EŞİT OLMAYAN ALANLI DİNAMİK TESİS YERLEŞİMİ PROBLEMİ İÇİN BİR GENETİK ALGORİTMA YAKLAŞIMI

Yüksek Lisans Tezi

Melis ÜREM

Eskişehir 2019

EŞİT OLMAYAN ALANLI DİNAMİK TESİS YERLEŞİMİ PROBLEMİ İÇİN BİR GENETİK ALGORİTMA YAKLAŞIMI

Melis ÜREM

YÜKSEK LİSANS TEZİ

Endüstri Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Nil ARAS

Eskişehir

Eskişehir Teknik Üniversitesi

Lisansüstü Eğitim Enstitüsü

Ağustos 2019

JÜRİ VE ENSTİTÜ ONAYI

Melis ÜREM'in "Eşit Olmayan Alanlı Dinamik Tesis Yerleşimi Problemi İçin Bir Genetik Algoritma Yaklaşımı" başlıklı tezi 23/08/2019 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Eskişehir Teknik Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği"nin ilgili maddeleri uyarınca Endüstri Mühendisliği Anabilim dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Jüri Üyeleri	**Unvanı-Adı-Soyadı**	**İmza**
Üye (Tez Danışmanı)	: Doç. Dr. Nil ARAS	
Üye	: Prof. Dr. Berna ULUTAŞ	
Üye	: Dr. Öğr. Üyesi Banu GÜNER	

Prof. Dr. Murat TANIŞLI
Lisansüstü Eğitim Enstitüsü Müdürü

ÖZET

EŞİT OLMAYAN ALANLI DİNAMİK TESİS YERLEŞİMİ PROBLEMİ İÇİN BİR GENETİK ALGORİTMA YAKLAŞIMI

Melis ÜREM

Endüstri Mühendisliği Anabilim Dalı
Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Ağustos 2019

Danışman: Doç. Dr. Nil ARAS

Tesis yerleşim problemleri, çözümü zor problemler olmakla birlikte, çıktıları işletmelerin üretim süreçleri ve karlılıkları açısından önemlidir. Birim maliyeti; hammadde, enerji, işgücü gibi faktörlerin yanında, taşıma maliyetleri de etkilemektedir. Son yıllarda yaşanan rekabet etkisi ile müşteri taleplerine uygun maliyetle ve zamanında cevap vermek önem kazanmıştır. Bu nedenle, tesislerin; ürün, süreç, üretim teknolojisi, işgücü ve yönetim sistemlerindeki her türlü değişikliğe hızlı uyum sağlama gerekliliği doğmuştur. Tesis yerleşiminin, talep değişiklikleri ve bu değişikliklerin gerçekleştiği dönemler göz önüne alınarak, dinamik bir şekilde planlanması gerekmektedir.

Talep değişikliğinin olmadığı durumlarda bile çözümü zor olan bu problemler, işin içine dinamiklik boyutu girdiğinde, daha da zorlaşmaktadır. Zor problemlerin çözümünde optimuma yakın sonuçlar veren meta-sezgisel yöntemler kullanılmaktadır. Çalışmada, bu yöntemlerden, tesis yerleşim problemleri için iyi sonuçlar verdiği bilinen genetik algoritmalar kullanılarak bir üretim işletmesinin bir bölümünde eşit olmayan alanlı iş istasyonlarının tesis yerleşimi, sabit bir yerleşim şekli üzerinde baştan ele alınmıştır. C# ile probleme özel bir program hazırlanmış olup, tesis için yerleşim planı önerisi alınmıştır. Elde edilen çözüm mevcut durum ile kıyaslandığında %39 oranında bir iyileşme sağlandığı görülmektedir.

Anahtar Sözcükler: Tesis yerleşimi problemi, Dinamik tesis yerleşimi problemi, Genetik algoritmalar, Eşit olmayan alanlı

ABSTRACT

A GENETİC ALGORITHM APPROACH FOR UNEQUAL AREA DYNAMIC FACILITY LAYOUT PROBLEM

Melis ÜREM

Department of Industrial Engineering Program
Eskişehir Technical University, Institute of Graduate Studies, August 2019

Supervisor: Doç. Dr. Nil ARAS

Facility layout problems, although difficult to solve, problem's outputs are important in terms of production processes and profitability of enterprises. In addition to factors such as raw materials, energy, labor; transportation costs because of plant layout also affect the unit cost. In recent years, due to the competitive effect experienced in the production sectors, it has become important to respond to customer demands in a timely, and inexpensive manner. Therefore, the facilities must adapt quickly to any changes as product, process, production technology, labor and management systems. The plant layout needs to be planned dynamically, taking into consideration these changes and change periods.

These problems are difficult enough to solve even in the absence of dynamics, it becomes more difficult when the dimension of dynamism enters. To solve these difficult problems, meta-heuristic methods are used which give close to optimum results. In this study, genetic algorithms, which are known to give good results for facility layout problems, are used. The facility layout of workstations with unequal area in a department has been re-planned on a fixed layout. A problem specific program was prepared by using C #. The facility layout plan suggestion was obtained and the current situation was compared with the situation after improvement. It is seen that 39% cost improvement is achieved.

Keywords: Facility layout problem, Dynamic facility layout problem, Genetic algorithms, Unequal area

TEŞEKKÜR

Yüksek lisans tez çalışmamın yönlendirilmesi ve yürütülmesi aşamasında bilgi ve tecrübelerini esirgemeyen kıymetli danışmanım Doç. Dr. Nil ARAS'a, ihtiyaç duyduğum her an fikirlerini ve zamanlarını benimle paylaşan değerli meslektaşlarım ve mesai arkadaşlarım Ünsal ERBEDEN ve Erhan KÖKEN'e, kilometrelerce uzaklıktan, bunaldığım her konuda beni rahatlatan, yüreklendiren okul arkadaşlarım ve meslektaşlarım Cihan AYGÜN, Hande SARAÇOĞLU ve Mehmet KORANA'ya, tez çalışmalarımın en yoğun döneminde gerçekleştirdiğim iş değişimi sürecinde henüz tanışmış olmama rağmen yöneticim Suat Hayri BEKİRCAN'a destekleri ve anlayışları için,

Beni hayata hazırlayan, her zaman güçlü olmayı, zorluklarla mücadele etmeyi öğreten, sayelerinde kendimi çok şanslı saydığım, ilk öğretmenlerim olan canım annem ve babama, eğitim hayatımın en yakın şahidi, birlikte öğrenip, büyüdüğüm sevgili kız kardeşim Zeynep'e bu süreçte bana göstermiş oldukları sabır ve anlayıştan ötürü,

Teşekkür ediyorum.

Melis ÜREM

Ağustos 2019

23/08/2019

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; bu çalışmanın Eskişehir Teknik Üniversitesi tarafından kullanılan "bilimsel intihal tespit programı"yla tarandığını ve hiçbir şekilde "intihal içermediğini" beyan ederim. Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

Melis ÜREM

İÇİNDEKİLER

Sayfa

TABLOLAR DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

SİMGELER DİZİNİ

a_i : Tesisin En Boy Oranı

A_{tijl} : t Döneminde i İş İstasyonunun j Konumundan l Konumuna Atanması Maliyeti

C_{tijkl} : t Döneminde i İş İstasyonu ile k İş İstasyonu Arasındaki Taşımaların Maliyeti

d_{jl} : j Konumunun l Konumuna Olan Mesafesi

dx : İş İstasyonunun x Koordinatına Paralel Olan Uzunluğu

dx_i : i İş İstasyonunun x Koordinatına Paralel Olan Uzunluğu

dx_j : j İş İstasyonunun x Koordinatına Paralel Olan Uzunluğu

dy : İş İstasyonunun y Koordinatına Paralel Olan Uzunluğu

dy_i : i İş İstasyonunun y Koordinatına Paralel Olan Uzunluğu

dy_j : j İş İstasyonunun y Koordinatına Paralel Olan Uzunluğu

f_{tik} : t Döneminde i İş İstasyonundan k İş İstasyonuna Malzeme Akış Sayısı

h_i : Tesisin Yüksekliği

n : İş İstasyonlarının Sayısı

Pc : Çaprazlama Oranı

Pm : Mutasyon Oranı

t : Dönem Sayısı

w_i : Tesisin Genişliği

α : Elit Gen Sayısı

KISALTMALAR DİZİNİ

DP	:	Dinamik Programlama
DTYP	:	Dinamik Tesis Yerleşim Problemi
GA	:	Genetik Algoritma
GAMS	:	General Algebraic Modelling System
GTB	:	Geliştirilmiş Tavlama Benzetimi
MP	:	Matematiksel Programlama
İSG	:	İş Sağlığı ve Güvenliği
KAP	:	Kareli Atama Problemi
KKA:	:	Karınca Kolonisi Algoritması
STYP	:	Statik Tesis Yerleşim Problemi
TA	:	Tabu Arama
TB	:	Tavlama Benzetimi
TYP	:	Tesis Yerleşimi Problemi

1. GİRİŞ

Günümüz üretim işletmelerinde, gelişen üretim sistemleri nedeni ile piyasalardaki rekabet artmış, buna paralel olarak müşterilerin ürün ve hizmetlerden beklentileri yükselmiştir. İşletmelerin sürekliliğini sağlayabilmeleri için, gün geçtikçe daha kaliteli ve özellikli ürünleri üretip, daha ucuz fiyata müşterilere sunması önem kazanmıştır. Ürün fiyatları, kâr oranı göz önüne alınarak değil, rekabet koşullarına göre belirlendiğinden, işletmelerin sürdürülebilir olması, elde edilen kâr oranının artması ya da korunması için, olabildiğince az maliyetli üretim yapma zorunluluğu ortaya çıkmıştır.

Üretim sürecinde birim üretim maliyetini oluşturan unsurlar göz önüne alındığında en önemli unsurları hammadde, enerji ve işçilik oluşturmaktadır. Bu unsurlar içerisinde ürün üzerinde herhangi bir değer yaratmayan katma değersiz işlerin belirlenmesi ve ortadan kaldırılması, üretim maliyetlerinin azaltılmasında önemli rol oynamaktadır. Bu katma değersiz işlerin başında ürünlerin taşınması gelmektedir. Üretim süreci içerisinde, ürünün oluşmasına katkı sağlayan operasyonların tamamlanması için, ürünlerin bir iş istasyonundan diğerine taşınması söz konusudur. Bu taşımaların ürün üzerinde herhangi bir katma değeri olmadığı gibi, makine/ekipman ya da insan sayesinde gerçekleştirilen taşımalar ürünün işçilik ve enerji gibi birim maliyet unsurlarını artırmakta, ürün maliyetini yükseltmektedir.

Tesis yerleşimi problemi (TYP), bir tesis içindeki bölümlerin bazı amaçlara göre en yüksek faydayı sağlayacak şekilde düzenlenmesinin hedeflendiği problemdir. "Çalışmalarda, malzeme taşıma maliyetinin, işletme maliyetinin yüzde 20 ile 50'sini, ürünün toplam üretim maliyetinin ise yüzde 15 ile 70'ini oluşturduğunu gösterdiği belirtilmektedir [1]." Malzeme taşıma maliyetleri, tesis içerisinde bulunan bölümler arasında hangi miktarda malzeme aktarıldığına ve bu bölümler arasında bulunan mesafelere bağlı olarak belirlenmektedir.

Tüketici pazarı sürekli değişkenlik gösterdiğinden, üreticilerin rekabetçi olmalarını talep etmektedir. Bu nedenle üreticilerin verimli çalışmaları ve üretim tesislerinin ürün talebindeki değişikliklere hızlı adapte olmaları gerekmektedir. Üretim yapan işletmeler rekabetçi olabilmek için, pazarda yaşanan bu hareketliliğe hızlı cevap vermek, yani dinamik olmak zorundadır. Sözlükte "dinamik" sözcüğü, "genellikle sürekli ve üretken aktivite veya değişim" olarak belirtilmektedir [2]. TYP'lerdeki dinamiklik ise, probleme zaman boyutunun eklenmesi ile ortaya çıkmaktadır.

Müşteri taleplerindeki dalgalanmalar ve üretilen ürün çeşitlerinin değişmesi, ya da üretim süreçlerindeki güncellemeler nedeni ile iş akışlarında ya da iş istasyonlarında değişiklik yapılması gereksinimi ortaya çıkabilmektedir. Pazarın dinamikliği sonucunda ürün taleplerinde yaşanan dalgalanmalar, ürünlerin üretim süreci içerisinde izlediği rotayı ya da bölümler arasındaki akış miktarlarını etkilemektedir. Bölümler arası malzeme akış miktarı, ürüne duyulan ihtiyacın ortadan kalkması ve bu nedenle başka bir ürünün devreye alınması, ürünün tasarımında değişiklik yapılması, üretim teknolojisi ya da metodunda farklılaşma, üretim miktarlarında farklılaşma gibi pek çok nedene dayalı olarak değişiklik gösterebilmektedir. Farklı dönemlerde, malzeme akışı dalgalanmaları sonucu malzeme taşıma maliyetleri de değişkenlik göstermektedir. Bu durumlarda tesis içerisinde bulunan bölümlerin yerleşim planlarının yeniden gözden geçirilmesi ve düzenlenmesi gerekmektedir. Bu problemi çözmek için ise "Dinamik Tesis Yerleşim Problemi (DTYP)" kullanılmaktadır.

Çalışmada, dinamik bir ortamda eşit olmayan alanlı iş istasyonlarının bulunduğu düzensiz şekilli bir tesisin yerleşim planlaması, belirli yerleşim koşulları altında DTYP yaklaşımı ile, meta-sezgisel yöntemlerden olan Genetik Algoritma (GA) yönteminden faydalanılarak gerçekleştirilmiştir. Problemde, üretim tesisinde alınan kararlar ve talep dalgalanmaları nedeni ile zamanla ürün gruplarına göre üretim adetlerinde değişiklik yaşanmaktadır. Toplam üretim hacminde bir grup ürünün üretim hacmi azalırken, diğer ürün grubunun üretim hacmi artmaktadır.

Literatürde GA'lar ile çözülmüş DTYP'lere rastlanmaktadır. Bunların çoğunluğunu eşit alanlı iş istasyonlarının TYP'leri oluşturmaktadır. Eşit olmayan alanlı iş istasyonları için az da olsa literatürde örnekler bulunmaktadır. Bu çalışmada, diğer çalışmalardan farklı olarak, sabit bir yerleşim şekli üzerinde, eşit olmayan alanlı iş istasyonlarının ağırlık merkezlerinin bu sabit yerleşim şeklinin tam ortasından geçen bir hat üzerine denk gelecek şekilde sırasıyla yan yana dizilmesi için çözüm araştırılmıştır. Üretim gereklilikleri, işletmede kullanılan taşıma sistemi ve iş istasyonlarının kolay yer değiştirmesi için böyle sabit bir yerleşim şekli planlanmıştır.

Bu bölümden sonra gelen ikinci bölümde TYP, üçüncü bölümde DTYP ve DTYP ile ilgili mevcut çalışmalar, dördüncü bölümde GA, beşinci bölümde çalışmaya konu olan DTYP probleminin çözümü için gerçekleştirilen uygulama, altıncı ve son bölümde ise sonuç ve önerilerden bahsedilecektir.

