90	0,50	0,15	1,6	5558
8	2	4	2	3	0,90	0,50	0,15	1,6	5921
8	2	4	2	3	0,90	0,50	0,15	1,6	5845
8	2	4	2	3	0,90	0,50	0,15	1,6	5791
8	2	4	2	3	0,90	0,50	0,15	1,6	5800
9	3	1	4	3	0,85	0,65	0,25	1,6	5315
9	3	1	4	3	0,85	0,65	0,25	1,6	5488
9	3	1	4	3	0,85	0,65	0,25	1,6	5791
9	3	1	4	3	0,85	0,65	0,25	1,6	5675
9	3	1	4	3	0,85	0,65	0,25	1,6	5488

10	3	2	3	4	0,85	0,60	0,20	1,8	5454
10	3	2	3	4	0,85	0,60	0,20	1,8	5608
10	3	2	3	4	0,85	0,60	0,20	1,8	6171
10	3	2	3	4	0,85	0,60	0,20	1,8	6012
10	3	2	3	4	0,85	0,60	0,20	1,8	5858
11	3	3	2	1	0,85	0,55	0,15	1,2	5906
11	3	3	2	1	0,85	0,55	0,15	1,2	5905
11	3	3	2	1	0,85	0,55	0,15	1,2	5935
11	3	3	2	1	0,85	0,55	0,15	1,2	5837
11	3	3	2	1	0,85	0,55	0,15	1,2	5717
12	3	4	1	2	0,85	0,50	0,10	1,4	5736
12	3	4	1	2	0,85	0,50	0,10	1,4	6272
12	3	4	1	2	0,85	0,50	0,10	1,4	5600
12	3	4	1	2	0,85	0,50	0,10	1,4	6039
12	3	4	1	2	0,85	0,50	0,10	1,4	5805
13	4	1	2	4	0,80	0,65	0,15	1,8	6074
13	4	1	2	4	0,80	0,65	0,15	1,8	5799
13	4	1	2	4	0,80	0,65	0,15	1,8	6117
13	4	1	2	4	0,80	0,65	0,15	1,8	5768
13	4	1	2	4	0,80	0,65	0,15	1,8	5538
14	4	2	1	3	0,80	0,60	0,10	1,6	6392
14	4	2	1	3	0,80	0,60	0,10	1,6	5722
14	4	2	1	3	0,80	0,60	0,10	1,6	6127
14	4	2	1	3	0,80	0,60	0,10	1,6	5554
14	4	2	1	3	0,80	0,60	0,10	1,6	5601
15	4	3	4	2	0,80	0,55	0,25	1,4	5823
15	4	3	4	2	0,80	0,55	0,25	1,4	6044
15	4	3	4	2	0,80	0,55	0,25	1,4	6155
15	4	3	4	2	0,80	0,55	0,25	1,4	5601
15	4	3	4	2	0,80	0,55	0,25	1,4	5641
16	4	4	3	1	0,80	0,50	0,20	1,2	5720
16	4	4	3	1	0,80	0,50	0,20	1,2	6186
16	4	4	3	1	0,80	0,50	0,20	1,2	5619
16	4	4	3	1	0,80	0,50	0,20	1,2	5585
16	4	4	3	1	0,80	0,50	0,20	1,2	5802

Yukarıdaki 16 farklı deney tasarımı yapılmış olup hepsi 5'er kere çalıştırılmıştır. Bundan sonra yukarıdaki sonuçlar MİNİTAB 14'te Taguchi analizinde kullanılmıştır.

İlk öncelikle tablodaki verilere ANOVA analizi gerçekleştirilmiştir. ANOVA sonuçları aşağıdaki gibidir.

Source	DF	Seq SS	Adj SS	Adj MS	F	P
cr	3	75125	75125	25042	15,66	0,025
ar	3	24395	24395	8132	5,08	0,107
rar	3	13160	13160	4387	2,74	0,215
pf	3	39934	39934	13311	8,32	0,058
Error	3	4799	4799	1600		
Total	15	157413				

S = 39,9949 R-Sq = 96,95% R-Sq(adj) = 84,76%

Şekil 4.5. Taguchi deney tasarımının ANOVA analiz sonuçları

R^2 değeri 96.95 olup, deney tasarımındaki sonuçların yüzde 96,95'inin seçilen parametrelerle açıklanabileceği görülmektedir. CR, AR ve PF istatiksel olarak anlamlı çıkmaktadır. Bir tek RAR istatistiksel olarak anlamsız çıkmıştır ancak bu bilgi de RAR'nin yüksek değerlerinin sonuçları rastgeleliğe götüreceğinden beklenmektedir. Genelde F değeri 4 değerinden büyükse istatistiksel olarak parametreye anlamlı yorumu yapılabilmektedir.

ANOVA analizine göre amaç fonksiyonunu en çok etkileyen parametre %47 ile CR olup, onu %25 ile PF (ceza katsayısı), %15 ile AR ve %8 ile RAR izlemektedir.

ANOVA analizinden sonra Taguchi analizi gerçekleştirilmiş ve tablolaştırılmıştır.

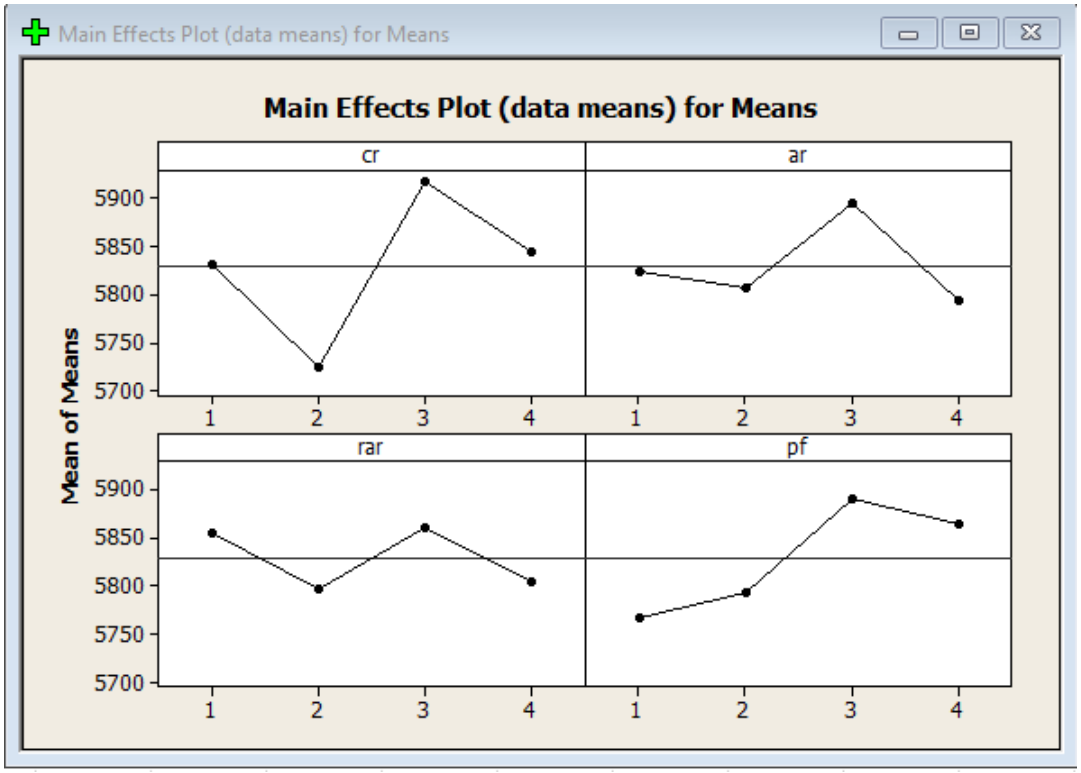
Response Table for Signal to Noise Ratios
Smaller is better

Level	cr	ar	rar	pf
1	-75,31	-75,30	-75,35	-75,22
2	-75,15	-75,28	-75,26	-75,26
3	-75,44	-75,41	-75,36	-75,40
4	-75,33	-75,26	-75,27	-75,36
Delta	0,29	0,15	0,09	0,18
Rank	1	3	4	2

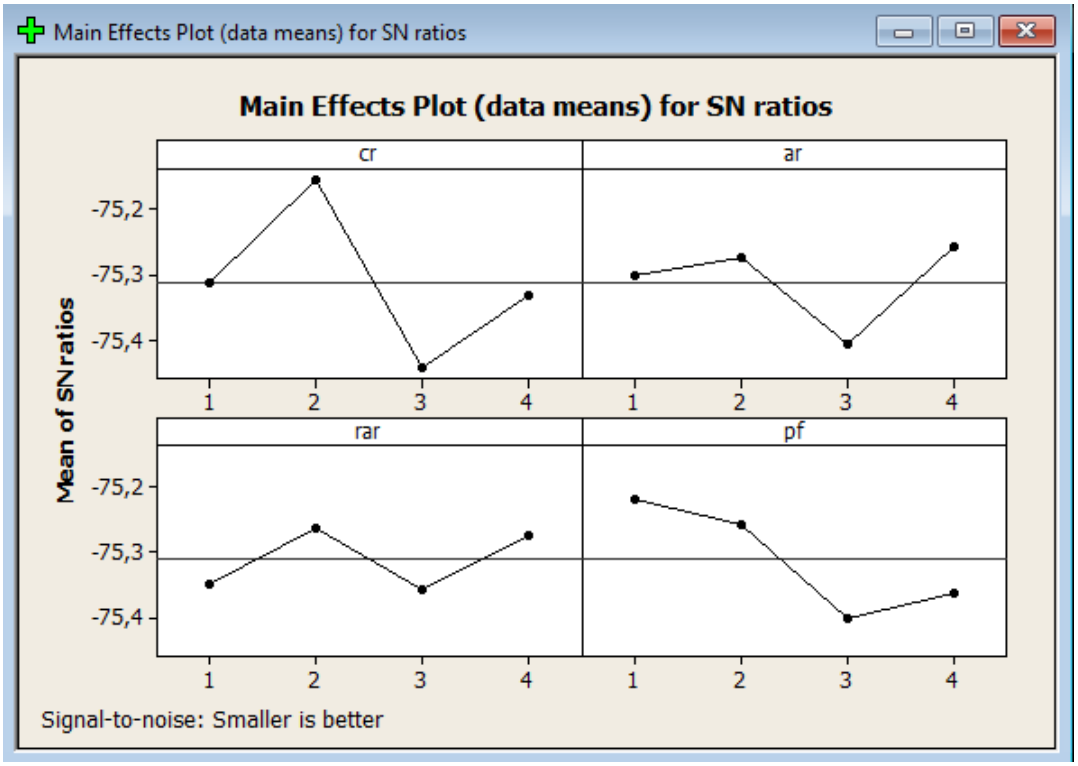
Şekil 4.6. Taguchi deneyinin S/N oranı sonuçları

Taguchi analizinde S/N oranı dikkate alınmıştır ve bu deney yapılırken amaç değerinin en küçüklenmesi istendiği için "küçük daha iyidir" prensipine göre işlemler gerçekleştirilmiştir.