2. TESİS YERLEŞİMİ

Üretim faaliyetleri ekonomik sistemin temelini oluşturan yapıtaşlarından en önemlisidir. Üretim süreci içerisinde tesislerde, emek, sermaye, hammadde, yöntem ve bilgi gibi girdilerin, imalat, montaj, işleme gibi süreçlerden geçerek dönüşüme uğraması ile ürün, yani çıktılar elde edilmektedir. Bir işletmenin temel varoluş nedeni göz önüne alındığında üretim sürecinin ana amacı, girdilerin verimli bir şekilde kullanılmasıyla birlikte, kaliteli ve düşük maliyetli ürünlerin üretilip en yüksek karın elde edilmesidir.

Bir ürünün birim maliyeti incelendiğinde, hammadde, işçilik, enerji, amortisman giderleri gibi pek çok unsur ile karşılaşılmaktadır. Tesis yerleşimi, üretim süreçleri içerisindeki operasyonların verimli bir şekilde gerçekleştirilmesi için önem teşkil etmektedir. Tesis planlaması ürün maliyetini etkileyen faktörlerden biridir. İyi planlanmış bir tesis yerleşimi ile ürün maliyeti düşürülebilmektedir. Bu da günümüz rekabet koşullarında kar oranını artırabilmek için çok önemli bir faktördür.

Üretimi gerçekleştirilen her bir ürünün kendine özgü bir üretim rotası bulunmaktadır. Her bir ürün tesis içerisinde yer alan tüm süreçlerden geçmek zorunda olmadığı gibi, ürün türüne göre süreçlerin sırası da değişiklik gösterebilmektedir. Ürünün birim maliyeti içerisinde yer alan taşıma maliyeti, üretim süreci içerisinde ürünün izlediği rotaya bağlı olarak değişiklik gösterebilmektedir. Tesis planlaması, üretim tesisinin etkin kapasite kullanım oranının artırılması ve tesis içerisinde hareket edecek malzeme ve işgücünün hareket miktarını en az seviyeye indirmek için önemli bir faktördür. Uygulanabilecek olan en iyi yerleşim planı ile tesis içerisinde oluşabilecek kargaşa, ürünlerin karışık bir rota izlemesi nedeni ile üretim süreçlerinin uzaması, ürünlerin tesis içerisinde kaybolması, dağınık yapılan yerleşim nedeni ile gereksiz alanlar için ortam iklimlendirmeye harcanan enerji kayıpları, taşıma ve yürümeler gibi pek çok olumsuzluğun da önüne geçilebilmektedir.

2.1. Tesis Yerleşiminin Amaçları

Tesis yerleşim planlamasının amaçlarından en önemlileri aşağıdaki gibidir;

- Malzeme, sermaye, insan gibi üretim girdilerinden en yüksek faydayı sağlamak,
- İşletme içerisindeki malzeme taşıma maliyetini en aza indirmek,
- Malzeme ve üretim kalitesini güvence altına almak,
- Toplam üretim süresini azaltmak,

- Mevcut alanları verimli kullanarak tesis ve donanım yatırımını azaltmak,
- Üretim süreçleri ve ekipmanların yeniden düzenlenmesi esnekliğini sağlamak,
- Çalışanların güven içerisinde, zorlanmadan, ergonomik şekilde çalışmasını sağlamak,
- Üretim süreçlerinin iş akışına göre standart iş talimatı ve süresine uygun yapılmasını sağlamak.

Tüm bu amaçlar göz önünde bulundurularak tesis içi yerleşim planlaması yapılmaktadır. Ancak bu amaçların hepsine tam olarak ulaşmak gerçek üretim işletmelerinde neredeyse imkânsızdır. En fazla faydanın sağlanabilmesi için çözüm yöntemleri üzerinde çalışmalar devam etmekte, bu sırada amaçların dengelendiği en iyiye yakın çözümler elde edilebilmektedir.

Yeni bir tesis kurulurken tesis içi yerleşim planlaması yapılmasının yanı sıra, yeni alan ihtiyaçlarının doğması, üründeki değişiklikler, üretim süreçlerinde ve teknolojilerinde yaşanan değişiklikler nedeni ile mevcut tesisin tamamen baştan düzenlenmesi, mevcut tesiste küçük değişiklikler yapılması ya da tesisin taşınması gibi durumlarda da tesis içi planlama yapılmaktadır. Çeşitli amaçlar için gerçekleştirilen tesis yerleşim planlaması problemleri, temel olarak aşağıdaki gibi 4 başlık altında toplamıştır;

- Mevcut tesis düzeninde küçük değişikliklerin yapılması,
- Mevcut tesis düzeninin yeniden ele alınması,
- Mevcut tesislere taşınma,
- Yeni bir tesisin kurulması [3].

2.2. Tesis Yerleşimini Etkileyen Faktörler

Bir tesisin yerleşim planlamasını etkileyen faktörler; malzeme, makine, insan, hareket, bekleme, hizmet, bina ve değişim olarak sınıflanmaktadır.

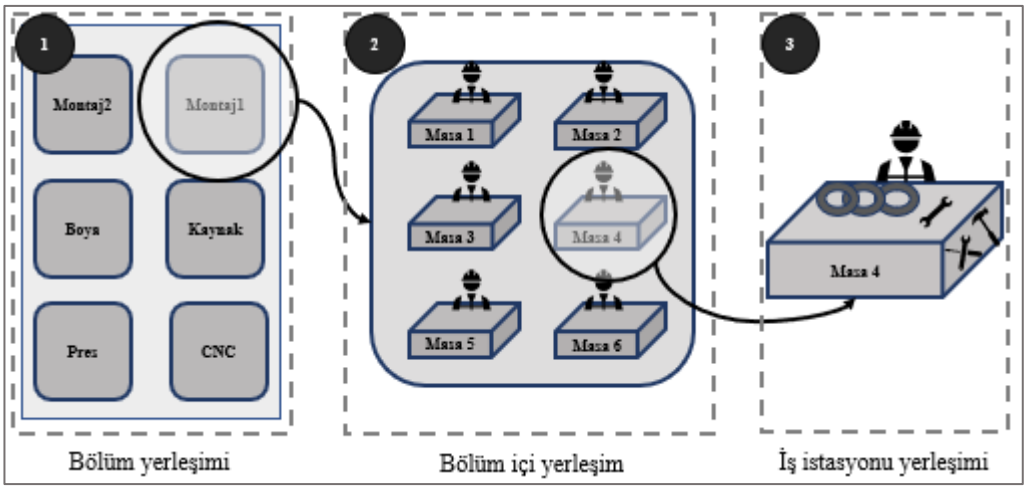
- Malzeme: Tasarım, değişiklik, miktar, gerekli işlemler ve bunların sırası.
- Makine: Üretim araçları, takımlar ve bunların kullanımı.
- İnsan: Gözetim, denetim ve hizmet.
- Hareket: Bölüm içi ve bölümler arası taşıma, değişik işlemler, depolamalar ve muayeneler.
- Bekleme: Sürekli ve geçici depolamalar ve gecikmeler.
- Hizmet: Bakım, muayene vb. hizmetler.

- Bina: Binanın iç ve dış özellikleri, kullanım özellikleri ve donanım.
- Değişim: Genişleme, esneklik. [3]

Tesis yerleşimi planlaması yapılırken, tesiste herhangi fiziksel bir değişiklik yapılmadan önce planlama ve analiz çalışmaları gerçekleştirilir. Tezgâh ve ekipmanlar bir defa yerlerine yerleştirildikten sonra yapılacak olan yeniden yerleştirme maliyetinin yüksek olması nedeni ile doğru karar vermek ve ardından aksiyon almak önemlidir. Öncelikli olarak problemin tanımlanması ya da ihtiyacın belirlenmesi aşaması tamamlanır. Problem tanımlanıp, ihtiyaç belirlendikten sonra, gerekli bilgilerin bir araya getirilmesi ile analiz çalışmaları gerçekleştirilir. İş akışı, ürün tasarımı, çalışma saatleri, makine kapasiteleri, gerekli işgücü, talep miktarları gerek görülen bilgiler arasında yer almaktadır. Sonrasında çeşitli çözüm önerileri değerlendirilerek en uygun olanın seçilmesinin ardından karar uygulamaktadır.

Bir tesisin toplam ürün taşıma maliyeti, işletmede üretilen ürün gruplarının üretim adetleri, ürünlerin izlediği rota, bir ürünü bir metre taşıma maliyeti, ürünün bir iş istasyonundan diğer iş istasyonuna taşınma mesafesi gibi pek çok faktöre bağlı olarak değişmektedir.

Tesis içi yerleşim planlaması yapılırken öncelikle tesis içerisinde yer alan bölümlerin yerleri alan ihtiyaçları göz önünde bulundurularak belirlenir. Tesisin kuruluş amacı olan üretimi gerçekleştirecek üretim bölümlerinin yerleri belirlendikten sonra, yardımcı işletmeler, idari ofisler, tuvaletler ve sosyal alanların yerleri üretim ile olan ilişkileri doğrultusunda planlanır. Bölümlerin yerleri belirlendikten sonra, bölüm içi makine ve ekipmanların yerleşim planlaması yapılır. Bu aşamada, üretim bölümleri arasında akışın nerede başladığı, hangi yöne doğru ilerlediği, sürecin sonunda ürünlerin hangi bölüme gideceği ve iş istasyonları arasındaki akışlar göz önüne alınarak toplam taşıma mesafesinin en küçüklenmesi amaçlanarak yerleşim yapılmaktadır. Üçüncü ve son olarak, her bir iş istasyonunun, kullanılacak olan ekipman ve el aletlerinin yerleşimi, ergonomik faktörler dikkate alınarak düzenlemesi yapılmaktadır. Şekil 2.1.'de bölümlerin, ekipmanların ve iş istasyonlarının tesis içi yerleşim planı aşamaları örnek üzerinde gösterilmektedir. Önce bölümlerin alanları belirlenip yerleştirilmiş, ardından bölüm içi yerleşimler yapılmıştır. Son olarak her bir iş istasyonu için ayrı ayrı yerleşim planı çalışması gerçekleştirilmiştir.



Şekil 2.1. *Tesis içi yerleşim planlama aşamaları*

2.3. Tesis Yerleşimi Karakteristikleri

Tesis yerleşimini etkileyen pek çok karakteristik bulunmaktadır. Yerleşime karar vermek için tesiste hangi tür üretim sisteminin kullanıldığı, ürün çeşitliliği, ürün özellikleri, süreç gereklilikleri, iş istasyonları ya da bölümler arası akışlar, tesisin ve iş istasyonlarının fiziksel ve boyutsal özellikleri gibi pek çok unsurun değerlendirilmesi gerekmektedir. Bu unsurlara göre yerleşim planlarının ne şekilde sınıflandırıldığı bu bölümde detaylı olarak anlatılmaktadır.

2.3.1. Üretim sistemlerine göre tesis yerleşim modelleri

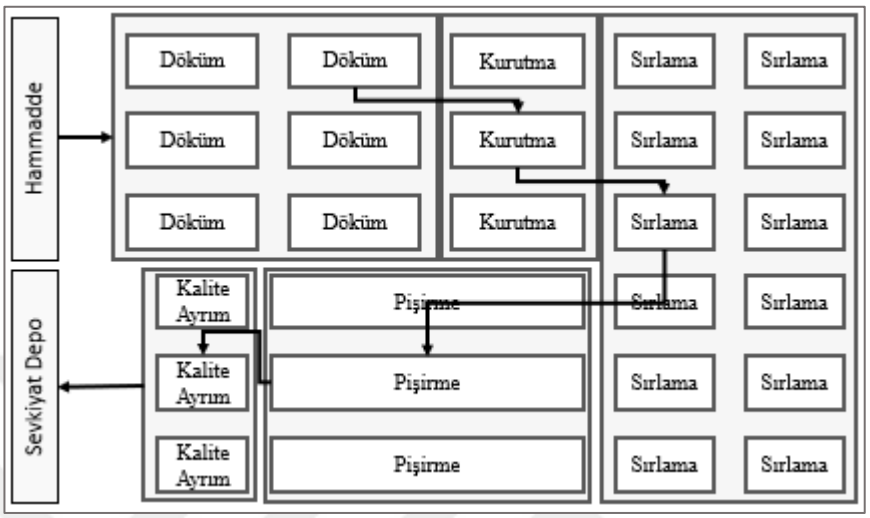
Tesis planlaması yapılırken akış tipine karar verildikten sonra makine ve ekipmanların yerleştirilmesi gerçekleştirilir. Yerleşim yapılırken genellikle ekipman ve işgücünün verimli kullanılması, stokların azaltılması, üretimin hızlanması ve taşımaların azalması amaçlanmaktadır.

Tesis yerleşim modelleri; sürece göre yerleşim, ürüne göre yerleşim, sabit pozisyonlu ürüne göre yerleşim, hücresel yerleşim ve birleşik yerleşim olmak üzere beş ana başlık altında incelenmektedir [4].

2.3.1.1. Sürece göre yerleşim

En çok tercih edilen yerleşim biçimidir. Makine ve ekipmanlar gerçekleştirdikleri fonksiyona göre benzer işi yapanlar bir arada konumlanacak şekilde yerleştirilmektedir. Ürün çeşitliliğinin fazla olması ve ürünlerin üretim hacimlerinin düşük olması durumunda tercih edilmektedir. Ekipmanlar, pres, torna, freze, kaynak, boyama vb. gibi işlemlere

göre gruplandırılmaktadır. Her bir ürün kendi prosesi özelinde bir işlemden çıkıp diğer işleme geçiş yapmaktadır. Ürünün izlediği rotalar genellikle uzun ve her bir ürün için farklı olmaktadır. Şekil 2.2.'de sürece göre yerleşim örneği gösterilmektedir.



Şekil 2.2. *Sürece göre yerleşim planı*

Sürece göre yerleşimin faydaları;

- Makine, ekipman kullanım oranı yüksektir.
- Makine ve çalışan planlaması yapılırken oldukça esneklik sağlar.
- Farklı işler nedeni ile iş ilgi çekicidir.
- Uzmanlaşma sağlanır.
- Üretilen tüm ürünler için ayrı makineye gerek olmayacağından daha az yatırım gerektirir.

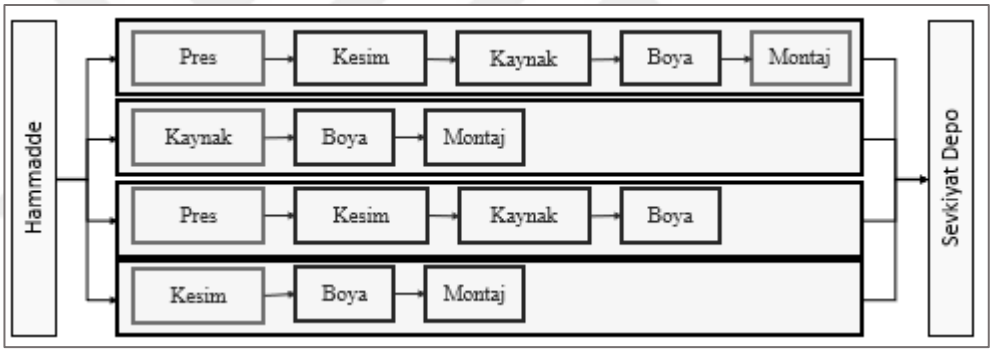
Sürece göre yerleşimin kusurları;

- Malzeme taşımalarında uzun mesafeler nedeni ile taşıma maliyeti artar.
- Malzeme taşımada otomasyon sistemleri kullanılması zordur ve malzeme taşıma işlemleri genellikle verimsizdir.
- Üretilen ürünler, üretim sahası içerisinde karışıklığa neden olur.
- Ara stok miktarları fazladır.
- Bir ürünün üretim süresi uzun olur.
- Tesis içerisindeki taşımanın sistemleşememesi ve ara stoklar gibi nedenler İş Sağlığı ve Güvenliği (İSG) risklerini artırır.
- Üretim planlama yapmak karmaşıktır.

- Ayar süreleri ve çalışan eğitimleri uzun sürer [5].

2.3.1.2. Ürüne göre yerleşim

Her bir ürünün kendine özgü bir iş akışı bulunmaktadır. İş akışının gerektirdiği rotanın esas alınarak makine ve ekipmanların düzenlendiği yerleşim biçimine ürüne göre yerleşim denilmektedir. Bir ürün türünün üretim adedinin diğer ürünlere kıyasla daha yüksek olması durumunda bu yerleşim tercih edilmektedir. Ürüne göre yerleşim kararı vermek için makine yatırımı yapılmadan önce çok yönlü ve detaylı bir ön çalışma yapılması gerekmektedir. Sonradan ortaya çıkabilecek bir değişiklik ihtiyacı daha büyük maliyetlere yol açabilmektedir. Ayrıca makineler sadece ilgili ürüne özel kullanılacağı için verimleri göz önünde bulundurulmalıdır. Şekil 2.3.'te ürüne göre yerleşim örneği bulunmaktadır.



Şekil 2.3. *Ürüne göre yerleşim planı*

Ürüne göre yerleşimin faydaları;

- Sade ve otomasyonun kullanılabildiği hatlardır.
- Üretim süreleri oldukça kısadır.
- Ürünlerin akış süreleri kısa ve taşıma mesafeleri ve maliyetleri düşüktür.
- Yarı mamul stokları azdır.
- Üretim tesisi daha sade olduğundan İSG açısından daha güvenlidir.
- Üretim planlama yapmak kolaydır.
- Niteliksiz işgücü kullanımına olanak sağlar.

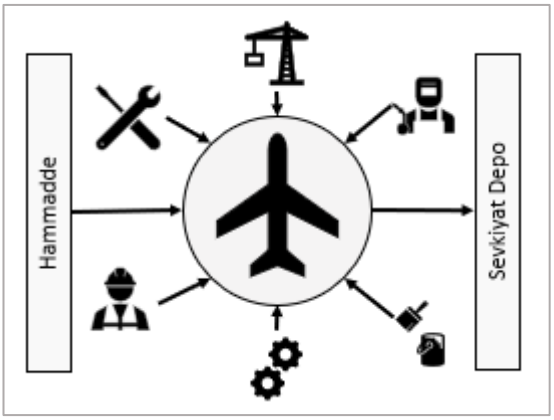
Ürüne göre yerleşimin kusurları;

- Üretim süreci esnek değildir.
- Hat içerisinde bir makinenin arızalanması tüm hattı durdurur.

- Ürün tasarımında ya da süreçte yaşanabilecek bir değişiklik hattın tamamen yeniden düzenlenmesine neden olabilir.
- Yüksek yatırım gerektirir.
- Üretim hızını darboğaz yaratan makine belirler, bu da diğer makinelerin verimini olumsuz yönde etkileyebilir.
- İşçi ve makine için zamanlamalar önemlidir, zamana uyulmazsa üretim aksayabilir.
- İşler monoton olduğundan bir süre sonra ilgi çekiciliğini kaybetmektedir [5].

2.3.1.3. Sabit pozisyonlu ürüne göre yerleşim

Sabit pozisyonlu ürüne göre yerleşim, hacimce büyük, ağır ve taşınması zor ürünlerin üretimi söz konusu olduğunda tercih edilmektedir. Proje tipi yerleşim olarak da adlandırılmaktadır. Üretilen ürünler sabit konumlarda dururken, çalışanların ve üretimde kullanılacak makine ve ekipmanın üretim noktasına gelmesi ile üretim gerçekleşmektedir. Uçak, gemi gibi büyük ulaşım araçlarının üretimi, binaların inşaatı bu tip üretime örnek gösterilebilir. Bir temel üzerinde bina inşa edilirken kullanılacak olan çimento, tuğla, beton, boya, boru, demir, mobilya gibi malzemeler inşaat alanına getirilir. Burada proje planına göre aşamalar halinde malzemeler ve ekipmanlar kullanılarak inşaat tamamlanır. Şekil 2.4.'te uçak üretiminin tesis yerleşimi görülmektedir.



Şekil 2.4. *Sabit pozisyonlu ürüne göre yerleşim*

Sabit pozisyonlu ürüne göre yerleşimin faydaları;

- Yerleşim tipi, ürün tasarımı ve üretim adedi konusunda oldukça esnektir.

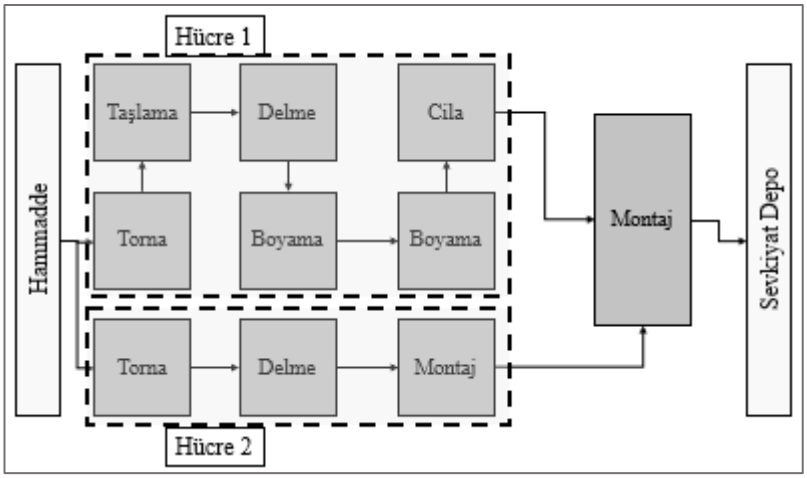
- Taşıma sayısı ve mesafesi az olduğundan taşıma maliyetleri düşüktür.
- Ekip çalışması, çalışanların sorumluluk bilincini geliştirir.
- Makine için sermaye yatırımı azdır. Ekipmanlar genellikle kiralanır.
- İş zenginleştirme ve işgücü yetkinliklerinin artırılmasına olanak sağlar.

Sabit pozisyonlu ürüne göre yerleşimin kusurları;

- Ekipman ve işçilerin taşınması yüksek maliyetlidir.
- Genellikle geniş alan ihtiyacı vardır.
- Nitelikli çalışana ihtiyaç duyulur.
- Üretim planlaması yapılırken yüksek koordinasyon gerektirir [5].

2.3.1.4. Hücresel yerleşim

Sürece göre ve ürüne göre yerleşimin birleşimidir. İki biçimin faydalı özelliklerini bünyesinde barındırdığı gibi, sakıncalarını da taşımaktadır. Hücresel yerleşimde, aynı işlemlerden geçen ürün grupları belirlenmekte, bunların aynı hücrelerde üretilmesi için yerleşim planı yapılmaktadır. Bu plan sonucunda daha kısa akışlar ve düşük taşıma maliyeti elde edilmektedir. Hücresel yerleşime örnek Şekil 2.5.'te gösterilmektedir.



Şekil 2.5. *Hücresel yerleşim planı*

Hücresel yerleşimin faydaları;

- Ürün taşıma rotaları sade ve taşıma mesafeleri kısadır.
- Üretim süreleri kısadır.
- Akış maliyeti azdır.
- Ekip çalışmasına uygundur.

Hücresel yerleşimin kusurları;

- Talebe göre makine kapasite kullanım oranları düşebilir.
- Sürece göre yerleşim ile kıyaslandığında esnek değildir.
- Tampon stok tutma gerekliliği oluşabilir.
- Hatlarda iş dengelemek zordur.
- Nitelikli işgücü ihtiyacı bulunur [5].

2.3.1.5. Birleşik yerleşim

Gerçek hayatta tesislerin yerleşimleri incelendiğinde, bir tesis içerisinde, yukarıda sayılmış olan yerleşim türlerinden sadece bir tanesine tek başına rastlanmaz. İşletmede üretilen ürün çeşitliliği, parti büyüklükleri gibi etkenlere göre uygun olan yerleşim türleri farklı oranlarda olsa da bir arada kullanılmaktadır. Kamyon frenleri üretilen bir fabrika ele alındığında, yerleşim yüksek oranda sürece göre düzenlenmiş olsa da fren pabuçları ya da piston yuvası üretimlerinin ayrı hücrelerde gerçekleşmesi, birleşik üretime örnek verilebilir.

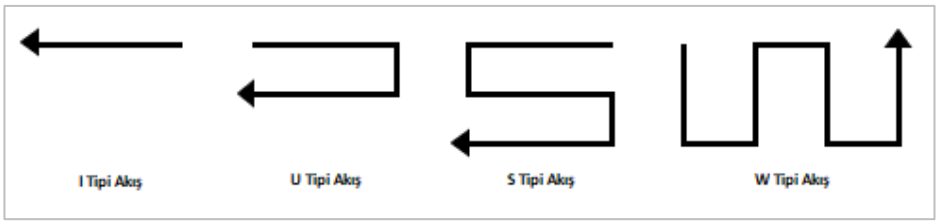
Her bir ürün ve bu ürünlerin üretim miktarları karşılaştırıldığında, ürünlerin hangi yerleşim tipi ile üretilmesi gerektiği ortaya çıkmaktadır. Yapılan çalışmanın sonucunda ise, genellikle bir yerleşim türünün diğerlerine göre daha etkili olduğu görülmektedir.

2.3.2. Tesis yerleşimi düzenine göre akış modelleri

Üretimin girdisi olan hammaddeler, tesise girdikten sonra çeşitli süreçlerden geçip, müşteri için bir değer olan ürüne dönüşerek tesisten çıkana kadar, tesis içerisinde farklı rotaları izlemektedir. Tesis içi yerleşim planlama adımlarında olduğu gibi; akış modelleri bölümler arası akışlar, bölüm içi akışlar ve iş istasyonu içerisindeki akışlar olmak üzere üç ana başlık altında incelenebilir.

2.3.2.1. Bölümler arası akışlar

Bölümler arasındaki akışlar, dört tip olarak incelenmektedir [1]. Bu akışlar Şekil 2.6.'da gösterilmektedir. Bir tesis içerisinde bölümler arası akışlar, tesislerin ve bölümlerin giriş çıkışları, tesisin ve bölümlerin şekilleri gibi faktörlere göre aşağıda gösterilmekte olan tiplerin kombinasyonları şeklinde oluşmaktadır.

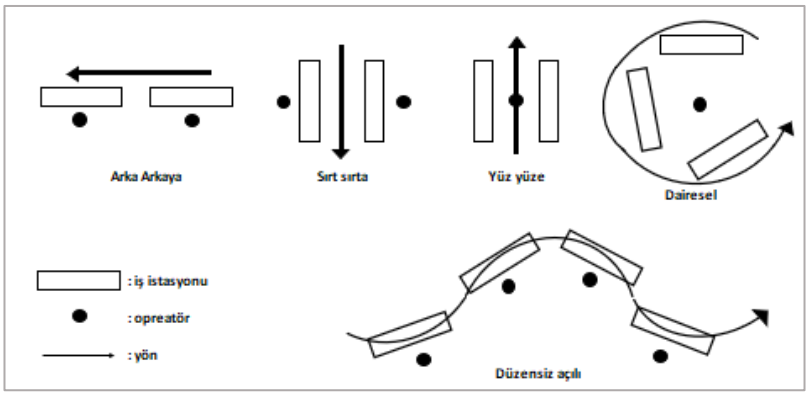


Şekil 2.6. *Genel akış modelleri (Tompkins vd, 1996)*

I tipi akışlar genellikle üretim adedi yüksek ve işlem sayısı az ürünlerin üretimi için kullanılmaktadır. Hammadde giriş ve ürün çıkış noktaları iki ayrı uçta yer almaktadır. U tipi akışlarda hammadde girişi ve ürün çıkış noktaları aynı tarafta olmakla birlikte, genellikle tam zamanında üretim yapılan işletmelerde tercih edilmektedir. En önemli özelliği işgücünün birden fazla makineye müdahale edebiliyor olmasıdır. Bu sayede hem ekipmanın hem de işgücünün verimliliği sağlanabilmektedir. S tipi ve W tipi akışlar ise yine üretim adedinin yüksek, işlem sayısının fazla ve aynı alanda olmasını gerektiren durumlarda tercih edilmektedir. Hammadde giriş ve ürün çıkış noktalarının aynı tarafta olması gereken durumlarda W tipi, farklı tarafta olması gereken durumlarda ise S tipi akış kullanılmaktadır.

2.3.2.2. Bölümler içerisindeki akışlar

Bölümler içerisindeki akışlar, iş istasyonları arasında gerçekleşmektedir. Bir iş istasyonundan diğerine aktarılan ürün, iş istasyonundan çıkıp koridora, koridordan da diğer iş istasyonuna geçmektedir. Bölüm içi akışlar ise Şekil 2.7.'de gösterildiği gibi sınıflandırılmıştır [1].



Şekil 2.7. *Ürün akış modelleri (Tompkins vd, 1996)*

2.3.2.3. İş istasyonu içindeki akışlar

İş istasyonu içindeki akışlar ise çalışan operatörün hareketine göre düzenlenmektedir. Burada yerleşim, insan antropometrisine ve ergonomik şartlara göre planlanmakta, operatörün rahat ve ritmik çalışması amaçlanmaktadır.

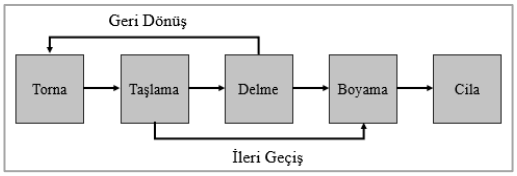
Bu sınıflandırmaların dışında daha farklı akış modelleri de bulunmaktadır. Bunlar üretim süreçlerine ve tesisin ihtiyacı olan özelliklerin ön plana çıkması ile tercih edilen akış tipleridir. Özellikle montaj hatlarında tercih edilen, ağaç biçimli, dal biçimli, kılçık şeklinde ve çok katlı akış tipleri sayılabilir. Bunun yanı sıra, tesis üzerinde, bir iş istasyonunun merkeze konulması ve böylece merkezden dışa doğru spiral şeklinde diğer iş istasyonlarının ilerlemesiyle, birbirleriyle yakından ilişkili bölümlerin bitişikliğini arttırmak ve malzeme taşıma maliyetlerinde bir düşüş sağlamanın amaçlandığı spiral tipi akış da örnek gösterilmektedir [6].