Burada Signal to Noise oranı kullanılmıştır ve hesaplamalar amaç fonksiyonunun düşük daha iyidir mantığı kullanılarak gerçekleştirilmiştir. Daha sonrasında bu değerler ve Ana efektler grafiğe dönüştürülmüştür.



Şekil 4.7. Veri ortalamalarının grafiği



Şekil 4.8. S/N oranlarının grafiği

Hem ortalamaların grafiğine hem de S/N oranlarının grafiğine bakınca deney tasarımında en iyi sonuç veren parametrelerin CR:0.9, AR:0.5, RAR:0.15 ve PF:1.2 olduğu anlaşılmaktadır.

4.4. Evrimsel algoritmanın deney setlerinde kullanılması

Bu çalışmada kullanılan evrimsel algoritma O7,O8,O9,VC10a ve AB20 veri setleri üzerinde kullanılmıştır. Her bir problem için 20 kez çalıştırılıp algoritmadan alınan en iyi STS sonucu, en iyi FBS sonucu algoritmanın çalışma süresi raporlanmıştır. Daha sonrasında da bulunan sonuçlar bugüne kadarki bulunan sonuçlarla karşılaştırılmıştır.

4.4.1. O7 problemi

Tablo 4.5. O7 Problemi için veri setleri

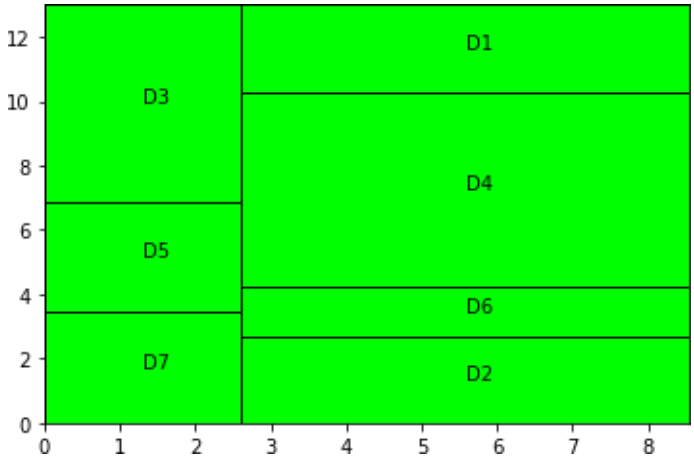
Departman sayısı	Tesis Eni	Tesis Boyu	Departmanların en-boy oranı
7	8,54	13	4

Tablo 4.6. Departmanlar arası veri akışı ve alanları

ff	1	2	3	4	5	6	7	Alanı
1	0	0	0	5	0	0	1	16
2	0	0	0	3	0	0	1	16
3	0	0	0	2	0	0	1	16
4	0	0	0	0	4	4	0	36
5	0	0	0	0	0	0	2	9
6	0	0	0	0	0	0	1	9
7	0	0	0	0	0	0	0	9

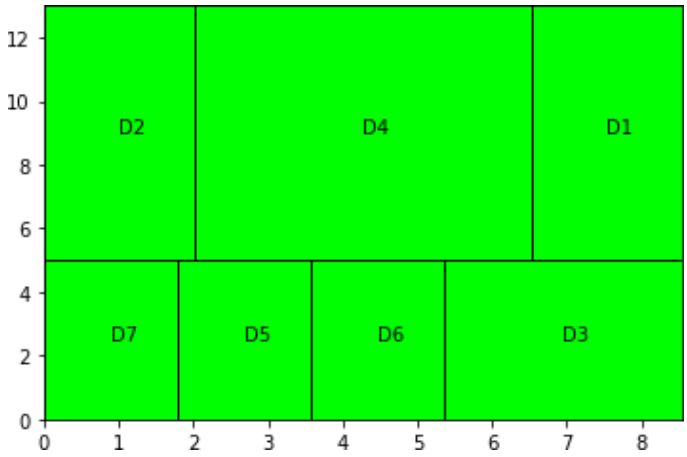
Tablo 4.7. Kullanılan parametreler ve sonuçlar

Parametreler			**Sonuçlar**			
Populasyon	9000			Dikey FBS	Yatay FBS	Karışık FBS
Önem oranı (CR)	0,9		Ortalama OFV	134,18	136,58	131,66
Ayarlama oranı (AR)	0,5		Ortalama Süre(s)	19,28	19,28	19,28
Yeniden ayarlama oranı (RAR)	0,15		En iyi	134,18	136,58	131,66
Ceza katsayısı (PF)	1,2		En kötü	134,18	136,58	131,66
Maksimum iterasyon sayısı	100.000					



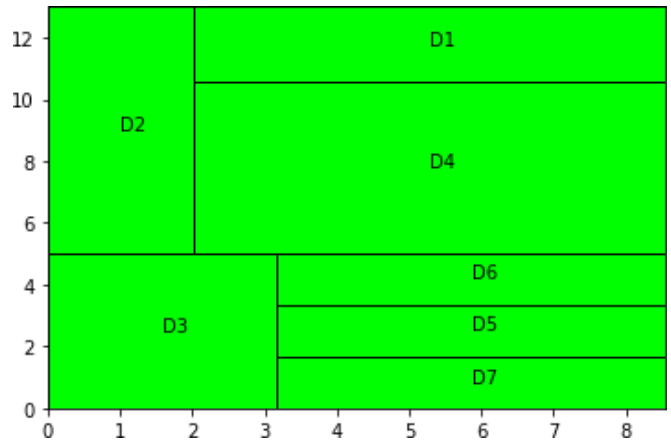
Şekil 4.9. Dikey FBS O7

OFV: 134,18



Şekil 4.10. Yatay FBS O7

OFV: 136,58



Şekil 4.11. Karışık FBS O7

OFV: 131,66

4.4.2. O8 problemi

Tablo 4.8. O8 Problemi için veri setleri

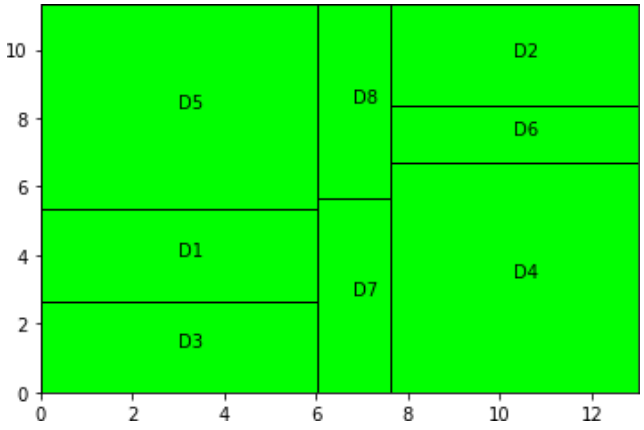
Departman sayısı	Tesis Eni	Tesis Boyu	Departmanların en-boy oranı
8	11,31	13	4

Tablo 4.9. Departmanlar arası veri akışı ve alanları

ff	1	2	3	4	5	6	7	8	Alanı
1	0	0	0	5	5	0	0	1	16
2	0	0	0	3	3	0	0	1	16
3	0	0	0	2	2	0	0	1	16
4	0	0	0	0	0	4	4	0	36
5	0	0	0	0	0	3	0	4	36
6	0	0	0	0	0	0	0	2	9
7	0	0	0	0	0	0	0	1	9
8	0	0	0	0	0	0	0	0	9

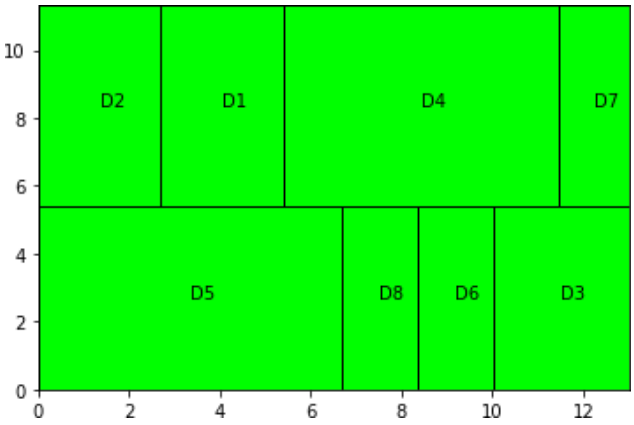
Tablo 4.10. Kullanılan parametreler ve sonuçlar