2.3.3. Akış hareketinin yönüne göre tesis yerleşimi problemleri

Bir akış hattında, akış hareketinin yönüne göre TYP'ler; önceki iş istasyonlarından birine geri dönüş (backtracking) ya da sonraki iş istasyonlarından birine ileri geçiş (bypassing) olmak üzere ikiye ayrılmaktadır.

Bir üretim hattında art arda dizili iş istasyonlarında ilerleyen ürünün, önceki iş istasyonlarından birine geri dönerek işlem görmesi geri dönüşlü TYP'ler başlığı altında incelenmektedir. Bu problemlerin amacı geri dönüş sayısını en küçükleyerek, taşıma mesafesini ve maliyetini minimuma indirmektir.

Eğer üretim hattında, üretilen ürünler üretim sürecine göre dizilmiş olan iş istasyonlarından birine ya da birkaçına uğramayıp, sonraki iş istasyonlarına geçiyor ise, bu tür problemler de ileri geçişli TYP'ler başlığı altında incelenmektedir. Burada amaç, ürünün üretim sürecine göre iş istasyonlarını birbirine en yakın şekilde konumlandırarak, taşımalar sırasında kat edilen mesafenin en küçüklenmesidir. Her iki yönlü akışın örneği Şekil 2.8.'de gösterilmektedir.

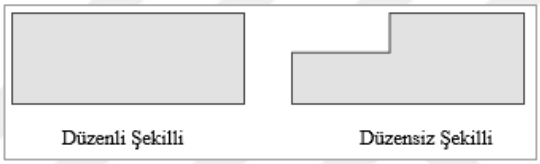


Şekil 2.8. *Geri dönüş ve ileri geçiş hareketi*

2.3.4. Tesis şekline göre tesis yerleşimi problemleri

Tesis şekline göre TYP'ler düzenli ve düzensiz şekilli olmak üzere iki farklı kategoride incelenmektedir. Düzenli şekilli tesisler genellikle dikdörtgen şeklinde olmakla birlikte, düzensiz şekiller genellikle çokgen şeklindedir. Tesislerin şekilleri, kenar uzunlukları ve varsa kenarlar arasındaki açılar kullanılarak tanımlanmaktadır. Buna ek olarak literatürde, tesisin şeklinin, tesis alanı ya da en boy oranı ile de tanımlanacağı belirtilmekte ve tesisin en boy oranının aşağıdaki formül ile ifade edilmektedir [7]. Şekil 2.9.'da düzenli ve düzensiz şekilli tesislere örnek gösterilmektedir.

$$a_i = \frac{i\ tesisi\ yüksekliği}{i\ tesisi\ genişliği} = \frac{h_i}{w_i} \tag{2.1}$$



Şekil 2.9. *Düzenli ve düzensiz şekilli tesisler*

2.3.5. Planlama dönemine göre tesis yerleşimi problemleri

"Planlama döneminin uzunluğuna göre TYP'ler Statik Tesis Yerleşim Problemi (STYP) ve DTYP olmak üzere iki gruba ayrılmaktadır [8]." Planlama dönemi boyunca tesisler arası akışlar sabit ise problem STYP olarak incelenmektedir. Eğer planlama dönemi boyunca tesisler arası akışlarda değişkenlik söz konusu ise, problem DTYP olarak ele alınmaktadır.

Gün geçtikçe artan rekabet, işletmeleri çevik olmaya zorlamaktadır. İşletmeler ürün çeşitliliğinin artması, üretim teknolojilerinin hızla gelişmesi gibi pek çok rekabet unsuruna hızlı adapte olmak zorundadır. Ancak bu şekilde pazar payının artırılması ya da korunması mümkündür. Üründe, üretim sisteminde, yönetim modellerinde, ürerim miktarları ya da organizasyonda yaşanan değişikliklere hızlı ayak uydurma gerekliliği, tesislerin yeniden düzenlenmesi ihtiyacını ortaya çıkartmaktadır. Kolayca değişiklik yapılabilen esnek tesis yerleşimlerine olan ihtiyaç giderek artmaktadır.

Tüm sistemler dinamik olmakla birlikte, bazı sistemlerde değişim hızı çok düşük olup, tüm planlama dönemleri göz önüne alındığında kısa dönemlerdeki değişiklikler önemsenmeyecek kadar küçük olabilir. Bu durumda tesis içerisinde herhangi bir değişikliğe ihtiyaç duyulmamaktadır.

Üretim gerekliliklerinde yaşanan değişiklikler sonucunda yeniden düzenleme yapmak zor ve maliyetli ise, gürbüz yerleşim planı önerilmektedir. Gürbüz yerleşim planında, yeniden düzenleme ve üretim kesintisi maliyetlerinin çok yüksek olduğu ve bu nedenle tek bir tesis yerleşim planı yapılarak tüm malzeme taşıma maliyetlerinin her dönem için en aza indirilmesinin amaçlandığını belirtilmektedir. Gürbüz yerleşim planının, değişikliğin yaşandığı pek çok dönem için tek bir düzen önermekte olduğundan bahsedilmektedir [9].

2.4. Tesis Yerleşimi Probleminin Meta-sezgisel Çözüm Yöntemleri

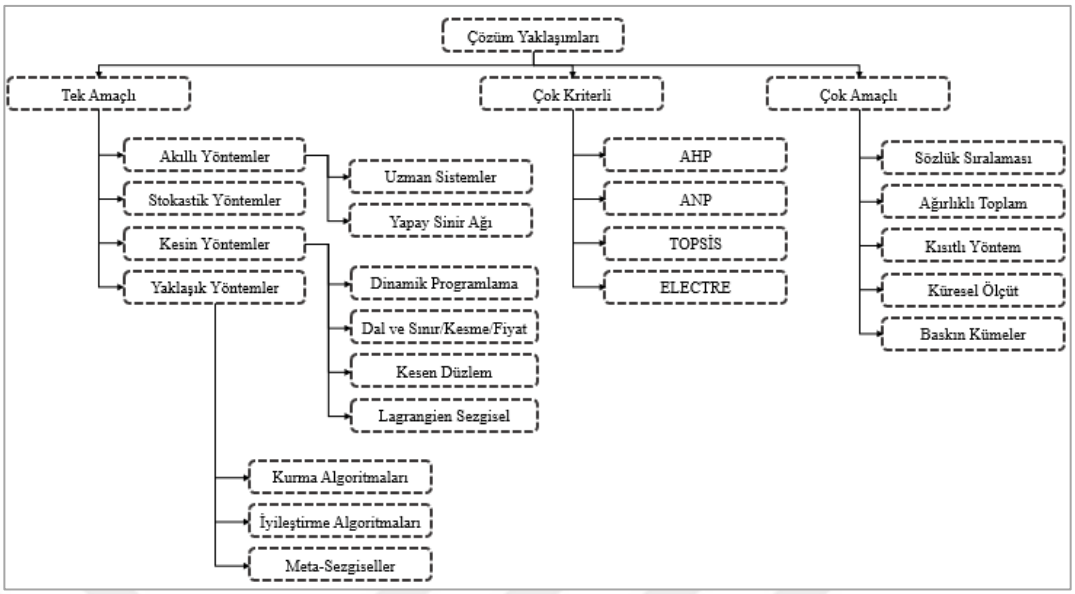
TYP, bir tesisin faaliyetlerini sürdürebilmesi için gerekli olan yeterli sayıdaki makine ve ekipmanın, ilgili bölümlere belirli bir alan içerisinde en iyi şekilde yerleştirilmesi ile ilgilenmektedir. Bölümlerin veya makinelerin hangilerinin nereye konumlandırılması gerektiğini araştırır. Amacı malzeme taşıma maliyetlerinin en küçüklenmesidir. İş istasyonlarının birbirine olan mesafelerinden ve birbirleri arasında gerçekleştirdikleri akış miktarlarından faydalanmaktadırlar. Bu tür problemlerin çözümlerinin çok kolay olmadığı ve çözüm sürelerinin uzun olduğu bilinmektedir.

Problemleri en iyi sonuca en kısa sürede ulaşacak şekilde en kolay yöntemle çözmek için pek çok çözüm yöntemi geliştirilmiştir. Kısacası TYP'ler en iyi çözümü aradığı için en iyileme problemi, hem de tesis yerleşiminin ne şekilde olması gerektiğini aradığı için tasarım problemidir.

Çözüm yöntemleri ile ilgili olarak literatüre bakıldığında, tesis planlaması konusunda uzman kişilerin geliştirdikleri birtakım prosedürlerin takip edilerek çözüme ulaşıldığından bahsedilmektedir. Ayrıca, amaç fonksiyonun yerini tutan bir fonksiyonun kısıt ve amaçlarını basitleştiren algoritmik çözümler olduğu ve literatürün büyük bir bölümünde bu çözüm yöntemlerine yer verildiği konusunda bilgi vermektedir [10].

Sözlükte "optimizasyon" sözcüğü "bir şeyi (tasarım, sistem veya karar gibi), mümkün olduğu kadar mükemmel, işlevsel, etkili hale getirme eylemi, süreci veya metodolojisi" olarak açıklanmaktadır [2].

Nasab vd. (2017), TYP'leri yapmış oldukları çalışmada Şekil 2.10.'da görüldüğü gibi sınıflandırmıştır [8].



Şekil 2.10. *TYP'lerin çözüm yöntemlerine göre sınıflandırılması*

Bu sınıflandırmanın yanı sıra çözüm yaklaşımları daha farklı şekilde de ele alınabilmektedir. Algoritmalar;

- Kullandıkları verilere göre nitel veri kullananlar, nicel veri kullananlar ve melez olarak,
- Benimsenen amaca göre toplam maliyetin en küçüklenmesi ve fayda / yakınlık puanının en büyüklenmesi
- Yerleşim planının gösterimine göre kesikli ve sürekli
- Kullanım amacına göre kurma ve geliştirme esaslı olarak incelenmektedir.

Meta-sezgisel algoritmalar, bir problemin çözüm alanının detaylı aramasının yapılarak, en iyi çözümün bulunmasını amaçlayan algoritmalardır. Yerel arama yapan meta-sezgiseller, popülasyon araması yapan meta-sezgiseller ve öğrenme mekanizmaları olmak üzere üç sınıfta incelenmektedir. Meta-sezgisel algoritmalardan bazıları şunlardır:

- Karınca Kolonisi Algoritması (KKA)
- Tavlama Benzetimi (TB) sezgiseli
- Tabu Arama (TA) algoritması
- GA

GA'lar, biyolojik evrimden ilham alarak oluşturulmuş arama ve optimizasyon algoritmalarıdır. Doğada en iyi olanın hayatta kalma şansının daha yüksek olması örnek alınarak, GA'larda da en iyi çözüme sahip bireylerin bir sonraki nesle aktarılması olasılığının daha yüksek olmasına dayanmaktadır. İlk olarak Holland tarafından incelenen

GA'lar, popülasyon içindeki bireylerin rekabet etmelerini ve rekabet sonucu elenmelerini sağlayarak evrimsel süreci simüle etmektedir [11].

GA'ların en önemli avantajlarından biri popülasyonu oluşturan bireylerin sayısı kadar çözüm üzerinde çalışmasıdır. Her bir çözüm problemin yapısına uygun bir şekilde dizilere kodlanarak bireyleri oluşturmakta, bireyler de popülasyonları oluşturmaktadır. Çaprazlama ve mutasyon adı verilen genetik operatörler ile bireyler doğada olduğu gibi değişikliklere uğrayarak yeni popülasyonları oluşturmaktadır. Bu şekilde nesiller simüle edilmektedir. Popülasyon içerisindeki bireylerin uyumları ise problemlerin amaç fonksiyonlarına göre değerlendirilmektedir. Bu döngü istenen sayıda nesil oluşana kadar ya da daha iyi bir çözüm elde edilemeyinceye kadar tekrarlanmaktadır. Bu zamana kadar elde edilen en iyi çözüm problemin çözümü olarak kabul edilmektedir.

3. DİNAMİK TESİS YERLEŞİMİ PROBLEMİ VE MEVCUT ÇALIŞMALAR

Üretim gerekliliklerinde dönemler bazında yaşanan değişiklikler için ihtiyaç duyulan yeniden düzenleme maliyetinin düşük ve düzenleme süresince yaşanacak üretim kesintisi maliyetinin de katlanılabilir boyutta olması durumunda tesis kolayca yeniden düzenlenebilmektedir. Bu durumda da DTYP tercih edilmektedir. "Taşıma maliyetlerindeki artış katlanılamaz hale geldiğinde yerleşim düzenlemesini de değiştirmek gerekir. Bu da dinamik yerleşim probleminin kaynağını oluşturur. [12]."

3.1. Dinamik Tesis Yerleşimi Problemi

DTYP'ler, üretim gerekliliklerinde sürekli değişiklikler yaşanması nedeni ile ihtiyaç duyulan bir TYP türüdür. Ürünün tasarımında, üretim süreçlerinde ve teknolojilerinde, talep miktarlarında veya yönetim sisteminde yapılan değişiklikler sonucunda işletmeler hızlıca bu değişikliklere ayak uydurmak durumunda kalmaktadır. Değişikliklere ayak uydurulamaması durumunda;

- Müşterinin talebine zamanında cevap verememe,
- İstenen kalite ve standartta ürün üretememe,
- Ürünün birim maliyetinin artması gibi pek çok sorun yaşanabilmektedir.

Bunlara ek olarak yaşanan en büyük problem ise, müşterilerin, bu ihtiyaçlara daha hızlı cevap veren rakip işletmelerden ürün veya hizmet almaya karar vermesi, yani pazar payının giderek küçülmesidir.

Bu hızlı değişikliklere zamanında cevap verebilmek için DTYP kullanılmaktadır. DTYP karmaşık kombinatoryel optimizasyon problemidir ve çözümü oldukça zordur. Problemin çözümünde, en iyiye yakın sonuçlar veren meta-sezgisel algoritmalardan faydalanılmaktadır. DTYP, talepteki ve pazardaki değişimlere göre çok dönemli tesis planlamasından oluşmaktadır [13].

Üretimi gerçekleştirilen ürünler bir iş istasyonundan diğerine taşınırken teker teker ya da belirli sayıdaki gruplarla taşınmaktadır. Taşımalar her bir ürün için tek tek gerçekleşiyor ise, taşıma sayısı ilgili dönemdeki üretim miktarına eşittir. Ürünlerin belirli sayıdaki gruplar halinde taşınması durumunda ilgili ürünün rotasında o döneme ait taşıma sayısını bulmak için üretim miktarının, bir defada taşınan grup içindeki ürün sayısına bölünerek bulunacağı belirtilmektedir. İster tek tek ister grup halinde olsun, bir birim yükün bir metre taşınması için harcanan paraya ise, bir birim yükün taşıma maliyeti denilmektedir [14].

Taşıma maliyetleri incelenirken, genellikle sadece iş istasyonları arasındaki taşımalar göz önünde bulundurulmaktadır. Taşıma için kullanılan araçların maliyetleri, yükleme, boşaltma maliyetleri ya da yeniden işlemeler için gerçekleştirilen taşımalardan kaynaklanan maliyetler genellikle ihmal edilmektedir.