Parametreler			**Sonuçlar**			
Populasyon	9000			Dikey FBS	Yatay FBS	Karışık FBS
Önem oranı (CR)	0,9		Ortalama OFV	251,38	245,48	244,064
Ayarlama oranı (AR)	0,5		Ortalama Süre(s)	25,74	25,74	25,74
Yeniden ayarlama oranı (RAR)	0,15		En iyi	251,38	245,48	243,11
Ceza katsayısı (PF)	1,2		En kötü	251,38	245,48	245,5
Maksimum iterasyon sayısı	100.000					



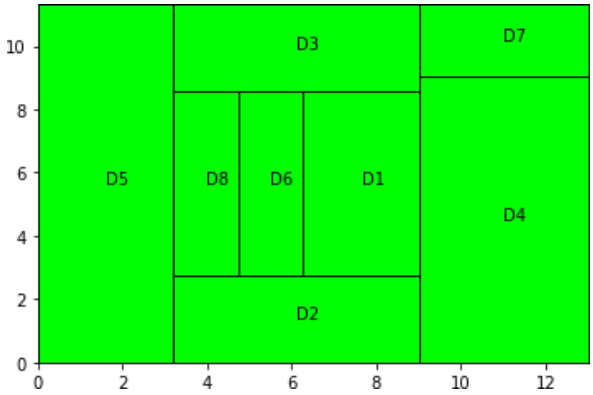
Şekil 4.12. Dikey FBS O8

OFV: 251,38



Şekil 4.13. Yatay FBS O8

OFV: 245,48



Şekil 4.14. Karışık FBS O8

OFV: 243,11

4.4.3. O9 problemi

Tablo 4.11. O9 Problemi için veri setleri

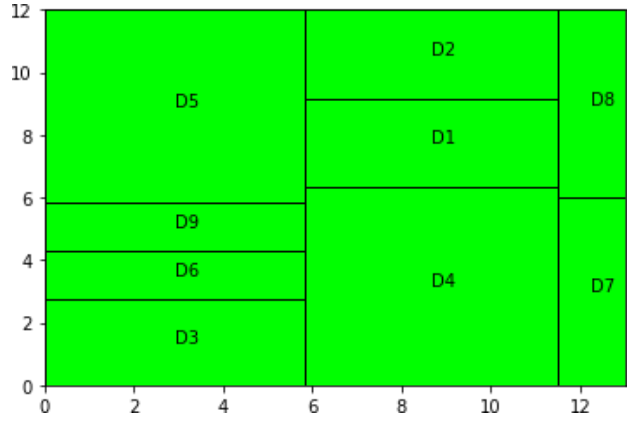
Departman sayısı	Tesis Eni	Tesis Boyu	Departmanların en-boy oranı
9	12	13	4

Tablo 4.12. Departmanlar arası veri akışı ve alanları

ff	1	2	3	4	5	6	7	8	9	Alanı
1	0	0	0	5	5	0	0	0	1	16
2	0	0	0	3	3	0	0	0	1	16
3	0	0	0	2	2	0	0	0	1	16
4	0	0	0	0	0	4	4	0	0	36
5	0	0	0	0	0	3	0	0	4	36
6	0	0	0	0	0	0	0	0	2	9
7	0	0	0	0	0	0	0	0	1	9
8	0	0	0	0	0	0	0	0	0	9
9	0	0	0	0	0	0	0	0	0	9

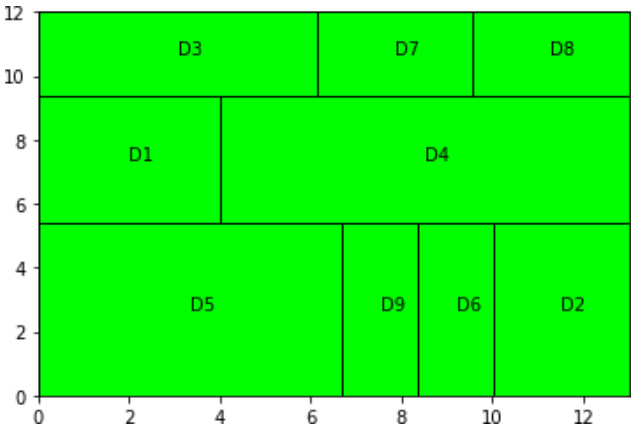
Tablo 4.13. Kullanılan parametreler ve sonuçlar

Parametreler			**Sonuçlar**			
Populasyon	9000			Dikey FBS	Yatay FBS	Karışık FBS
Önem oranı (CR)	0,9		Ortalama OFV	241,06	254,38	240,2805
Ayarlama oranı (AR)	0,5		Ortalama Süre(s)	141,3	141,3	141,3
Yeniden ayarlama oranı (RAR)	0,15		En iyi	241,06	254,38	236,14
Ceza katsayısı (PF)	1,2		En kötü	241,06	254,38	242,72
Maksimum iterasyon sayısı	1.000.000					



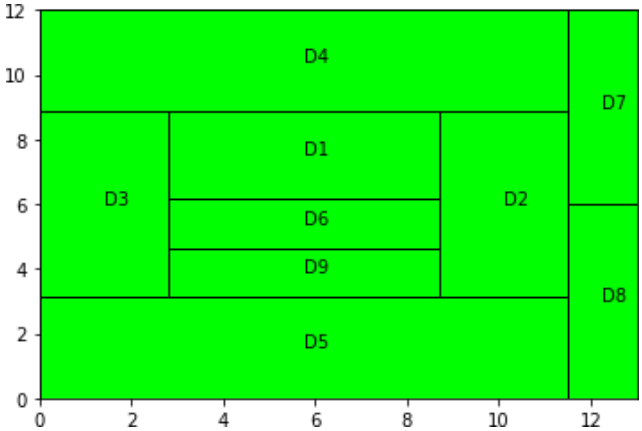
Şekil 4.15. Dikey FBS O9

OFV: 241,06



Şekil 4.16. Yatay FBS O9

OFV: 254,38



Şekil 4.17. Karışık FBS O9

OFV: 236,14

4.4.4. VC10a problemi

Tablo 4.14. VC10a Problemi için veri setleri

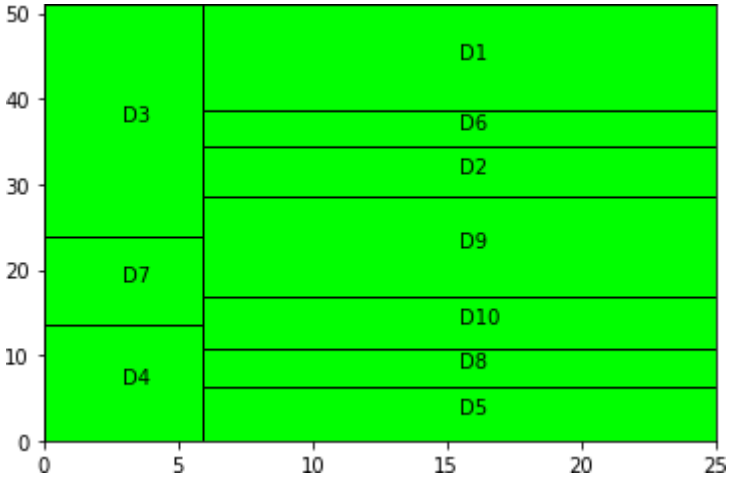
Departman sayısı	Tesis Eni	Tesis Boyu	Departmanların en-boy oranı
10	25	51	5

Tablo 4.15. Departmanlar arası veri akışı ve alanları

ff	1	2	3	4	5	6	7	8	9	10	Alanı
1	0	0	0	0	0	218	0	0	0	0	238
2	0	0	0	0	0	148	0	0	296	0	112
3	0	0	0	28	70	0	0	0	0	0	160
4	0	0	0	0	0	28	70	140	0	0	80
5	0	0	0	0	0	0	0	210	0	0	120
6	0	0	0	0	0	0	0	0	0	0	80
7	0	0	0	0	0	0	0	0	0	28	60
8	0	0	0	0	0	0	0	0	0	888	85
9	0	0	0	0	0	0	0	0	0	59	221
10	0	0	0	0	0	0	0	0	0	0	119

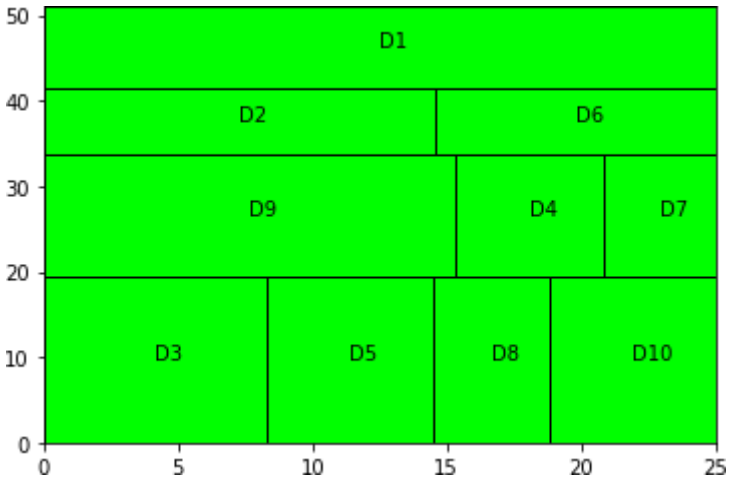
Tablo 4.16. Kullanılan parametreler ve sonuçlar

Parametreler			**Sonuçlar**			
Populasyon	9000			Dikey FBS	Yatay FBS	Karışık FBS
Önem oranı (CR)	0,9		Ortalama OFV	20140,35	21456,83	20786,4
Ayarlama oranı (AR)	0,5		Ortalama Süre(s)	671,75	671,75	671,75
Yeniden ayarlama oranı (RAR)	0,15		En iyi	20140,35	21456,83	18520
Ceza katsayısı (PF)	1,2		En kötü	20140,35	21456,83	22630
Maksimum iterasyon sayısı	1.000.000					