"Yapılan çalışmalar sonucunda malzeme taşıma maliyetinin, işletme maliyetinin yüzde 20 ile 50'sini, ürünün toplam üretim maliyetinin ise yüzde 15 ile 70'ini oluşturduğunu belirtilmektedir [1]." Taşıma maliyeti oranları bu kadar yüksek iken, bunların göz ardı edilmesi mümkün değildir. Bu nedenle taşıma maliyetlerini düşürebilmek için etki edilebilir faktör olan tesis yerleşim planlaması üzerinde çalışılabilir.

Tesis yerleşim planının değişmesine neden olabilecek pek çok faktör bulunmaktadır. Bunların içinde en sık karşılaşılanlarına talep miktarındaki değişiklik, üründe ya da üretim sürecindeki değişiklik nedeni ile gerçekleşen ürünün izlediği rotanın değişmesi örnek gösterilebilir. Üretim sürecinde bir iş istasyonunun ortadan kaldırılması ya da yeni bir iş istasyonu eklenmesi veya ürüne yeni bir parça eklenmesi gereklilikleri ile taşıma rotalarında mesafeler artabilmektedir. Talepte meydana gelen değişiklikler ise, değişiklik öncesi durumdaki talebe göre düzenlenmiş tesis yerleşim planını, yeni duruma göre kabul edilemez hale getirebilmektedir. Talepte yaşanan bir değişiklik nedeni ile en çok üretilen ürün göz önüne alınarak oluşturulmuş tesis yerleşim planının nasıl etkilendiği bir örnek ile açıklanacaktır. Şekil 3.1.'de örnek ile ilgili bilgiler bulunmaktadır.

İlk Yerleşim Planı

1	4	3	7
6	5	2	8

Ürün Tipi	Talep Miktarı	Parti Büyüklüğü	Taşıma Sayısı	Rota	Maliyet
A	1000	50	20	1-4-5-2-8	₺80
B	200	20	10	5-3-7-8	₺40
C	600	30	20	4-5-3-8	₺100

Son Yerleşim Planı

2	6	3	7
1	4	5	8

Ürün Tipi	Talep Miktarı	Parti Büyüklüğü	Taşıma Sayısı	Rota	Maliyet
E	450	15	30	1-2-6-3	₺90
B	1200	20	60	5-3-7-8	₺180
C	150	30	5	4-5-3-8	₺20

Şekil 3.1. *Üretim gereklilikleri değişiminin yerleşime etkisi*

Şekil 3.1'de gösterilmekte olan ilk yerleşim planı işletmenin bir dönemini ifade etmektedir. Bu ilk dönemde A, B ve C tipi olmak üzere 3 çeşit ürün üretilmektedir. Ürün talep miktarları, taşıma parti büyüklükleri, taşıma sayıları ve iş istasyonları arasında izledikleri rotalar gösterilmektedir.

Her bir ürün için ayrı ayrı taşıma maliyetleri hesaplanmış olup, bu maliyetler şeklin son sütununda bulunmaktadır. Maliyet hesaplaması yapılırken iş istasyonları arası uzaklıklar dik doğrusal olarak hesaplanmıştır. Her bir iş istasyonunun eşit alanlı olduğu varsayımı ile, iş istasyonlarının bir kenar uzunluğu bir birim olarak tanımlanmıştır. Bir partinin bir birim taşınma maliyeti ise değişiklik öncesi ve sonrası durum karşılaştırılacağından işlem kolaylığı olması için 1 TL olarak kabul edilmiştir. Şekil 3.1'de verilen değerlerin tamamı bir dönemi temsil etmektedir.

İlk durumdaki A ürünü için, ilk iş istasyonundan son iş istasyonuna kadar ürünün izlediği rotada kat ettiği mesafe 1'den 4'e 1 birim, 4'ten 5'e 1 birim, 5'ten 2'ye 1 birim ve 2'den 8'e 1 birim, toplamda 4 birim olmak üzere hesaplanır (burada eğer 1'den 3'e ya da 1'den 5'e gibi olursa her biri 2 birim olarak hesaplanacaktır).

Ürünün izlemiş olduğu rotanın uzunluğu, toplam taşıma adedi ve bir birim uzunluk için taşıma maliyeti birbirleri ile çarpılarak bir dönem için taşıma maliyeti 4(m)*20(ad)*1(br)=80 TL olarak bulunmaktadır. İlk durum için diğer ürünlerin de taşıma maliyetleri hesaplandığında, işletmenin bir dönemde toplam taşıma maliyeti 220 TL olarak bulunmaktadır.

Şekil 3.1.'de ifade edilmekte olan ikinci durumda ise işletmenin bir sonraki dönem için tercih ettiği son yerleşim planı görülmektedir. Bu dönemde, bir önceki dönemde üretimi gerçekleştirilen A ürünü artık üretilmemekte, yerine E ürünü üretilmektedir. E ürününün talep miktarı, parti büyüklüğü, rotası ve dolayısı ile taşıma sayıları A'dan farklıdır. Buna ek olarak üretimine devam edilen B ve C ürünleri için talep miktarları değişse de parti büyüklükleri ve rotaları aynı kalmıştır. Talep miktarındaki değişim, ürünlerin taşıma sayılarını önemli ölçüde değiştirmiştir. İşletmede en çok üretilen ve en yüksek sayıda taşıma adedine sahip ürün B ürünüdür.

İkinci durumda son tesis yerleşimine göre, ikinci döneme ait işletmenin toplam taşıma maliyeti 290 TL olarak hesaplanmıştır. Ancak tesis yerleşim planında değişiklik yapılmayıp, üretim faaliyetleri ilk durumdaki gibi devam etseydi, işletmenin toplam taşıma maliyeti 505 TL olacaktı. Yani tesisin yerleşim planında yapılan değişiklik ile taşıma maliyeti %42,6 oranında azalmaktadır. Ancak, tesis yerleşiminin yeniden planlanmasını sadece taşıma maliyeti yönünden incelemek yetersiz kalmaktadır.

Tesis yerleşim planlamasında değişiklik yapılması genellikle oldukça maliyetli bir durumdur. Üretim faaliyetlerine devam eden bir tesiste yerleşimin yeniden düzenlenmesi;

- Üretimin geçici süre ile durması kaynaklı yaşanacak üretim kayıpları kaynaklı maliyete,
- İş istasyonlarının yer değiştirmesi sürecinde kullanılacak ekipman ve iş gücü maliyetlerine,
- Yerleşim planı değiştirildikten sonra işgücünün yeniliğe alışması sürecindeki kayıpların maliyetlerine katlanmayı gerektirmektedir.

Bu noktada tesis yerleşim planının DTYP olarak düzenlenip düzenleyemeyeceği konusu detaylı bir analiz ve çalışma gerektirmektedir. Ancak yapılan analizler sonucunda tesis yerleşim planında değişiklik yapıldıktan sonra ortaya çıkacak olan yer değişimi maliyeti miktarı ile elde edilecek olan kazanç miktarı, iş istasyonlarının taşınmasının toplam maliyetinden büyük ise gerçekleştirilir. Diğer durumda değişiklik yapıp taşıma maliyetlerinden fayda sağlamak, işletmenin toplam maliyetinin artmasına neden olacağından mantıksızdır.

Şekil 3.1.'deki örnek incelendiğinde, 3, 7 ve 8 numaralı iş istasyonlarının bir önceki yerleşime göre konumunu koruduğu ancak, 1, 2, 4 ve 5 numaralı iş istasyonlarının eski konumlarından yeni konumlara taşındığı görülmektedir.

Bir iş istasyonunun yer değişimi maliyetinin 50 TL olduğunu kabul ettiğimizde, işletmedeki yeni tesis yerleşim planına göre toplam yer değiştirme maliyeti dört iş istasyonunun yer değiştirdiğini düşünürsek 200 TL olarak bulunmaktadır. Bir önceki duruma göre tesis yerleşim planının değiştirilmesi sonucunda toplamda 505-290=215 TL kazanç sağlanmaktaydı. Bununla birlikte 200 TL yer değiştirme maliyeti de dikkate alındığında, tesis yerleşim planının değiştirilmesi ile sağlanan kazancın, yer değiştirme maliyetinden büyük olduğu görülmektedir. Yapılan değişiklik sayesinde işletme toplam 15 TL kazanç sağlamaktadır.

Ancak, bir iş istasyonunun yer değiştirme maliyetinin 60 TL olduğunu kabul edersek, bu defa toplam yer değiştirme maliyeti 240 TL olacak ve bu da tesis yerleşim planının değiştirilmesi ile elde edilen kazançtan küçük olacaktır. Sonuçta işletme, 25 TL daha fazla maliyete katlanmış olacaktır.

Örnekte kullanılan sayılar hesaplama kolaylığı açısından her ne kadar önemsiz görünecek kadar küçük olsa dahi, gerçek hayat problemlerinde çok büyük maliyetlere karşılık gelmektedir. Bu nedenle DTYP ile tesis yerleşimi yapmaya karar vermek zor ve fazlaca analiz gerektirmektedir. Üretim gerekliliklerindeki değişikliklerin ne şekilde gerçekleştiği takip edilmeli, değişikliklerin katlanılır şekilde yavaş bir şekilde mi, yoksa

hızlı etki yaratacak şekilde aniden mi ortaya çıktığı belirlenmelidir. Ani değişikliklerin meydana geldiği zamanlar ve durumlar takip edilmeli, eğer değişiklik yapılacaksa dönemlerin ne zaman başlayıp ne zaman son bulacağı öngörülmelidir.

"TYP'ler, problemin özelliklerine göre farklı şekillerde modellenebilmektedir. Bu yöntemlerin içerisinde en çok tercih edilen ise Kareli Atama Problemidir (KAP). Kareli küme kapsama, doğrusal tamsayılı programlama, karışık tamsayılı programlama ve çizge kuramı modelleri olarak faklı şekillerde de modellenmektedir. [15]."

KAP, tüm bölümlerin birbiri ile eşit alanlı olduğu durumlarda kullanılmaktadır. İlk defa Koopmans ve Beckman tarafından 1957 yılında modellenmiş olup, günümüzde araştırmacıların ilgi odağı haline gelmiştir. Kısıtlar, değişkenlerin doğrusal fonksiyonları olsalar dahi, amaç fonksiyonu değişkenlerin ikinci dereceden bir fonksiyonu olduğundan, problemin isminin KAP'tir. [14]. Her zaman KAP'ın eşit alanlı tesisler için kullanılması zorunlu değildir. Literatürde eşit olmayan alanlı tesislerin yerleşim problemlerinde de KAP kullanıldığı görülmektedir. Eşit olmayan alanlı tesislerin uygun yerlerden bölünerek daha küçük eşit alanlar elde edilmesi ve bir bölümü oluşturan parçaların birbirinden ayrılması engellenerek, bölümün dağılmasının önüne geçilebilmektedir. DTYP'nin çözülebilmesi için, modele zaman boyutu eklenmektedir. Şu şekilde modellenmektedir:

n: iş istasyonlarının / konumların sayısı

t: dönem sayısı

f_{tik}*:* t döneminde i iş istasyonundan k iş istasyonuna malzeme akış sayısı

d_{jl}*:* j konumunun l konumuna olan mesafesi

A_{tijl}*:* t döneminde i iş istasyonunun j konumundan l konumuna atanmasının sabit maliyeti

C_{tijkl}*:* t döneminde i iş istasyonu ile k iş istasyonu arasındaki taşımaların maliyeti

x_{ij}*:* i iş istasyonu j konumuna atanmış ise 1, diğer durumda 0

$C_{tijkl} = f_{tik} * d_{jl}$

$Y_{tijl} = x_{(t-1)ij}\, x_{til}$ olmak üzere:

$$\sum_{j=1}^{n} x_{tij} = 1, \quad i = 1, 2, \dots, n \quad t = 1, 2, .., t \tag{3.1}$$

$$\sum_{i=1}^{n} x_{tij} = 1, \quad j = 1, 2, \dots, n \quad t = 1, 2, .., t \tag{3.2}$$

$$x_{tij} \in \{0,1\}, \quad \forall\, i, j, s \tag{3.3}$$

Kısıtları altında;

$$Enk \quad \sum_{t=2}^{t}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n} A_{tijl}y_{tilj}$$
$$+\sum_{t=1}^{t}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} C_{tijkl}x_{tij}x_{tkl} \qquad (3.4)$$

Modelde bulunan kısıtlar incelendiğinde, (3.1) ile gösterilen denklem ilgili t dönemi için, her bir iş istasyonunun ancak ve sadece bir konuma atanabileceğini belirtmektedir. (3.2) numaralı denklem ise, her bir konuma sadece bir iş istasyonunun gelebileceğini göstermektedir. (3.3) numaralı denklemde, iş istasyonunun atanıp atanmaması durumuna göre karar değişkeninin hangi değeri alacağı ifade edilmektedir. (3.4) ile gösterilen son denklem ise problemin amaç fonksiyonu olup, tüm dönemleri kapsayacak şekilde hem iş istasyonları arası taşıma maliyetlerini hem de dönemler arasında iş istasyonlarının yer değiştirmesi durumunda ortaya çıkacak iş istasyonunun taşınması maliyetlerinin toplamını en küçüklemeyi amaçlamaktadır.

En çok kullanılan modelleme yöntemi KAP olduğu için bu bölümde sadece bu modelin açılımı gösterilmiştir. Diğer yöntemler de DTYP için kullanılabilmekle birlikte, tek bir dönem için modelleme örneklerini Kursiak ve Heragu, 1987 yılında yapmış oldukları "TYP" başlıklı makalelerinde bir araya toplamıştır. DTYP problemlerinde bu modellere zaman boyutu eklenmesi gerekmektedir.

Ayrıca, diğer modellere zaman boyutu eklemenin yanı sıra, probleme ait iş istasyonlarının eşit alanlı olup olmamasına göre modellere farklı kısıtlar da eklenmektedir. Literatürde, KAP'lere göre bu tür kısıtları olan problemlere daha az rastlanmaktadır.

Modelleme ile çözümlerin yanında, modellemenin tüm kısıtları problemin amacını karşılamadığında, probleme özgü çeşitli algoritmalar geliştirilerek de pek çok özel problemin çözümü sağlanabilmektedir.

3.2. Dinamik Tesis Yerleşimi Problemiyle İlgili Mevcut Çalışmaların İncelenmesi

Talep dinamikliğinin bulunduğu bir ortamda, eşit olmayan alanlı iş istasyonlarının DTYP'sinin GA ile çözüldüğü bu yüksek lisans tezine ışık tutması için bir literatür taraması yapılmıştır. DTYP ile ilgili daha önce yapılmış olan çalışmalar, devam eden paragraflarda anlatılmaktadır.

Balakrishnan, Jacobs ve Venkataramanan (1992), TYP'yi değişen talep varsayımı ve kısıtlı yerleşim düzenleme kaynakları altında incelemişlerdir. Problem formüle edilmiş, çözüm için yeni bir algoritma önerilip, önceden çözüm için kullanılan Dinamik

Programlama (DP) algoritmasının bir uzantısı ile kıyaslanmıştır. Çalışmanın sonucunda, DTYP için kullanılabilecek kaynaklar kısıtlı olduğunda, kısıtlı En Kısa Yol Problemlerinin, DP'den daha iyi sonuç verdiği görülmüştür. Problem boyutu küçük ve kısıtların çok sıkı olması durumunda ise DP algoritmasının EKYP algoritmasından daha iyi performans gösterdiği görülmüştür. Bunların kombinasyonları DP algoritmasını destekliyor olsa da pratikte bu koşulların gerçekleşmesi olası olmadığından, EKYP'nin kısıtlı DTYP için daha iyi bir alternatif olduğu düşünülmektedir [16].