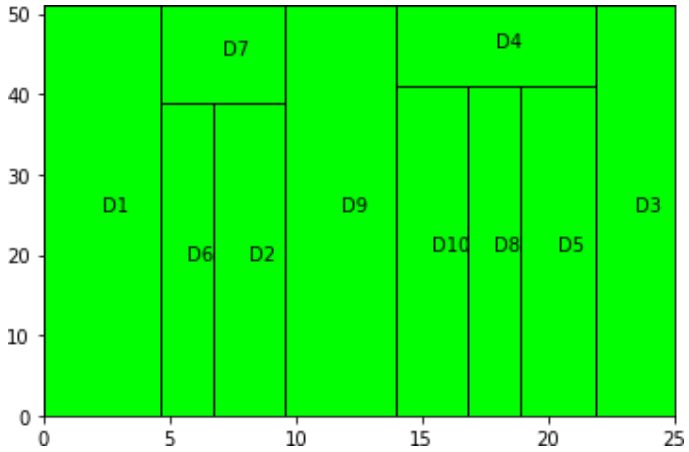
Şekil 4.18. Dikey FBS VC10a

OFV: 20140,35



Şekil 4.19. Yatay FBS VC10a

OFV: 21456,83



Şekil 4.20. Karışık FBS VC10a

OFV: 18520

4.4.5. AB20 problemi

Tablo 4.17. AB20 Problemi için veri setleri

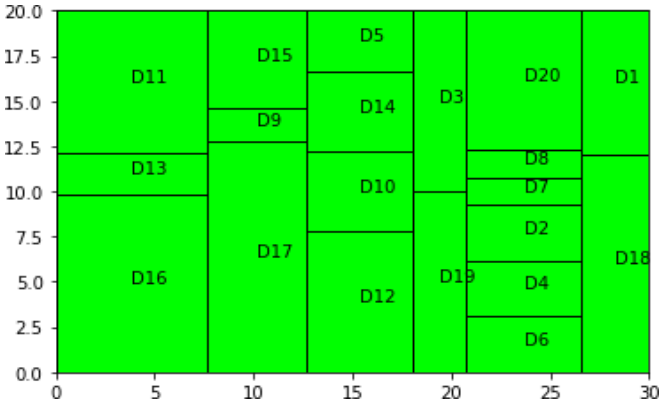
Departman sayısı	Tesis Eni	Tesis Boyu	Departmanların en-boy oranı
20	20	30	4

Tablo 4.18. Kullanılan parametreler ve sonuçlar

Parametreler			**Sonuçlar**			
Populasyon	9000			Dikey FBS	Yatay FBS	Karışık FBS
Önem oranı (CR)	0,9		Ortalama OFV	5769,25	5867,882	5633,1
Ayarlama oranı (AR)	0,5		Ortalama Süre(s)	1339,35	1339,35	1339,35
Yeniden ayarlama oranı (RAR)	0,15		En iyi	5283,64	5544,54	4972
Ceza katsayısı (PF)	1,2		En kötü	6109,25	6150,18	6305
Maksimum iterasyon sayısı	1.000.000					

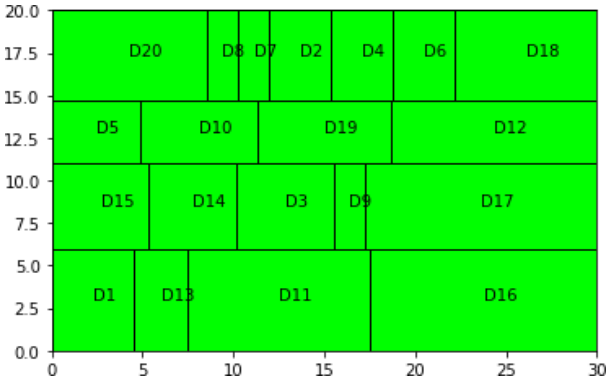
Tablo 4.19. Departmanlar arası veri akışı ve alanları

ff	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Alanı
1	0	1,8	1,2	0	0	0	0	0	0	1,04	1,12	0	0	1,2	0	0	0	0	0	0	27
2	1,8	0	0,96	24,45	0,78	0	13,95	0	1,2	1,35	0	0	0	0	0	0	0	0	6,9	0	18
3	1,2	0,96	0	0	0	2,21	0	0	3,15	3,9	0	0	0	13,05	0	0	0	0	13,65	0	27
4	0	24,45	0	0	1,08	5,7	7,5	0	2,34	0	0	1,4	0	0	0	0	0	1,5	15,75	0	18
5	0	0,78	0	1,08	0	0	2,25	1,35	0	1,56	0	0	0	0	1,35	0	0	0	0	0	18
6	0	0	2,21	5,7	0	0	6,15	0	0	0	0	0,45	0	0	0	0	0	1,05	0	0	18
7	0	13,95	0	7,5	2,25	6,15	0	24	0	1,87	0	0	0	0,96	0	0	0	1,65	0	3,75	9
8	0	0	0	0	1,35	0	24	0	0	0	0	0	0,6	0	0	0	0	0	7,5	33,45	9
9	0	1,2	3,15	2,34	0	0	0	0	0	0	0	0	0	7,5	0	0	7,5	0	0	0	9
10	1,04	1,35	3,9	0	1,56	0	1,87	0	0	0	0,36	12	0	18,6	1,92	0	0	0	5,25	0	24
11	1,12	0	0	0	0	0	0	0	0	0,36	0	2,25	0	3	0,96	22,5	0	0	0	0	60
12	0	0	0	1,4	0	0,45	0	0	0	12	2,25	0	0	0	1,65	0	15	0	8,4	0	42
13	0	0	0	0	0	0	0	0,6	0	0	0	0	0	8	1,04	6	0	0	0	0	18
14	1,2	0	13,05	0	0	0	0,96	0	7,5	18,6	3	0	8	0	9,75	0	0	0,9	0	0	24
15	0	0	0	0	1,35	0	0	0	0	1,92	0,96	1,65	1,04	9,75	0	0	5,25	0	0	0	27
16	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	12	0	0	0	75
17	0	0	0	0	0	0	0	0	7,5	0	0	15	0	0	5,25	12	0	0	7,5	0	64
18	0	0	0	1,5	0	1,05	1,65	0	0	0	0	0	0	0,9	0	0	0	0	4,65	0	41
19	0	6,9	13,65	15,75	0	0	0	7,5	0	5,25	0	8,4	0	0	0	0	7,5	4,65	0	0	27
20	0	0	0	0	0	0	3,75	33,45	0	0	0	0	0	0	0	0	0	0	0	0	45



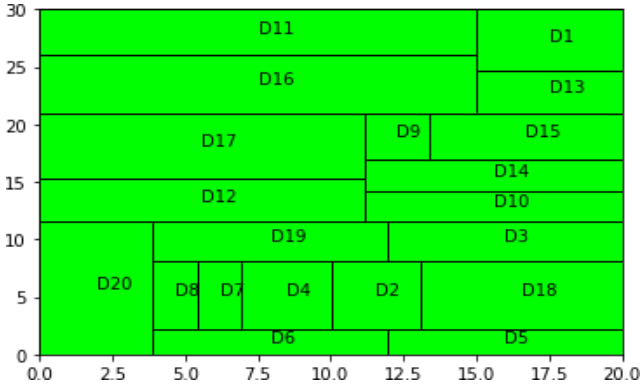
Şekil 4.21. Dikey FBS AB20

OFV: 5283,64



Şekil 4.22. Yatay FBS AB20

OFV: 5544,54



Şekil 4.23. Karışık FBS AB20

OFV : 4972

5. SONUÇ

Bu araştırmada kullanılan evrimsel algoritma, eşit olmayan departman alanlı tesislerin yerleşiminde genellikle kullanılan veriler kullanılarak denenmiştir. O7, O8, O9, VC10a ve AB20 aynı şekilde literatürden departman akışları, en-boy oranı ve departman alanları olarak bire bir alınmış olup bu algoritma çalıştırılmış ve sonuçlar raporlanmıştır.

O7 için bu algoritma STS tipi yerleşim gösterimi için 131,67 ve FBS tipi yerleşim içinse 136,58 sonuçlarını vermiştir. Bu sonuçların bilgisayarda ortalama işlem süresi 19,28 saniye sürmüştür. O7 için Castillo [12]çalışmasında en iyi tesis yerleşimini 131,58 olarak belirtmiştir. Castillo çalışmasında MIP kullandığı için 131,58'in en iyi sonuç olduğunu bilmekteyiz. O8 için kullanılan algoritma STS için 243,12 ve FBS tipi yerleşim için ise 245.48 sonuçlarını vermiştir. Bu sonuçlara ulaşma ortalama 26,57 saniye sürmüştür. Castillo çalışmasında O8 için 242,93 olarak en isi sonucu raporlamıştır. O9 için ise kullanılan algoritma STS için 236,13 ve FBS tipi yerleşim için ise 241,06 sonucunu vermiştir. Castillo çalışmasında O9 için en iyi yerleşimi 236,14 olarak raporlamıştır ve bu da algoritmanın verdiği STS sonucu ile aynıdır. Castillo çalışmasında VC10a problemini de çözmeyi denemiş ancak bilgisayarın süre kısıtlamasından dolayı en iyi sonucu elde edememiştir. Bu nedenle 10 departman ve daha fazlası için çözülüp bulunmuş bir en iyi sonuç bulunmamaktadır.

O7, O8 ve O9 problemleri için Castillo [12]'nin MIP formülasyonu ile ulaştığı sonuçlar ve kullanılan evrimsel algoritmanın sonuçları farklı olmasına rağmen oluşan tesis yerleşim şemaları aynıdır. Şemaların aynı olup çözüm değerlerinin farklı olmasının nedeni ise Castillo'nun çalışmasında doğrusal olmayan alan kısıtından dolayı her departmanın tam alan değerini bulamamasıdır. Ancak şu ana kadar bilgisayar tarafından çözülebilen 11 departmana kadar problemde evrimsel algoritmanın optimal tesis yerleşimini bulması bu algoritmanın gücünü ortaya koymaktadır.

VC10 probleminin literatürde iki farklı problem çeşidi bulunmaktadır. Birinci olarak her bir departmanın en kısa kenarının 1'den fazla olma gerekliliği bir diğeri ise her departmanın en-boy oranının 5'i geçmemesidir. Bu çalışmada geçen algoritmada en-boy oranı daha rahat kullanabildiğinden literatürde VC10a olarak geçen en-boy oranı kısıtı dikkate alınmıştır. Bu problemde STS yerleşimi olarak 18520,81 ve FBS yerleşim tipi olarak 21810,96 olarak en iyi yerleşim sonucu bulunmuştur. STS yerleşimiolarak Kang et al [51] 18522 sonucunu bildirmiş olup, FBS yerleşimi olarak ise Palomo-Romero [52] 20142 sonucunu bildirmiştir. STS sonucu en iyi

bildirilenden daha iyi olmasına rağmen FBS sonucu en iyi bildirilene göre daha kötü bir sonuç vermiştir.

AB20 literatürde en çok ele alınan problemlerden bir tanesidir. Bu problemde evrimsel algoritma STS tipi yerleşim için 4972.55 ve FBS tipi yerleşim için ise 5283.64 sonucunu bulmuştur. Yapılan literatür taramasında Kang et al [51] STS tipi yerleşimde 4959 en iyi yerleşim sonucunu bulmuş olup, FBS tipi yerleşimde ise 5286 ile Paluma-Romero[52] en iyi sonucu raporlamıştır. STS tipi yerleşimin Kang et al çalışmasından [51] farkı binde 1 olarak daha kötü sonuç vermiştir. FBS tipinde ise onbinde 5 daha iyi sonuç raporlanmıştır.

Genel olarak görüleceği üzere STS tipi yerleşimde yapılan arama, FBS tipinde yapılan aramaya göre daha iyi sonuçlar vermektedir. Bu aslında beklenen bir durumdur çünkü STS'nin aradığı problem uzayı FBS'nin arayabileceği çözüm uzayından oldukça fazladır. STS'nin elde edebileceği bazı tesis yerleşimplanlarını FBS ile elde etmek mümkün olmamaktadır.

Bu araştırmada kullanılan evrimsel algoritma literatürde de oldukça kullanılan 5 problemi aynı veri setleri ile alırken bir problemde ise boş alansız çözüm yaparak sonuçları vermektedir. Bu çalışmada kullanılan algoritma ne bir genetik algoritma ne de bir harmonik arama algoritmasıolarak tanımlanabilir. Aslında bu çalışmada iki tip algoritmanın da özellikleri harmanlanarak bir algoritma elde edilmiş ve sonuçlardan da görüleceği üzere olumlu ve iyi sonuçlar alınmıştır. Kullanılan algoritmada arama genetik algoritma ile benzemektedir ancak genetik algoritmadan ayıran en önemli özelliği yerel aramaya bir de yeniden ayarlama ihtimali getirmesidir. Getirilen yeniden ayarlama ile yerel arama daha da geliştirilmiş ve daha çok sonuç setini taramaktadır. Bu algoritmanın öbür çalışmalardan ayıran bir diğer özelliği ise hem STS tipi hem de FBS tipi çalışmayı algoritmada aynı anda oluşturması ve en iyi sonucu raporlarken hem STS için hem de FBS için farklı tesis yerleşimleri sonucu vermesidir. Algoritma da bu işlemtesis yerleşimlerini STS ya da FBS olarak tanımlayarak ve yapılan işlemleri STS veya FBS yerleşim tipini bozmayacak şekilde yapmasıyla olmaktadır.

Bu çalışmanın geliştirebileceği 3 farklı yön bulunmaktadır. Birincisi bu çalışmada uygun parametre seçimi Taguchi deney tasarımı metodu ile gerçekleştirilip, en iyi parametrelere bunun sonucunda karar verilmiştir. Bu tip deneytasarımının yerine Ful faktöriyel tasarım uygulanıp en iyi sonuçları verecek parametre bulunabilir. İkincisi ise problem seti olarak toplam departman alanı toplam tesis alanından küçük (kompaktlığı 1'den düşük olan) problemler dikkate alınmamıştır. Bu algoritma bu tip problemlere de uygulanabilecek şekilde düzenlenebilir. Üçüncü olarak ise yerleşimlerde değişmez departmanlar dikkate alınmamıştır. Yapılan bu çalışmada en-boy oranına uyduğu sürece departman boyutları değişebilir durumdadır. Ancak gerçek hayattakiuygulamalarda

bazı durumlarda departmanların boyutlarının sabit olması beklenebilir. Gelecekteki yapılacak çalışmalarda STS veya FBS yerleşim tipinin boyutları değişmez departmanlarla birlikte kullanılması incelenebilir.

KAYNAKLAR

[1] Koopmans, Tjalling C., and Martin Beckmann. “Assignment Problems and the Location of Economic Activities.” Econometrica, vol. 25, no. 1, Jan. 1957, p. 53. DOI.org (Crossref), https ://doi.org/10.2307/1907742.

[2] Hassan, Mohsen MD, and Gary L. Hogg. “A Review of Graph Theory Application to the Facilities Layout Problem.” Omega, vol. 15, no. 4, Jan. 1987, pp. 291–300. DOI.org (Crossref), https ://doi.org/10.1016/0305-0483(87)90017-X.

[3] Montreuil, Benoit. “A Modelling Framework for Integrating Layout Design and Flow Network Design.” Material Handling ’90, by Robert J. Graves et al., vol. 2, Springer Berlin Heidelberg, 1991, pp. 95–115. DOI.org (Crossref), https ://doi.org/10.1007/978-3-642-84356-3_8.

[4] Balakrishnan, Jaydeep, et al. “A Hybrid Genetic Algorithm for the Dynamic Plant Layout Problem.” International Journal of Production Economics, vol. 86, no. 2, Nov. 2003, pp. 107–20. DOI.org (Crossref), https ://doi.org/10.1016/S0925-5273(03)00027-6.

[5] Bozer, Yavuz A., et al. “An Improvement-Type Layout Algorithm for Single and Multiple-Floor Facilities.” Management Science, vol. 40, no. 7, 1994, pp. 918–32.

[6] Kulturel-Konak, Sadan, and Abdullah Konak. “A New Relaxed Flexible Bay Structure Representation and Particle Swarm Optimization for the Unequal Area Facility Layout Problem.” Engineering Optimization, vol. 43, no. 12, Dec. 2011, pp. 1263–87. DOI.org (Crossref), https ://doi.org/10.1080/0305215X.2010.548864.

[7] Kar Yan Tam. “Genetic Algorithms, Function Optimization, and Facility Layout Design.” European Journal of Operational Research, vol. 63, no. 2, Dec. 1992, pp. 322–46. DOI.org (Crossref), https ://doi.org/10.1016/0377-2217(92)90034-7.

[8] Shayan , E., and A. Chittilappilly. “Genetic Algorithm for Facilities Layout Problems Based on Slicing Tree Structure.” International Journal of Production Research, vol. 42, no. 19, Oct. 2004, pp. 4055–67. DOI.org (Crossref), https ://doi.org/10.1080/00207540410001716471.

[9] Meller, Russell D., and Kai-Yin Gau. “The Facility Layout Problem: Recent and Emerging Trends and Perspectives.” Journal of Manufacturing Systems, vol. 15, no. 5, Jan. 1996, pp. 351–66. DOI.org (Crossref), https ://doi.org/10.1016/0278-6125(96)84198-7.

[10] Meller, Russell D., et al. “Applying the Sequence-Pair Representation to Optimal Facility Layout Designs.” Operations Research Letters, vol. 35, no. 5, Sept. 2007, pp. 651–59. DOI.org (Crossref), https ://doi.org/10.1016/j.orl.2006.10.007.

[11] Lacksonen, Thomas A. “Static and Dynamic Layout Problems with Varying Areas.” The Journal of the Operational Research Society, vol. 45, no. 1, Jan. 1994, p. 59. DOI.org (Crossref), https ://doi.org/10.2307/2583951.

[12] Castillo, Ignacio, and Tapio Westerlund. “An ε-Accurate Model for Optimal Unequal-Area Block Layout Design.” Computers & Operations Research, vol. 32, no. 3, Mar. 2005, pp. 429–47. DOI.org (Crossref), https ://doi.org/10.1016/S0305-0548(03)00246-6.

[13] Nugent, Christopher E., et al. “An Experimental Comparison of Techniques for the Assignment of Facilities to Locations.” Operations Research, vol. 16, no. 1, Feb. 1968, pp. 150–73. DOI.org (Crossref), https ://doi.org/10.1287/opre.16.1.150.

[14] Lawler, Eugene L. “The Quadratic Assignment Problem.” Management Science, vol. 9, no. 4, 1963, pp. 586–99.

[15] Bazaraa, Mokhtar S., and Hanif D. Sherali. “Benders’ Partitioning Scheme Applied to a New Formulation of the Quadratic Assignment Problem.” Naval Research Logistics Quarterly, vol. 27, no. 1, Mar. 1980, pp. 29–41. DOI.org (Crossref), https ://doi.org/10.1002/nav.3800270104.