Conway ve Venkataramanan (1994), GA'ların zaman içinde TYP'ye uygunluğunu incelemiş, çok dönemli TYP için bir GA sunmuş ve iki test problemi için sonuçlarını incelemişlerdir. Çok dönemli problemler, GA'ların temel teorem koşullarını yüksek oranda karşılayabilmektedir. Araştırmada, dönem bütçesi kısıtları olan kısıtlı DTYP için sonuçlar elde edilmiştir. Kısıtlı DTYP için gerçekleştirilen önceki çalışmalar, boyut ya da ek kısıtlamaları yerine getirememesi ile kısıtlı iken; bu çalışma, mevcut bütçe değişime izin vermiyorsa, kısa vadeli borçlanma planını dahil etme gibi pek çok kısıtlamayı genişletme yaklaşımını sunmaktadır. Bunun yanı sıra yöntem, esnekliği nedeni ile karar destek sistemi ile etkileşime geçmek için kolayca genişletilebilmektedir [17].

Urban'a (1998) göre, DTYP için mevcut en uygun çözüm prosedürleri, dinamik bir program çerçevesinde tekrarlı KAP'lerin çözümünü gerektirmektedir. Çalışmada ele alınan problemin hesaplama gereksinimleri, problemin özel durumları için etkin algoritmaların ve genel durumu için ise güçlü sınırların geliştirilmesini gerektirmektedir. DTYP'ye en uygun çözümü bulmak için son derece düşük çözüm sürelerinde, sabit yeniden düzenleme maliyetleri ile eksik DP uygulanmış, daha büyük problemler için sezgiseller geliştirilmiştir. DTYP'ye en uygun çözümün, yeniden düzenleme maliyetlerinin sabit olduğu durumlarda daha kolay tespit edildiği gözlenmiştir. Tamamlanmış bir DP kullanan çözüm geliştirilmiştir. Bunun yanında, karşılaştırılabilir maliyet verilerinin elde edilmesinin zor olduğu durumlar için ayrı bir etkin sınır üretilmiştir. Gerçekçi bir şekilde çözülebilecek problemin büyüklüğü KAP'in çözümü ile sınırlı olduğundan, bu problem için sezgisel çözüm metodolojileri sunulmuştur [18].

Balakrishnan vd. (2003), GA'lara dayalı meta-sezgisel taramaların büyük DTYP'leri çözebileceğini göstermiştir. Çalışmada hibrit bir GA oluşturarak GA'ların kullanımını genişletmeye ve geliştirmeye odaklanılmıştır. Önerilen algoritmayı, mevcut GA'lar ve önceden simüle edilmiş bir TB algoritması ile karşılaştırmak için bir hesaplama çalışması yapılmıştır. Çalışmanın sonucunda önerilen algoritmanın etkili olduğu, daha

büyük problemler için diğer GA'lardan daha iyi performans gösterdiği görülmüştür. GA ile DP'nin birlikte kullanıldığı bir hibrid yaklaşımın yalnızca GA kullanmaktan daha iyi sonuçlar verdiği anlaşılmıştır [19].

Baysakoğlu, Dereli ve Sabuncu (2006), DTYP'yi bütçe kısıtlarını göz önüne alarak çözmek için KKA kullanmıştır. DYP problemlerinde KKA'nın bütçe kısıtlamaları ile ne şekilde uygulanabileceğini göstermek için yapılan ilk çalışma olması açısından önemlidir. KKA, DTYP gibi karmaşık problemleri çözmek için gelecek vaat etmektedir. Bu çalışmada kısıtlı ve kısıtsız DTYP'ye uygulanmıştır ve çözümü için basit ama etkili bir veri yapısına ve çözüm üretme mekanizmasına sahip bir KKA önerilmiştir. Algoritma, sonuçların bilindiği mevcut literatür test problemlerine uygulanmıştır. Bu testlerde en iyi çözümler değil, rekabetçi çözümler elde edilmiştir. Balakrishnan ve Cheng'in (2000) GA'sı NLGA, Conway ve Venkataramanan'ın (1994) GA'sı CONGA, Erel vd.'nin (2003) DP'si ve benzetilmiş TB meta-sezgiseli ile karşılaştırılmıştır. Karşılaştırmalar, önerilen KKA'nın NLGA ve CONGA'dan daha iyi performans gösterdiğini ancak daha büyük boyuttaki problemler için meta-sezgisel özelliklerinden daha iyi performans göstermediğini göstermiştir. KKA, bütçe kısıtlamalı DTYP problemine başarıyla uygulanmıştır. Bütçe kısıtlamalı DTYP için yayınlanan sonuçların mevcut olmaması nedeniyle karşılaştırma yapılmamıştır [20].

Şahin (2008), DTYP'yi çözmek için bir TB meta-sezgiseli geliştirmiştir. Önerilen TB meta-sezgiselinin başarısını ölçmek için literatürdeki test problemlerinden yararlanılmıştır. Bu problem seti için en iyi çözümleri veren yöntemler ile TB meta-sezgiseli karşılaştırılmış, en iyi sonuçlardan 7 tanesi için daha iyi sonuçlar elde edilmiş ve TB algoritmasının DTYP'yi çözmede etkin olduğu gözlemlenmiştir. Önerilen algoritma TB algoritmasının en yalın hali olduğundan, büyük boyutlu problemler (30 bölüm ve 10 dönem) için bir adet iyi çözüm bulunmuştur. Algoritmaya, yeniden ısıtma, algoritma içinde soğutma çizelgesinin değiştirilmesi gibi ek özellikler kazandırılırsa, büyük boyutlu DTYP için de daha iyi sonuçlar elde edilebileceği ön görülmüştür [21].

Şahin, Ertoğral ve Türkbey (2010), DTYP için bütçe kısıtlı bir TB meta-sezgiseli önermiş ve bu meta-sezgiselin etkinliğini sayısal deneyler ile göstermiştir. Bütçe kısıtlı DTYP önemli bir sorun olmasına rağmen, bu konuda yapılan çalışma sayısı oldukça azdır. Çalışmada problem ile ilgili olarak bir TB meta-sezgiseli önerilmiş ve sonuçları literatürden elde edilen sonuçlar ile karşılaştırılmıştır. Karşılaştırılan tüm problemlerde %1,27 ile %6,19 arasında değişen iyileştirilmiş çözümler elde edilmiştir. Gelecekteki

araştırmalara ışık tutmak için ise DTYP'de bütçe kısıtlamaları olan karma meta-sezgisel yöntemler geliştirilmesi önerilmiştir. Bunun yanında ürünlere olan talep oranlarının dönemsel değişkenlik gösterebilir olması nedeni ile stokastik akış hızına sahip DTYP çalışılması önerilmiştir [22].

Ripon, Glette ve Torresen (2011), çok hedefli DTYP'yi, yerleşimi bir Pareto-iyimserliğini çözüm seti olarak sunan belirsizlik altında çözmek için bir evrimsel yaklaşım araştırma çalışması gerçekleştirmişlerdir. Bu çalışmaya kadar, belirsizlik altında çok amaçlı DTYP'de Pareto-iyimserliği incelenmemiştir. Bu makalede önerilen yaklaşımın, çoklu hedefleri optimize etmedeki etkinliği, geriye doğru geçiş sezgisel yöntemi kullanılarak test edilmiştir. Sonuçlar yaklaşımın belirsizlik altında aynı anda birden fazla hedefi optimize etmede etkin bir evrimsel DTYP yaklaşımı olduğunu göstermektedir. Bu sonuçlarla, sabit ve deterministik değerler kullanan önceki çalışmalar karşılaştırıldığında bir farklılık olmadığını göstermektedir. Çalışma aynı zamanda bu yaklaşımı geriye geçişli sezgisel yöntem ile birleştirmenin yalnızca en yakın çözümlerin geliştirilmesine değil, aynı zamanda daha iyi yakınsama ve çeşitliliğe sahip son Pareto-iyimser düzenler oluşturulmasına yardımcı olduğunu da doğrulamaktadır [23].

Moslemipour ve Lee (2012), esnek üretim sistemlerinde dinamik makine yerleşimi problemine her dönem için optimal bir makine yerleşimi tasarlamak için KAP formülasyonu kullanarak yeni bir matematiksel model önermiştir. Ürün talepleri, rastgele dönemden döneme değişen olasılık yoğunluk fonksiyonu olarak bilinen normal dağılmış rastgele değişkenler olarak kabul edilmektedir. Modelde, karar vericinin tanımlanmış güven düzeyi göz önünde bulundurulmaktadır. Güven düzeyi, karar vericinin ürün taleplerindeki belirsizlik konusundaki tutumunu, problemin sonuçlarını önemli ölçüde etkileyecek şekilde temsil etmektedir. Modelin etkinliğini test etmek için farklı boyutlarda rastgele iki adet test problemi üretilmiştir. Problemi makul bir sürede çözmek için TB meta-sezgisel çözüm yaklaşımı kullanılmıştır. 10 parça, 12 makine ve 5 dönem ve 10 dönem olmak üzere 2 farklı dönemi olan test problemleri 3 farklı güven seviyesinde TA yaklaşımı ile çözülmüş ve sonuçları istatistiksel olarak analiz edilmiştir [24].

Mazinani, Abedzadeh ve Mohebali (2013), esnek bölme yapısına ilişkin çok amaçlı DTYP için karma tamsayılı DP modeli sunmaktadır. Modelde 3 ana amaç bulunmaktadır. Bunlar; malzeme taşıma ve tekrar yerleştirme maliyetlerini en aza indirme, bitişiklik oranını en üst seviyeye çıkarma ve şekil oranı farkını en aza indirmektir. Modeli çözmek için General Algebraic Modelling System (GAMS) yazılımı

ve paralel değişken komşuluk arama algoritması kullanılmıştır. Makalede çok amaçlı karmaşık tamsayılı DP'yi çözmek için bulanık programlama tekniği kullanılmıştır. Modelin etkinliğini ölçmek için 4 adet test problemi GAMS yazılımı ve paralel komşuluk arama algoritması ile çözülmüş, sonuçlarında ise, paralel komşuluk arama algoritmasının GAMS yazılımından daha verimli olduğu görülmüştür [25].

Chen (2013), araştırmaya odaklanan diğer sezgisel yaklaşımlardan faklı olarak, çözümü geliştirmek ve etkinlikleri meta-sezgisel bir çerçeve içinde depolamak için çözüm örneklerinin veri yapısını düzenleyen bir çalışma gerçekleştirmiştir. DTYP veri setindeki veri kodlama ve kod çözme şemalarının test edilmesi sonucu elde edilen sonuçlar, çözüm kalitesi ve hesaplama süresi açısından umut verici olmuştur. Makalede, kodlama ve kod çözme planlarının önerilmesinin yanı sıra, kodlama şeması ve bölüm çözümleme şeması da geliştirilmiştir. Diğer meta-sezgisel yöntemlerle karşılaştırıldığında 30 bölüm ve 10 dönemden oluşan bir DTYP için 3 kattan fazla ve hatta 8 kat daha hızlı olduğu görülmüştür [26].

Kaveh, Dalfard ve Amiri (2013), bir DTYP'yi bilgi akışının belirsiz olduğu durumda incelemiştir. Ürün talebi ve buna bağlı olarak malzeme akışı farklı üyelik fonksiyonlarına sahip bulanık sayılar olarak tanımlanmıştır. Problem bulanık programlamada modellenmiş ve beklenen değerin üç modeli; şans kısıtlı programlama, bağımlı şans programlama ve iki hibrit akıllı algoritma sunulmuştur. Algoritmaların verimliliği ise sayısal örneklerle gösterilmiştir. Pek çok çalışmada programlama sırasında belirli malzeme akışı bilgileri kullanılmaktadır. Ancak dinamik iş ortamlarında ürün çeşitlilikleri ve talepleri değişken olduğundan ürünlerin ve talebin hacmi bulanık sayılar olarak kabul edilmiştir. Problem bulanık programlama ve yukarıda bahsedilen 3 model ile modellenmiştir. Problemin karmaşıklığı ve çözüm imkansızlıkları nedeni ile GA, TB ve bulanık simülasyon içeren bir hibrit akıllı algoritma geliştirilmiştir. Sonuçlar, TB algoritmasının bu problemleri Hibrit GA'dan daha iyi çözdüğünü göstermiştir [27].

Ulutaş ve İşlier (2015), mevsimsel talep değişikliği bulunan bir endüstride tesis içerisindeki makinelerin yerleşimini incelemiştir. Gerçek veriler kullanılarak oluşturulan senaryolar ile DTYP çözmek için klonal seçim tabanlı bir algoritma önermişlerdir. Gerçek hayat problemleri büyük ve başa çıkması zor olduğundan, kesin yöntemler yerine daha kısa sürede uygun çözümlere ulaşılabilen sezgiseller kullanılmaktadır. Algoritmanın performansı, zaman aralıklarının çözüm kalitesi ve sonuçların uygulanabilirliği üzerindeki etkisi test edilmiş ve umut vadeden sonuçlar elde edilmiştir [14].

Wang vd. (2015), malzeme akışının farklı işlem dönemlerinde zaman içerisinde değiştiği çift sıra yerleşim problemi üzerine bir çalışma yapmıştır. Problem için karma tamsayılı programlama modeli kurulmuştur. Çözüm için ise Geliştirilmiş Tavlama Benzetimi (GTB) ile Matematiksel Programlamayı (MP) birleştiren bir metodoloji önerilmiştir. Birden fazla periyot için tesis sırasını ve tesisin tam yerini göz önünde bulunduran dinamik bir çift sıralı yerleşim problemi önerilmiştir. Yapılan deneyler sonucunda GTB-MP'nin küçük ölçekli problemler için en uygun çözümleri bulabildiği görülmüştür. Gerçek boyuttaki problemler için ise, tatmin edici çözümler bulabildiği görülmüştür [28].

Tayal vd. (2017), yerleşim tasarımının sürdürülebilir olmasını sağlamak için birtakım parametreler göz önünde bulundurmuş ve sürdürülebilir stokastik DTYP'yi modelleyerek çözmüştür. Çözüm için meta-sezgisel tekniklerden faydalanılmıştır. Oluşturulan yerleşim havuzu daha sonra verimli olan yerleşimleri belirlemek için veri zarflama analizi ile analiz edilmiştir. Bunun için sipariş tercihi tekniği, yorumlayıcı sıralama süreci ve analitik hiyerarşi süreci, Borda-Kendall ve tamsayılı doğrusal programlama tabanlı sıralama toplama teknikleri gibi çok amaçlı karar verme teknikleri uygulanmıştır. Metodolojiyi doğrulamak için 12 bölüm ve 5 dönemden oluşan Gauss talebine sahip bir veri kümesi kullanılmıştır. Sürdürülebilir yerleşim için önerilen metodoloji meta-sezgisel teknikleri birleştirmektedir. Açıklanmakta olan etkili sistematik karar verme, kötü bir yerleşim tasarımı seçme riskini azaltmaktadır. Önerilen metodoloji, çevre ve sosyal sonuçları, yerleşim tasarlandıktan sonra düzeltici faaliyet olarak değil, en başından tasarlama aşamasında kapsayıcı bir yaklaşım olarak benimsemektedir [29].