[16] Al-Khayyal, F., M. Goetschalckx, T. Van Voorhis. 2001. Alternative formulations and new solution algorithms for the facility layout problem. INFORMS Internat. Hawaii Conf. Maui, HI, June 17–20. (Also see An approach to solving the facility layout problem. Draft manuscript, February 3, 1997, School of Industrial and Systems Eng

[17] Frederick S. Hillier, Michael M. Connors, (1966) Quadratic Assignment Problem Algorithms and the Location of Indivisible Facilities. Management Science 13(1):42-57. http://dx.doi.org/10.1287/mnsc.13.1.42

[18] Seehof J.M. and Evans W.O., “Automated Layout Design Program”, The Journal of Industrial Engineering. Vol. 17. nº 12, pp. 690-695, (1967).

[19] Lee R.C. and Moore J.M., “CORELAP - Computerized Relationship Layout Planning”, The Journal of Industrial Engineering, vol. 18, nº. 3, pp. 195-200, (1967).

[20] Hassan M.M.D., Hogg G.L., Donald Smith, “SHAPE: A construction algorithm for area placement evaluation”. International Journal of Production Research, vol. 24, nº. 5, pp. 1283-1295, (1986).

[21] Armour G.C. and Buffa E.S., “A heuristic algorithm and simulation approach to relative location of facilities”, Management Science, vol. 9, nº. 2, pp. 294-309, (1963).

[22] Hillier F.S., “Quantitative tools for plant layout analysis”, The Journal of Industrial Engineering, vol. 14, nº 1, pp 33-40, (1963).

[23] Tompkins J.A. and Reed R., “An applied model for the facilities design problem”, International Journal of Production Research, vol. 14, nº. 5, pp. 583- 595, (1976).

[24] Bozer, Yavuz A., and Russell D. Meller. “A Reexamination of the Distance-Based Facility Layout Problem.” IIE Transactions, vol. 29, no. 7, July 1997, pp. 549–60. DOI.org (Crossref), https ://doi.org/10.1080/07408179708966365.

[25] Kirkpatrick S., Gelatt C and Vecchi, “Optimization by simulated annealing”, Sciencie. Vol. 220, nº. 4598, pp. 671-680, (1983).

[26] Burkard, R. E., & Rendl, F. (1984). A thermodynamically motivated simulation procedure for combinatorial optimization problems. European Journal of Operational Research, 17(2), 169–174.

[27] Rajesh Matai , S.P. Singh & M.L. Mittal (2013) Modified simulated annealing based approach for multi objective facility layout problem, International Journal of Production Research, 51:14, 4273-4288

[28] Glover F., Laguna M., Kelly J., "Tabu search for the multilevel generalized assignment problem". European Journal of Operational Research, vol. 82, nº 1, pp. 176-189, (1995).

[29] Skorin-Kapov, J. (1990). Tabu Search Applied to the Quadratic Assignment Problem. ORSA Journal on Computing, 2(1), 33–45.

[30] Kouvelis P., Chiang W., and Fitzsimmons J., "Simulated annealing for machine layout problems in the presence of zoning constraints". European Journal of Operational Researches, nº 57, pp. 203-223, (1992).

[31] Scholz, Daniel, et al. "STaTS: A Slicing Tree and Tabu Search Based Heuristic for the Unequal Area Facility Layout Problem." European Journal of Operational Research, vol. 197, no. 1, Aug. 2009, pp. 166–78. DOI.org (Crossref), https ://doi.org/10.1016/j.ejor.2008.06.028.

[32] Aiello, Giuseppe, et al. "A Multi Objective Genetic Algorithm for the Facility Layout Problem Based upon Slicing Structure Encoding." Expert Systems with Applications, vol. 39, no. 12, Sept. 2012, pp. 10352–58. DOI.org (Crossref), https ://doi.org/10.1016/j.eswa.2012.01.125.

[33] Gonçalves, José Fernando, and Mauricio G. C. Resende. "A Biased Random-Key Genetic Algorithm for the Unequal Area Facility Layout Problem." European Journal of

Operational Research, vol. 246, no. 1, Oct. 2015, pp. 86–107. DOI.org (Crossref),https ://doi.org/10.1016/j.ejor.2015.04.029.

[34] Bonabeau, Eric & Dorigo, Marco & Theraulaz, Guy. (2001). Swarm Intelligence : From Natural to Artificial Systems / E. Bonabeau, M. Dorigo, G. Theraulaz..

[35] Mahfound S. W and Goldberg D. E., "Parallel recombinative simulated annealing: A genetic algorithm". Parallel Computing, vol. 21, nº 1, pp. 1- 28, (1995).

[36] Katoch, Sourabh, et al. "A Review on Genetic Algorithm: Past, Present, and Future." Multimedia Tools and Applications, vol. 80, no. 5, Feb. 2021, pp. 8091–126. DOI.org (Crossref), https ://doi.org/10.1007/s11042-020-10139-6.

[37] Fox, B. R., and M. B. McMahon. "Genetic Operators for Sequencing Problems." Foundations of Genetic Algorithms, vol. 1, Elsevier, 1991, pp. 284–300. DOI.org (Crossref), https ://doi.org/10.1016/B978-0-08-050684-5.50021-5.

[38] Jebari, Khalid. (2013). Selection Methods for Genetic Algorithms. International Journal of Emerging Sciences. 3. 333-344.

[39] De Jong, K. (1988). Learning with genetic algorithms: An overview. Machine Learning, 3(2-3), 121–138.

[40] Brindle, A. (1981). Genetic algorithms for function optimization (Doctoral dissertation and Technical Report TR81-2). Edmonton: University of Alberta, Department of Computer Science.

[41] Chang-Yong Lee. “Entropy-Boltzmann Selection in the Genetic Algorithms.” IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), vol. 33, no. 1, Feb. 2003, pp. 138–49. DOI.org (Crossref), https ://doi.org/10.1109/TSMCB.2003.808184.

[42] Goldberg, D. E., & Deb, K. (1991). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. Foundations of Genetic Algorithms, 69–93.

[42] Soon, Gan Kim, et al. “A Comparison on the Performance of Crossover Techniques in Video Game.” 2013 IEEE International Conference on Control System, Computing and Engineering, IEEE, 2013, pp. 493–98. DOI.org (Crossref), https ://doi.org/10.1109/ICCSCE.2013.6720015.

[43] Davis, F. D. (1985). A Technology Acceptance Model for Empirically Testing New End-User Information Systems: Theory and Results. Massachusetts Institute of Technology.

[44] F. J. Burkowski, “Shuffle Crossover and Mutual Information,” Proceedings of the 1999 Congres on Evolutionary Computation, 1999, pp. 1574-1580.

[45] Oliver, I.M., Smith, D., Holland, J.R., Study of permutation crossover operators on the traveling salesman problem, In: Grefenstette, J. J. (ed.) Genetic Algorithms and Their Applications, Proceedings of the Second International Conference. Hillsdale, New Jersey: Lawrence Erlbaum, 1987, 224–230. 2.1.3

[46] Mudaliar, Dr. Devasenathipathi & Modi, N.K.. (2013). Unraveling Travelling Salesman Problem by genetic algorithm using m-crossover operator. International Conference on

Signal Processing, Image Processing and Pattern Recognition 2013, ICSIPR 2013. 1. 127-130. 10.1109/ICSIPR.2013.6497974.

[47] Piszcz, Alan & Soule, Terence. (2006). Genetic programming: Optimal population sizes for varying complexity problems. GECCO 2006 - Genetic and Evolutionary Computation Conference. 1. 953-954. 10.1145/1143997.1144166.

[48] Pelikan, Martin & Goldberg, David & Cantu-Paz, Erick. (2000). Bayesian Optimization Algorithm, Population Sizing, and Time to Convergence. 275-282.

[49] G. R. Harik, F. G. Lobo and D. E. Goldberg, "The compact genetic algorithm," in IEEE Transactions on Evolutionary Computation, vol. 3, no. 4, pp. 287-297, Nov. 1999, doi: 10.1109/4235.797971.

[50] Hassanat, Ahmad, et al. “Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach.” Information, vol. 10, no. 12, Dec. 2019, p. 390. DOI.org (Crossref), https ://doi.org/10.3390/info10120390.

[51] Kang, Sumin, and Junjae Chae. “Harmony Search for the Layout Design of an Unequal Area Facility.” Expert Systems with Applications, vol. 79, Aug. 2017, pp. 269–81. DOI.org (Crossref), https ://doi.org/10.1016/j.eswa.2017.02.047.

[52] Romero, David, and Adolfo Sánchez-Flores. “Methods for the One-Dimensional Space Allocation Problem.” Computers & Operations Research, vol. 17, no. 5, Jan. 1990, pp. 465–73. DOI.org (Crossref), https ://doi.org/10.1016/0305-0548(90)90051-8.

[53] Pablo Pérez-Gosende, Josefa Mula & Manuel Díaz-Madroñero (2021) Facility layout planning. An extended literature review, International Journal of Production Research, 59:12, 3777-3816, DOI: 10.1080/00207543.2021.1897176

[54] Jerzy Grobelny , Rafał Michalski , A novel version of simulated annealing based on linguistic patterns for solving facility layout problems, Knowledge-Based Systems (2017), doi: 10.1016/j.knosys.2017.03.001

[55] Chang, Mei-Shiang, and Ting-Chen Ku. "A Slicing Tree Representation and QCP-Model-Based Heuristic Algorithm for the Unequal-Area Block Facility Layout Problem." Mathematical Problems in Engineering, vol. 2013, 2013, pp. 1–19. DOI.org (Crossref),https ://doi.org/10.1155/2013/853586.