Turanoğlu ve Akkaya (2017), bölümler arası yakınlık oranlarını bir parametre olarak almış, 6 bölüm ve 5 dönemden oluşan bir DTYP çözmüştür. Oranların belirlenmesi için bulanık sistem önerilmiş, böylece çeşitli girdilerin değişkenleri birleştirilip nihai bir yakınlık oranı elde edilmesi sağlanmıştır. Sonuçlar incelendiğinde ise, önerilen modelin geleneksel model üzerinde üstünlüğü bulunduğu görülmektedir. Toplam maliyette %14,76 azalma olmuştur. Önceki çalışmalara göre yenilik içeren boyutu ise; bulanık sistem yaklaşımı ile yakınlık oranlarını tahmin etmede çevresel koşulları ergonomik bir faktör olarak görmesidir [30].

DTYP'ler, yapısı gereği karmaşık ve büyük boyutlu problemler olduğundan analitik yöntemler yerine daha kısa sürede uygun çözüme ulaşılabilen meta-sezgisel

yöntemler kullanılarak çözülmektedir. Araştırılan makalelerin neredeyse tamamında çözüme ulaşmak için meta-sezgisel yöntemlerden yararlanılmıştır.

"En iyi olan yaşar" evrimsel sürecinden esinlenilen GA'ların uygulandığı problemler incelendiğinde çok geniş bir uygulama alanı olduğu görülmektedir. Bir diğer meta-sezgisel yöntem ise; katıların belirli bir sıcaklığa kadar ısıtılıp, belirli bir süre bekletildikten sonra soğutulması ile malzemeye daha kararlı bir yapı kazandırma prosesinden yola çıkılan TB'dir. TB de, GA gibi problem bağımlı bir yöntemdir. Karıncaların yiyecek kaynağı ve yuvaları arasındaki en kısa yolu bulma davranışları örnek alınarak oluşturulan KKA'lar da bir meta-sezgisel yöntem olarak kullanılmaktadır. Çalışmalar incelendiğinde en fazla GA ve TB kullanıldığı, bu iki algoritmayı TA algoritması ve KKA'nın takip ettiği görülmektedir.

Pek çok makalede, geliştirilen algoritmaların etkinliğinin literatürde bulunan test problemleri kullanılarak ölçüldüğü görülmektedir. Bu makalelerin yanı sıra, bir işletmenin belirli sayıda bölümünün, belirli sayıda dönem için en iyi yerleşim önerilerini sunan çalışma sayısı çok azdır.

DTYP'yi çözmek için öncelikli olarak STYP'nin çözümünden faydalanılan bir yöntem de bulunmaktadır. Bu yöntemde, STYP çözülüp en iyi yerleşimlerin bulunduğu bir havuz oluşturulduktan sonra DP kullanılarak DTYP için çözümler elde edilmektedir. Ardından elde edilen çözümler çeşitli stratejiler kullanılarak geliştirilmektedir.

Sonuç olarak zor bir yapısı olan DTYP analitik yöntemlerle kabul edilebilir zamanlar içerisinde çözülememektedir. Meta-ezgisel yöntemler ise çözümde yetersiz kalmakta, bu nedenle algoritmalarda çeşitli değişiklikler yapılarak ya da birbirlerinin faydalı özelliklerinden yararlanmak için melezlenerek kullanılmaktadır.

"İncelenen DTYP makalelerinin amaç fonksiyonları göz önünde bulundurulduğunda ise neredeyse %75'inin tek amaçlı, %25'inin ise çok amaçlı olduğu görülmektedir [8]." Yerleşim tasarımının daha sürdürülebilir olması için çok amaçlı çalışmalara daha fazla ağırlık verilmelidir.

Makalelerdeki problemlerin matematiksel modelleri incelendiğinde ise; DTYP'nin çoğunlukla KAP ve karma tamsayılı programlama ile modellendiği görülmektedir.

EK-1.'de burada anlatılmış ve anlatılamamış olan DTYP ile ilgili bulunan tüm araştırmalar listelenmiştir.

4. GENETİK ALGORİTMALAR

Sözlükte genetiğin ilk anlamı, “bir şeyin kökeni, gelişimi ve nedensel öncülleri ile ilişkili olması ya da bunlar tarafından belirlenmesi” şeklinde açıklanmaktadır. Algoritma ise; “bir problemi çözmek, ya da problemin bir kısmını tamamlamak için sık sık tekrarlanan sınırlı sayıda adımın bulunduğu bir prosedür” olarak tanımlanmaktadır [2].

GA’lar, genel olarak çözümü zor problemler için kullanılmaktadır. Bir problemin kesin çözümünü bulmak ya da problemi modellemek imkânsız ya da çok zor ise, bu durumda GA’lar tercih edilmektedir. Doğadaki canlıların evrim sürecinden ilham alınarak oluşturulan bu algoritmalar, yine doğada olduğu gibi en iyi olanın neslini sürdürme olasılığının yüksek olması sayesinde, her zaman daha iyi çözümü bulmak için çözüm alanı içerisinde arama yaparlar.

“GA’lar ilk olarak 1960’larda John Holland tarafından ortaya atılmıştır. Holland, bu konuda yapmış olduğu çalışmaları 1975 yılında “Adaptation in Natural and Artificial Systems” adlı kitabında bir araya getirmiştir [11].”

4.1. Genetik Algoritmanın Faydaları ve Sakıncaları

GA’ların arama yeteneği göz önünde bulundurulduğunda; aramanın tek bir noktada değil, çözüm alanının tamamında yapılması nedeni ile büyük boyutlu problemlerin çözümünde çokça tercih edilmektedir. Tepe tırmanma gibi pek çok optimizasyon probleminde, optimum çözüme ulaşmak için bir noktadan çözüme başlanmakta ve daha iyi bir noktaya ulaşılması amaçlanmaktadır. Bu da yerel optimumlara takılıp kalma riskini artırmaktadır. GA’larda aynı anda paralel olarak birden çok çözüm incelendiğinden, yerel optimumlara takılma riski diğer algoritmalara göre çok düşüktür.

GA’ların faydaları şu şekilde belirtilmektedir;

- GA’lar, çözüm uzayı ile ilgili herhangi bir bilgiye ihtiyaç duymazlar.
- Optimizasyonun performansı üzerinde, çözüm uzayını oluşturan kısımlar arasındaki devamsızlıklar çok az etkilidir.
- Yerel optimum noktada takılı kalmaya direnç gösterirler.
- Özellikle büyük ölçekli problemlerde diğer algoritmalara göre iyi sonuç verirler.
- Sadece uyum fonksiyonu probleme özel olduğundan, çeşitli optimizasyon problemlerinin çözümü için kullanılabilirler.

Bütün bu avantajlarının yanı sıra birtakım sakıncaları da şu şekilde sıralanmaktadır;

- Bir defada çok fazla sayıda uyum fonksiyonu hesaplaması gerektirmektedir.
- Çözüm uzayında en uygun noktanın belirlenmesini garanti edemez.
- Kullanılan yapıda uyum fonksiyonu probleme özgü olduğundan, her bir problem için ayrı çalışma yapmak gerekmektedir.
- Nesiller arasında kötü özelliklerde olduğu gibi iyi özelliklerin de kaybolma ihtimali bulunmaktadır [31].

4.2. Genetik Algoritmaların Kullanım Alanları

GA'ların sadece uyum fonksiyonu probleme özgü olduğundan, uygulama alanları çok geniştir. GA'ların uygulama alanları genel uygulama alanları ve işletmelerdeki yaygın uygulama alanları olarak sınıflandırılmaktadır. Her iki sınıflandırmanın alt başlıkları da aşağıdaki şekilde aktarılmaktadır.

Genel uygulama alanları;

- Optimizasyon,
- Otomatik programlama ve bilgi sistemleri,
- Mekanik öğrenme,
- Ekonomik ve sosyal sistem modelleri,

İşletmelerdeki yaygın kullanım alanları;

- Finans,
- Pazarlama,
- Üretim/işlemler,
- Montaj hattı dengeleme problemi,
- Çizelgeleme problemi,
- TYP,
- Atama problemi,
- Hücresel üretim problemi,
- Sistem güvenilirliği problemi,
- Taşıma problemi,
- Gezgin satıcı problemi,
- Araç rotalama problemi,
- Minimum yayılan ağaç problemi [32].

4.3. Genetik Algoritmaların Temel Kavramları ve Tanımları

Genetik algoritmaların temel kavramları, tanımları ve kavramların birbirleri ile ilişkilerini anlamak problemin çözümü için büyük önem taşımaktadır. Kavramlar arasındaki bağlantıya ve algoritmanın çalışma prensibine hâkim olmak algoritmanın probleme uygunluğunun sağlanması, hem de çözümün kısa sürede elde edilmesini sağlayacak şekilde sade bir algoritma oluşturulması açısından önemlidir.

4.3.1. Gen

Tek başına bir genetik bilgi taşıyan en küçük yapıtaşına gen denilmektedir. Birden fazla gen dizilerek kendilerinden büyük olan yapıları yani dizileri/kromozomları oluşturmaktadır. GA'lardaki genler, probleme ait bir bilgiye karşılık gelmektedir. Tamamen probleme bağlı olarak problemi çözen kişi tarafından tanımlanmaktadır.

4.3.2. Kromozom (Dizi)

Bir ya da birden fazla belirli sayıda genlerin bir araya gelerek, farklı dizilimlerle oluşturdukları daha büyük yapılara kromozom ya da dizi denilmektedir. Dizi üzerindeki her bir gen problem ile ilgili bir bilgi taşıdığından, her bir dizi de problemin bir çözümüne karşılık gelmektedir. Birden fazla sayıda dizi bir araya gelerek popülasyonu yani diğer adlarıyla "yığın" ya da "kütle"yi oluşturmaktadır. Bir dizi üzerinde hangi bölgede hangi bilgiyi taşıyan genlerin dizileceği problemin türüne göre değişiklik gösterebilmektedir. Diziler, GA'ların en önemli birimi olduğundan tanımlamanın doğru yapılması büyük önem taşımaktadır.

4.3.3. Popülasyon

Popülasyon, yani yığın ya da kütle, ilgili probleme ait farklı çözümleri ifade eden belirli sayıda dizinin bir araya gelmesi ile oluşmaktadır. Bir popülasyonda kaç tane dizi bulunacağı problemin türüne ve parametrelerine göre değişiklik göstermektedir. Popülasyon değişiklikleri sırasında bazı diziler kaybolup, bazı yeni diziler oluşsa da popülasyon büyüklüğü her zaman sabit kalmaktadır.

Problemin çözüm süresi ve sonucun kalitesi popülasyon büyüklüğünden etkilenmektedir. Eğer popülasyon çok büyük ise çeşitlilik artacağından daha iyi sonuç bulma olasılığı artmaktadır. Ancak problemin çözüm süresi de büyüklük ile orantılı

olarak artış göstermektedir. Literatürde çözüm kalitesi ve süresi ile ilgili pek çok çalışma yapılmış olup, popülasyon sayısının problem türüne göre değişiklik gösterdiği görülmektedir.

4.3.4. Yeniden üretim işlemi

Yeniden üretim işlemi, mevcut popülasyonda bulunan dizilerden eldi edilen çözümlerin daha iyisini elde etmek amacı ile gerçekleştirilmektedir. En iyi çözümü sağlamış olan dizilerin bir sonraki popülasyona aktarılması ile daha iyi çözümler bulunması amaçlanmaktadır.

4.3.5. Başlangıç popülasyonunun oluşturulması

GA'lar yapısı gereği çözüme başlamak için diğer algoritmalarda olduğu gibi tek bir noktaya değil, birden fazla noktaya ihtiyaç duymaktadır. GA ile problem çözüleceği zaman ilk olarak başlangıç popülasyonu oluşturulmaktadır. Probleme göre belirlenen sayıda dizinin bir araya getirilmesi işlemi için pek çok yöntem bulunmaktadır. Başlangıç popülasyonunda yer alan diziler oluşturulurken genel olarak rastgele seçim yapılmaktadır. Bunun yanı sıra, problemin yapısına uygun bazı yöntemler kullanılarak seçim işlemi yapılması çözüme erişme zamanının kısa ve çözüm kalitesinin yüksek olması için önerilmektedir. Bu sayede popülasyonun uygun olmayan çözümlere yönelmesinin önüne geçilebilmektedir.

4.3.6. Uygunluk değeri

Bir popülasyonda yer alan dizilerin, problemin amacı doğrultusunda çözüm başarısının derecesini ifade eden bir değerdir. Uygunluk değeri için gerçekleştirilen değerlendirme işlemi, problemin amaç fonksiyonuna karşılık gelmektedir.

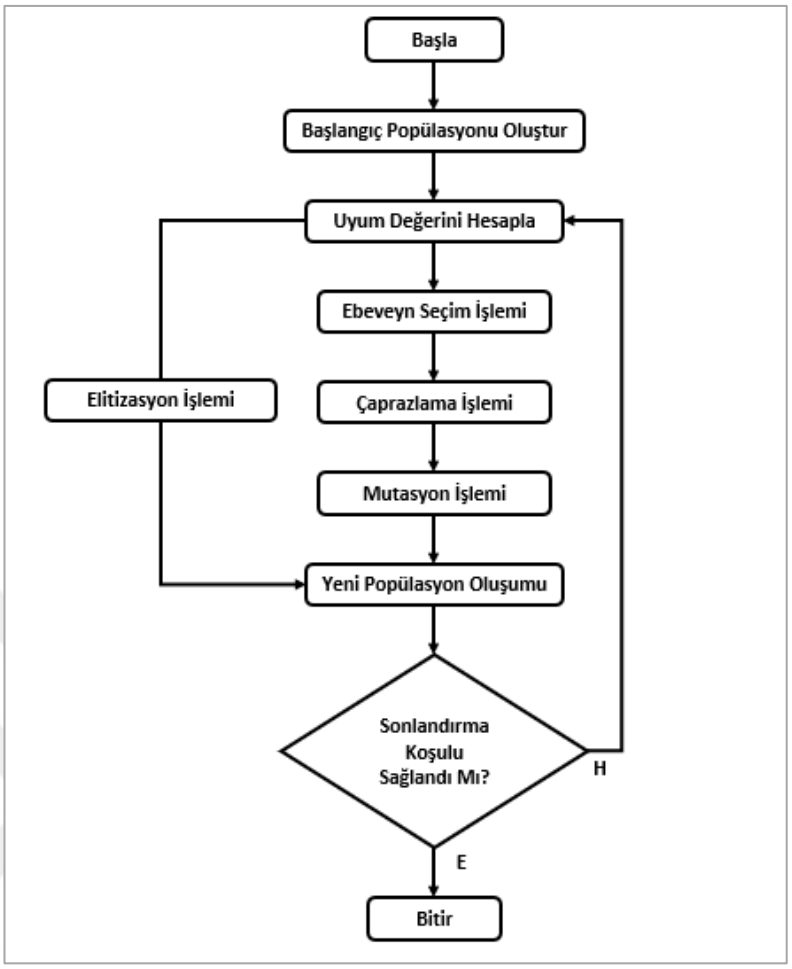
4.3.7. Genetik operatörlerin uygulanacağı dizilerin seçim işlemi

Bir popülasyon içerisinde yer alan dizilerin uygunluk değerleri karşılaştırıldığında, uygunluk derecesi en yüksek olanların seçim olasılıkları daha yüksek iken, uygunluk derecesi düşük olanların seçim şansı oldukça azdır.