[56] Haktanirlar Ulutas, Berna, and Sadan Kulturel-Konak. "An Artificial Immune System Based Algorithm to Solve Unequal Area Facility Layout Problem." Expert Systems with Applications, vol. 39, no. 5, Apr. 2012, pp. 5384–95. DOI.org (Crossref), https ://doi.org/10.1016/j.eswa.2011.11.046.

[57] Komarudin, and Kuan Yew Wong. "Applying Ant System for Solving Unequal Area Facility Layout Problems." European Journal of Operational Research, vol. 202, no. 3, May 2010, pp. 730–46. DOI.org (Crossref), https ://doi.org/10.1016/j.ejor.2009.06.016.

[58] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," Multimed Tools Appl, vol. 80, no. 5, pp. 8091–8126, Feb. 2021, doi: 10.1007/s11042-020-10139-6.

[58] C. Friedrich, A. Klausnitzer, and R. Lasch, "Integrated slicing tree approach for solving the facility layout problem with input and output locations based on contour distance," European Journal of Operational Research, vol. 270, no. 3, pp. 837–851, Nov. 2018, doi: 10.1016/j.ejor.2018.01.001.

EKLER

EK1: Algoritmanın çalışması için gereken Python class'ları

```
import matplotlib.pyplot as plt
import random
import time
```

EK2: Algoritmada tanımlanan parametrelerin Python kod hali

```
    def___init_(self, fs=None, ord=None, ori=None, cost=None, aspectRatioDict=None,
    lr=None, ul=None, type=None):
        self.fs = fs
        self.ord = ord
        self.ori = ori
        self.cost = cost
        self.aspectRatioDict = aspectRatioDict
        self.lr = lr
        self.ul = ul
        self.type = type
class AdvanceLayoutConst():
    def__init_(self, mainDep, width, height, aspectRatio, p, cr, ar, rar, stop, u):
        self.mainDep = mainDep
        self.width = width
        self.height = height
        self.aspectRatio = {}
        self.p = p
```

```
        self.cr = cr
        self.ar = ar
        self.rar = rar
        self.stop = stop
        self.u = u
        self.worst_fbs_layout = None
        self.worst_sts_layout = None
        self.best_fbs_layout = None
        self.best_sts_layout = None
        self.layoutList = []
        self.sts_list = []
        self.fbs_list = []
        for index, element in enumerate(aspectRatio):
            self.aspectRatio[index + 1] = element
```

EK3: Listenin en kötü ve en iyi sonuçlu yerleşimi almak için Python kodu

```
 def driverFunc(self):
        for i in range(self.p):
            layout = self.randomizedLayout()
            if (layout.type == "FBS"):
                self.fbs_list.append(layout)
            else:
                self.sts_list.append(layout)
            self.layoutList.append(layout)
        self.worst_fbs_layout = self.getWorstLayout(self.fbs_list)
```

```
        self.worst_sts_layout = self.getWorstLayout(self.sts_list)
        print("STS: ", len(self.sts_list))
        print("FBS: ", len(self.fbs_list))

    def getWorstLayout(self, layoutList):
        max = None
        layout = Layout()
        for i in layoutList:
            if (not max or i.cost > max):
                max =  i.cost
                layout = i
        return layout
    def getBestLayout(self, layoutList):
        min = None
        layout = Layout()
        for i in layoutList:
            if (not min or i.cost < min):
                min = i.cost
                layout = i
        return layout
```

EK4: Yeni çocuk yaratmak için Python fonksiyonu

```
    def  considerationLayout(self):
        # Randomly select 2 layouts
        child = Layout()
```

```
choice = random.choices([0, 1], weights = [0.01,0.99], k = 1)
if (choice[0] == 0):
    # print("Should be: ", "FBS")
    temp = random.sample(self.fbs_list, 2)
    child.type = "FBS"
else:
     # print("Should be: ", "STS")
    temp = random.sample(self.sts_list, 2)
    child.type = "STS"

l1, l2 = temp[0], temp[1]
# print("l1:", l1.fs);
# print("l2:", l2.fs);
 # Choice is used for selecting 1
cut = random.choice(list(range(1, len(l1.fs))))
child.fs = l1.fs[:cut] + l2.fs[cut:]
child.ord = l1.ord[:cut] + l2.ord[cut:]
child.ori = l1.ori[:cut] + l2.ori[cut:]
self.replaceRecurring(child.fs, l1.fs, l2.fs, cut)
self.replaceRecurring(child.ord, l1.ord, l2.ord, cut)
# print("\n")
# print("fs: ", child.fs)
# print("odr: ", child.ord)
# print("ori: ", child.ori)
# Let 0 represent no adjustment and 1 represent adjustment to be done.
```

```
    # Weights is for setting the probability
    adjustment = random.choices([0, 1], weights=[1-self.ar, self.ar], k=1)
    if (adjustment[0] == 1):
        self.doAdjustment(child)
    cost, aspectRatioDict, lr, ul = self.calculateCost(
        child.fs, child.ord, child.ori)
    child.aspectRatioDict = aspectRatioDict
    child.lr = lr
    child.ul = ul
    # child.type = type
    # Penalty Check
    cost = cost * self.penaltyCheck(child)
    child.cost = cost
    return child
    # print("OFV = ", cost)
    # if ((not self.best_layout) or (cost < self.best_layout.cost)):
    #     self.best_layout = child
    # self.layoutList.append(child)
# Function to replace recurring elements from child
def replaceRecurring(self, l, l1, l2, cut):
    temp = []
    for index, element in enumerate(l):
        if (element in temp):
            n = self.nonCommon(l, l1, l2, cut)
            if (n):
```

```
                l[index] = n
            else:
                temp.append(element)
    # Function to get value from parent layout which is not present in child layout
    def nonCommon(self, child, l1, l2, cut):
        for j in l1[cut:]:
            if j not in child:
                return j
        for k in  l2[:cut]:
            if k not in child:
                return k
        return None
```

EK5: Ayarlama için Python fonksiyonu

```
    def doAdjustment(self, child):
        # Well we can have all weights as 1 also but no problem in that one too
        adjustment = random.choices([1, 2, 3, 4], weights=[
                                0.25, 0.25, 0.25, 0.25], k=1)
        if (adjustment[0] == 1):
            # selecting any 2 index to swap department values
            index1, index2 = map(int, random.sample(
                list(range(0, len(child.fs))), 2))
            child.fs[index1], child.fs[index2] = child.fs[index2], child.fs[index1]
        elif(adjustment[0] == 2):
            index1, index2 = map(int, random.sample(
```

```
        list(range(0, len(child.ord))), 2))
    child.ord[index1], child.ord[index2] = child.ord[index2], child.ord[index1]
elif(adjustment[0] == 3):
    # selecting any 2 indexs, 1st the value to be displaced and 2nd the position
    index1, index2 = map(int, random.sample(
        list(range(0, len(child.fs))), 2))
    val = child.fs.pop(index1)
    child.fs.insert(index2, val)
else:
    # selecting any 2 indexs, 1st the value to be displaced and 2nd the position
    index1, index2 = map(int, random.sample(
        list(range(0, len(child.ord))), 2))
    val = child.ord.pop(index1)
    child.ord.insert(index2, val)
```

EK6: Yeniden ayarlama için Python fonksiyonu

```
# Re-Adjustment(1: Do readjustment, 0: Don't)
readjustment = random.choices([0, 1], weights=[1, self.rar], k=1)
if (readjustment[0] == 1 and child.type != "FBS"):
    for index, element in enumerate(child.ori):
        if (element == 0):
            child.ori[index] = 1
        else:
            child.ori[index] = 0
return
```

EK7: Ceza fonskiyonu için Python kodu

```
def penaltyCheck(self, newLayout):
    v = 0
    for i in newLayout.fs:
        if (newLayout.aspectRatioDict[i] > self.aspectRatio[i]):
            v += 1
    return (self.u**v)
```

EK8: Rastgele çocuk yaratmak için Python kodu

```
def randomizedLayout(self):
    fs = list(range(1, len(self.mainDep)+1))
    odr = list(range(1, len(self.mainDep)))
    random.shuffle(fs)
    random.shuffle(odr)
    ori = []
    choice = random.choices([0, 1], weights=[0.9,0.1], k=1)


    if (choice[0] == 1):
        # print("Should be: ", "FBS")
        cut = random.choice(list(range(0, len(self.mainDep)-1)))
        for i in range(len(self.mainDep)-1):
            if (i<cut):
```

```
                ori.append(1)
            else:
                ori.append(0)

        # print("cur, list : ", cut, ori)
    else:
        # print("Should be: ", "STS")
        ori = [random.randrange(0, 2) for i in range(len(self.mainDep)-1)]
    # print("fs: ", fs)
    # print("odr: ", odr)
    # print("ori: ", ori)
    cost, aspectRatioDict, lr, ul = self.calculateCost(fs, odr, ori)
    if (choice[0] == 1):
        new = Layout(fs, odr, ori, cost, aspectRatioDict, lr, ul, "FBS")
    else:
        new = Layout(fs, odr, ori, cost, aspectRatioDict, lr, ul, "STS")
    # Penalty Check
    cost = cost * self.penaltyCheck(new)
    new.cost = cost
    return new
    # print("OFV = ", cost)
    # if ((not self.best_layout) or (cost < self.best_layout.cost)):
    #     self.best_layout = new
    # self.layoutList.append(new)
def intersection(self, lst1, lst2):
```

```
    lst3 = [value for value in lst1 if value in lst2]
    return lst3
```

EK9: Amaç fonksiyonun hesaplamak için Python kodu

```
def calculateCost(self, fs, odr, ori):
    LC, RC, lr, ul, R = [], [], [], [], []
    ff = [[0,1.8,1.2,0,0,0,0,0,0,1.04,1.12,0,0,1.2,0,0,0,0,0,0],
[1.8,0,0.96,24.45,0.78,0,13.95,0,1.2,1.35,0,0,0,0,0,0,0,0,6.9,0],
[1.2,0.96,0,0,0,2.21,0,0,3.15,3.9,0,0,0,13.05,0,0,0,0,13.65,0],
[0,24.45,0,0,1.08,5.7,7.5,0,2.34,0,0,1.4,0,0,0,0,0,1.5,15.75,0],
[0,0.78,0,1.08,0,0,2.25,1.35,0,1.56,0,0,0,0,1.35,0,0,0,0,0],
[0,0,2.21,5.7,0,0,6.15,0,0,0,0,0.45,0,0,0,0,0,1.05,0,0],
[0,13.95,0,7.5,2.25,6.15,0,24,0,1.87,0,0,0,0.96,0,0,0,1.65,0,3.75],
[0,0,0,0,1.35,0,24,0,0,0,0,0,0.6,0,0,0,0,0,7.5,33.45],
[0,1.2,3.15,2.34,0,0,0,0,0,0,0,0,0,7.5,0,0,7.5,0,0,0],
[1.04,1.35,3.9,0,1.56,0,1.87,0,0,0,0.36,12,0,18.6,1.92,0,0,0,5.25,0],
[1.12,0,0,0,0,0,0,0,0,0.36,0,2.25,0,3,0.96,22.5,0,0,0,0],
[0,0,0,1.4,0,0.45,0,0,0,12,2.25,0,0,0,1.65,0,15,0,8.4,0],
[0,0,0,0,0,0,0,0.6,0,0,0,0,0,8,1.04,6,0,0,0,0],
[1.2,0,13.05,0,0,0,0.96,0,7.5,18.6,3,0,8,0,9.75,0,0,0.9,0,0],
[0,0,0,0,1.35,0,0,0,0,1.92,0.96,1.65,1.04,9.75,0,0,5.25,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,6,0,0,0,12,0,0,0],
[0,0,0,0,0,0,0,0,7.5,0,0,15,0,0,5.25,12,0,0,7.5,0],
[0,0,0,1.5,0,1.05,1.65,0,0,0,0,0,0,0.9,0,0,0,0,4.65,0],
```

[0,6.9,13.65,15.75,0,0,0,7.5,0,5.25,0,8.4,0,0,0,0,7.5,4.65,0,0],

[0,0,0,0,0,0,3.75,33.45,0,0,0,0,0,0,0,0,0,0,0,0]]

```
    n = len(fs)
    LC.insert(0, fs)
    for i in range(0, n):
        lr.append((self.width, 0))
        ul.append((0, self.height))
    j = 0
    while j != n-1:
        k = n-2
        while k >= 0:
            try:
                if (fs[odr[j]-1] in LC[k]):
                    R = LC[k]
                    break
                elif (fs[odr[j]-1] in RC[k]):
                    R = RC[k]
                    break
                k = k-1
            except IndexError:
                k = k-1
        temp = odr[j]
```

```
if (j == 0):
    LC[j] = self.intersection(fs[0: temp], R)
else:
    LC.insert(j, self.intersection(fs[0: temp], R))
RC.insert(j, self.intersection(fs[temp:], R))
sumA = 0
for i in LC[j]:
    sumA = sumA + self.mainDep[i-1]
if ori[j] == 0: # Vertical
    for i in LC[j]:
        lr[i-1] = ((ul[i-1])[0] +
                (sumA/((ul[i-1][1]-lr[i-1][1]))), (lr[i-1])[1])
    for i in RC[j]:
        ul[i-1] = ((ul[i-1])[0] +
                (sumA/((ul[i-1][1]-lr[i-1][1]))), (ul[i-1])[1])
elif ori[j] == 1: # Horizontal
    for i in LC[j]:
        lr[i-1] = ((lr[i-1])[0],  (ul[i-1])[1] -
                (sumA/((lr[i-1])[0] - (ul[i-1])[0])))
    for i in RC[j]:
        ul[i-1] = ((ul[i-1])[0], (ul[i-1])[1] -
                (sumA/((lr[i-1])[0] - (ul[i-1])[0])))
j = j + 1
```

EK10: Yerleşimin bilgilerini içeren Python kodu

```
aspectRatioDict = {}
        cordinates = []
        for i in range(0, n):
            w = (lr[i][0]-ul[i][0])
            h = (ul[i][1]-lr[i][1])
            ar = max([w, h]) / min([w, h])
            # print("\n")
            # print(f"Aspect Ratio for department {fs[i]}: ",ar)
            # print(f"Area for department {fs[i]}: ", w*h)
            aspectRatioDict[fs[i]] = ar
            cordinates.append(
                [(ul[i][0], lr[i][1]), w, h])
        cost = 0
        for j in range(0, n):
            for i in range(0, n):
                dist = abs(lr[i][0]/2+ul[i][0]/2 - lr[j][0]/2-ul[j][0]/2) + \
                    abs(ul[i][1]/2+lr[i][1]/2 - ul[j][1]/2-lr[j][1]/2)
                cost = cost + dist * ff[i][j]
        # type = self.getType(cordinates, ori)
        # print("Type: ", type)
        # print("\nOFV =", cost)
        return (cost, aspectRatioDict, lr, ul)
```

EK11: En iyi amaç fonksiyonuna sahip yerleşimi göstermek için Python kodu

```
def display(self):
    print("The best FBS layout is: ")
    print("\nDepartment Sequence: ", self.best_fbs_layout.fs)
    print("Cut Sequence: ", self.best_fbs_layout.ord)
    print("Cut Code: ", self.best_fbs_layout.ori)
    for i in range(0, len(self.best_fbs_layout.fs)):
        w = (self.best_fbs_layout.lr[i][0]-self.best_fbs_layout.ul[i][0])
        h = (self.best_fbs_layout.ul[i][1]-self.best_fbs_layout.lr[i][1])
        print("height,width: ", h, w)
        ar = max([w, h]) / min([w, h])
        print("\n")
        print(
            f"Aspect Ratio for department {self.best_fbs_layout.fs[i]}: ", ar)
        print(f"Area for department {self.best_fbs_layout.fs[i]}: ", w*h)
    print("\nWith the OFV of ", self.best_fbs_layout.cost)
    print("Here's the layout:")
    plt.figure()
    plt.xlim([0, self.width])
    plt.ylim([0, self.height])
    for i in range(len(self.best_fbs_layout.fs)):
        h = self.best_fbs_layout.ul[i][1]-self.best_fbs_layout.lr[i][1]
        w = self.best_fbs_layout.lr[i][0]-self.best_fbs_layout.ul[i][0]
        rectangle = plt.Rectangle(
```

```
            (self.best_fbs_layout.ul[i][0], self.best_fbs_layout.lr[i][1]), w, h, fc='green',
ec="red")
        plt.text(self.best_fbs_layout.lr[i][0]/2+self.best_fbs_layout.ul[i][0]/2,
                 self.best_fbs_layout.ul[i][1]/2+self.best_fbs_layout.lr[i][1]/2, 'dpt%s' % (i+1))
        plt.gca().add_patch(rectangle)
    plt.title("FBS Layout")
    plt.savefig('FBS Layout')
    # STS
    print("\n\nThe best STS layout is: ")
    print("\nDepartment Sequence: ", self.best_sts_layout.fs)
    print("Cut Sequence: ", self.best_sts_layout.ord)
    print("Cut Code: ", self.best_sts_layout.ori)
    for i in range(0, len(self.best_sts_layout.fs)):
        w = (self.best_sts_layout.lr[i][0]-self.best_sts_layout.ul[i][0])
        h = (self.best_sts_layout.ul[i][1]-self.best_sts_layout.lr[i][1])
        print("height,width: ", h, w)
        ar = max([w, h]) / min([w, h])
        print("\n")
        print(
            f"Aspect Ratio for department {self.best_sts_layout.fs[i]}: ", ar)
        print(f"Area for department {self.best_sts_layout.fs[i]}: ", w*h)
    print("\nWith the OFV of ", self.best_sts_layout.cost)
    print("Here's the layout:")
    plt.figure()
    plt.xlim([0, self.width])
```

```
        plt.ylim([0, self.height])
        for i in range(len(self.best_sts_layout.fs)):
            h = self.best_sts_layout.ul[i][1]-self.best_sts_layout.lr[i][1]
            w = self.best_sts_layout.lr[i][0]-self.best_sts_layout.ul[i][0]
            rectangle = plt.Rectangle(
                (self.best_sts_layout.ul[i][0], self.best_sts_layout.lr[i][1]), w, h, fc='green', ec="red")
            plt.text(self.best_sts_layout.lr[i][0]/2+self.best_sts_layout.ul[i][0]/2,
                     self.best_sts_layout.ul[i][1]/2+self.best_sts_layout.lr[i][1]/2, 'dpt%s' % (i+1))
            plt.gca().add_patch(rectangle)
        plt.title("STS Layout")
        plt.savefig('STS Layout')
```

EK12: Problemin parametrelerinin belirtildiği Python kodu

```
if(__name__== "_main_"):
    layout = AdvanceLayoutConst(
        mainDep=[27,18,27,18,18,18,9,9,9,24,60,42,18,24,27,75,64,41,27,45],
        width=20,
        height=30,
        aspectRatio=[4, 4, 4, 4, 4, 4, 4,4,4,4,4,4,4,4,4,4,4,4,4,4],
        p=5000,
        cr=0.6,
        ar=.7,
        rar=.3,
        stop=1000000,
```

u=1.2)

EK13: Geçen zamanı göstermek için kullanılan Python kodu

```
tic = time.perf_counter()
tak = time.perf_counter()
print(f"Execute the program in {tak - tic:0.4f} seconds")
```