4.4. Genetik Algoritmanın Adımları

GA ile problem çözümü sırasında tekrarlanan algoritma Şekil 4.1.'de gösterilmektedir. Prosedür adımları ise şu şekildedir:

- Problem için çeşitli yöntemler kullanılarak farklı çözümleri içeren bir başlangıç çözüm kümesi yani popülasyon oluşturulur.
- İkinci aşamada, oluşturulan popülasyon içerisinde yer alan her bir çözüm dizisi için problemin amaç fonksiyonu olan uyum fonksiyonu hesaplanır.
- Ardından, daha iyi bir uyum değeri elde edebilmek için yeni popülasyon oluşturulur. Bu popülasyonun oluşması için aşağıdaki işlemler uygulanır;
    - Popülasyonda yer alan iki tane dizinin çaprazlanarak yeni yavru oluşturması için seçim işlemi yapılır.
    - Seçilen bireyler yeni yavrular oluşturmak için belirtilen orana göre birbirleri ile çaprazlanırlar.
    - Çaprazlama sonrası elde edilen yeni bireyler üzerinde verilmiş olan oran kadar bireyde bazı genlerin yerleri değiştirilerek mutasyon işlemi yapılarak çeşitliliğin korunması sağlanır.
    - Bir popülasyon içerisinde bulunan en iyi çözümün kaybolmaması için bu çözümü sağlayan birey herhangi bir işleme uğramadan yeni popülasyona aktarılarak elitizm işlemi gerçekleştirilir.
    - Tüm bu adımların sonucunda yeni bir popülasyon oluşmuş olur.
- Oluşturulan bu yeni popülasyon eskisi ile yer değiştirilir.
- Yeni popülasyonlarda da aynı işlemler gerçekleştirildikten sonra döngü sonunda bitirme koşulu sağlanıyor ise algoritma durur ve en iyi çözüm elde edilir. Eğer koşul sağlanmaz ise aynı döngü devam ettirilir.

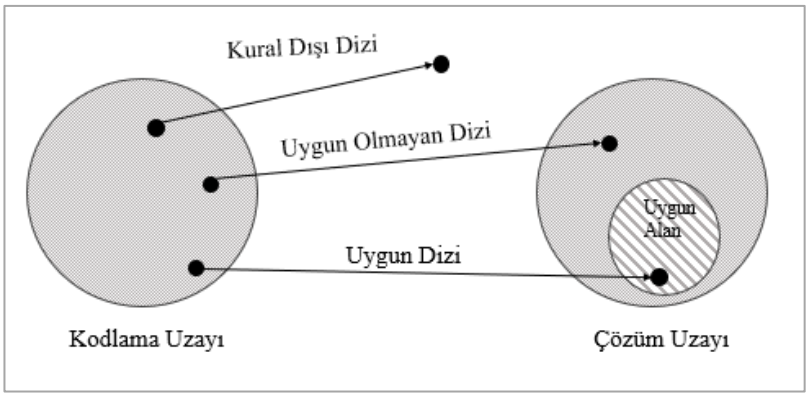


Şekil 4.1. *GA akış şeması*

4.4.1. Dizilerin oluşturulması

GA'da dizilerin oluşturulması yani dizilere genlerin kodlanması, problemin çözümüne ulaşmak için en önemli faktördür. Kodlamanın düzgün yapılması, algoritmanın verimli çalışması için gereklidir. Bu konuda dikkat edilmesi gereken üç nokta aşağıdaki şekilde belirtilmekte ve çözüm uzayı ile kod uzayı arasında bu işlemlerin nasıl gerçekleştiği Şekil 4.2.'deki gibi gösterilmektedir;

- Dizinin uygun olup olmadığı,
- Dizinin belirlenmiş kurallara uyup uymadığı,
- Dizinin kodlama haritası içerisinde tek olup olmadığı [33].



Şekil 4.2. *Kod ve çözüm uzayında uygunluk ve kontroller (Cheng vd., 1996)*

Diziler oluşturulurken farklı kodlama yöntemlerinden faydalanılmaktadır. Bu yöntemlerden bazıları şu şekildedir;

- İkili kodlama
- Permütasyon kodlama
- Değer kodlama
- Ağaç kodlama
- Gri kodlama

4.4.1.1. İkili kodlama

En yaygın kullanılan kodlama türüdür. Sayıların ikili sistemde kodlanmasına dayanmaktadır. Dizilere genetik operatörler uygulandığında her ne kadar uygun çözümü garantileme faydası bulunsa dahi, değişken sayısının fazla olduğu fonksiyonlarda çok uzun diziler oluşma sakıncası vardır.

4.4.1.2. Permütasyon kodlama

Sıralama problemlerinde kullanılan bu yönteme literatürde çok rastlanmaktadır. Bir dizi üzerinde bulunan her bir sayının bulunduğu konum, dizi ürerindeki bu sayının sırasını temsil etmektedir.

4.4.1.3. Değer kodlama

Genellikle karmaşık değerlerin kullanıldığı problemlerde tercih edilmektedir. Değerler problem özelinde herhangi bir sayı, karakter vb. her şey olabilmektedir. Bu

kodlamanın zor kısmı, değere özel yeni genetik operatör yöntemleri geliştirme gerekliliğinin bulunmasıdır.

4.4.1.4. Ağaç kodlama

Evrimleşen programlarda kullanımına rastlanmaktadır. "Ağaç gösterimi yalnızca matematiksel ifadeleri değil bilgisayar programlarını da kapsamaktadır. Operatör seti; işlemleri, tek değişkenli diğer matematiksel fonksiyonları, şartlı işlemler ve döngüler için kullanılan değimleri ve fonksiyonları da içermektedir [34].

4.4.1.5. Gray kodlama

"Gray kodlama ikili sistemde ardışık iki tamsayının tek bit değişimiyle dönüştürülmesini sağlar [34]." İkili kodlamadan farkı Hamming Uzaklığı kavramını geliştirmiş ve kullanıyor olmasıdır. Hamming uzaklığı, bir dizide değişen gen sayısını belirtir. Sayılar ikili sistemde diziye yazılmaya başlandığında ilk olarak Hamming uzaklıkları ölçülmektedir. Hamming uzaklığının 1 olması istenen durumdur [35].

4.4.2. Başlangıç popülasyonunun oluşturulması

Başlangıç popülasyonu oluşturulmadan önce ne kadar büyüklükte bir popülasyona ihtiyaç olduğu belirlenmelidir. Çünkü, çözüm uzayının büyüklüğü değişmekle birlikte sınırlıdır. Popülasyonun büyük olması çeşitliliğin sağlanması için önemli bir faktör iken, gereksiz derecede büyük popülasyonlar sadece problemin çözüm süresinin uzamasına neden olmaktadır. Farklı büyüklükteki problemler için farklı büyüklüklerde popülasyonlar literatürde önerilmekle birlikte, evrim süresi boyunca tüm çözümleri kapsayacağı düşünülen büyüklüğün belirlenmesi çözüm süresi açısından kritiktir.

Bir çalışmada 30, 50, 81 ve 100 istasyonlu dört ayrı test problemi için üçer defa program çalıştırarak parametre değerleri belirlenmiş ve popülasyon büyüklüğü 120 olarak belirlenmiştir. Özellikle 30 istasyonlu problem için optimuma çok yakın değer elde edilirken, problemin boyutu arttıkça çözüm kalitesinin düştüğünü belirtilmiştir [11].

Başka bir çalışmada ise Schaffer vd. 1989 yılında yapmış oldukları çalışmaya atıfta bulunulmuş ve çok sayıda test problemi üzerinde yapılan çalışmalar sonucunda 20-30 adet dizi içeren popülasyon büyüklüğünün iyi sonuçlar verdiği belirtilmiştir [32].

4.4.3. Uygunluk değerinin hesaplanması

Uygunluk değeri uyum fonksiyonu ile hesaplanmaktadır. Uygunluk fonksiyonu problemin amaç fonksiyonuna karşılık gelmektedir. Bir popülasyonda bulunan tüm dizilerin şifreleri çözülerek bu fonksiyon yardımı ile uygunluk değerleri belirlenir. Bu şekilde bir dizinin çevresi ile uyumu değerlendirilmekte ve iyi olan dizilerin çoğalmaları, kötü olanların ise elenmeleri sağlanmaktadır.

GA içerisinde probleme özgü çalışan tek kısım bu fonksiyondur. Algoritmanın başarısını etkileyen en önemli faktörlerden biri bu fonksiyonun her defasında düzgün çalışmasıdır. Ayrıca bu fonksiyonun kodlamasının en sade şekilde yapılması çözüm süresi üzerinde de etkilidir.

4.4.4. Yeni popülasyon oluşumu için seçim işlemi

GA'larda çeşitli yöntemlerle oluşturulan başlangıç popülasyonundan yeni popülasyon meydana getirilmektedir. Bunun amacı, daha yüksek uygunluk değerine sahip bireyler, yani daha iyi çözümler üretmektir. Bu sebeple, başlangıç popülasyonunda uygunluk değeri hesaplanmış bireyler arasından seçim yapılarak, yeni popülasyon için yavrular oluşturacak ebeveyn bireyler farklı yöntemlerle seçilmektedir.

Seçim işleminin en önemli amacı, daha yüksek uyuma sahip bireylerin seçilerek en iyi uyum değerine ulaşılmasıdır. Yavru oluşturacak bireylerin ne şekilde seçileceğine karar verildikten sonra, seçilen bireyler ile eşleşme havuzu ya da üreme havuzu denilen ara bir popülasyon oluşturulur. Ardından bu havuzdan bireyler rastgele seçilerek eşleştirilip, yeni yavrular elde edilmektedir.

Seçim işleminin yapılmasına yardımcı olan farklı mekanizmalar bulunmaktadır. Bu mekanizmalardan probleme en uygun olanı belirlenerek seçim işlemi gerçekleştirilir. Bu işlemler sırasında popülasyon çeşitliliği ve seçicilik baskısının dengelenmesi gerekmektedir. Literatürdeki mekanizmalar şu şekilde sınıflandırılmaktadır;

- Orantılı yeniden üretim mekanizması
- Sıralı yeniden üretim mekanizması
- Turnuva yeniden üretim mekanizması
- Denge durum yeniden üretim mekanizması

Bu sınıflandırmadan farklı olarak rastgele seçim yöntemi ve elitizm yöntemleri de de bulunmaktadır.

4.4.4.1. Orantılı yeniden üretim mekanizmaları

Orantılı yeniden üretim mekanizmaları kendi içerisinde dörde ayrılmaktadır. Bunlar; rulet çemberi yöntemi, beklenen değer yöntemi, kalanı stokastik örnekleme yöntemi ve evrensel örnekleme yöntemidir.

Rulet çemberi yöntemi

Uygunluk değeri en iyi olan dizinin seçim şansını artırmaktadır. Yerine koyarak stokastik örnekleme ismi ile de bilinmektedir. En sık kullanılan yöntemlerden biridir. Öncelikle popülasyon içerisindeki tüm diziler için uyum değeri hesaplanır. İkinci adımda tüm popülasyonun uygunluk değeri, yani tüm dizilerin uygunluk değeri toplamı bulunur. Bireylerin seçilme olasılığını bulmak için, her bir dizinin kendi uyum fonksiyonu, popülasyonun toplam uyum fonksiyonuna bölünür. Her bir dizi için, bulunan olasılık değerleri sınır kabul edilir. Birey sayısı kadar rassal sayı atanıp, bu sayılara karşılık gelen diziler yeni bireyler olarak alınır.

En sık kullanılan yöntemlerden biri olmasına karşın, nesil sayısında artış oldukça hata olasılığı da artmakta ve zamansız yakınsamalara neden olabilmektedir. Bunun sebebi, işlem hataları kaynaklı bireylere atanan yavru sayısının beklenenden farklı olabilmesidir.

Beklenen değer yöntemi

Bu yöntem, rulet çemberi yönteminin hatalarını önlemek için geliştirilmiş daha hassas bir mekanizma ile çalışmaktadır. Yerine koymadan stokastik örnekleme ismi ile de bilinmektedir. Rulet çemberinden farklı olarak, popülasyonun ortalama uyum değeri hesaplanmaktadır. Her bir birey için, bireyin kendi uyum değeri, popülasyonun ortalama uyum değerine bölünerek beklenen yavru sayısı hesaplanmaktadır. Her birey için beklenen yavru sayısı değeri, birey direkt olarak üremeye seçildiğinde 1, çaprazlamaya seçildiğinde ise 0.5 azaltılarak beklenenden farklı yavru sayısı kaynaklı hatalar engellenmektedir.

Kalanı stokastik örnekleme yöntemi

Rulet çemberi ya da beklenen değer yöntemi şeklinde uygulanabilmektedir. Beklenen yavru sayıları hesaplandıktan sonra, olasılığın tamsayı kısımları bireye yavru sayısı olarak atanmaktadır. Bu şekilde popülasyon büyüklüğü tamamlanamaz ise,

olasılıkların ondalık kısımlarından faydalanılmaktadır. Kesirli kısımlar, rulet çemberi yönteminde bireylere pay edilen ağırlıkları hesaplamak için kullanılmaktadır. Beklenen değer yönteminde ise, ondalık kısımlar Bernoulli denemeleri yapmak için kullanılmaktadır.

Evrensel örnekleme yöntemi

Bu yöntem ile tek bir seçim yapılarak tüm bireyler örneklenmektedir. Rulet çemberinin üzerine seçim çemberi adı verilen başka bir çemberin eklenmesi ile gerçekleştirilmektedir. Ancak bu çemberin bölmeleri birbirine eşittir. Çember dönüp durduğunda her gösterge bir bireye karşılık gelmektedir. Bu yöntemde, bir bireye birden fazla gösterge karşılık gelebilir ancak, bir bireyin bütün popülasyona hâkim olmasını engellemektedir.

4.4.4.2. Sıralı seçim yöntemi

Sıralı seçim yöntemi rank seçim yöntemi olarak da adlandırılmaktadır. Her bir birey uygunluk değerine göre en kötü 1 olmak üzere numaralandırılmaya başlanır. N büyüklüğündeki bir popülasyon için en iyi değer n numarasını almaktadır. Yöntem adını bu sıralama işleminden almaktadır. Burada bir dizinin seçilme olasılığı, sıralamada almış olduğu numaranın, tüm popülasyonda bulunan dizilerin numaraları toplamına bölünmesiyle bulunmaktadır. Sıralamaya göre olasılıklar belirlendikten sonra, rassal sayılar yardımıyla yeni popülasyondan hangi bireylerin seçileceği belirlenmektedir.

Örneğin 10 diziden oluşan bir popülasyon düşünüldüğünde, toplam puan popülasyonda 1'den 10'a kadar değerler ile numaralandırılan tüm dizilerin numaraları toplanarak 55 olarak bulunmaktadır. Burada en iyi değer 10 puan almış olacağından, seçilme olasılığı 10/55 olarak belirlenmektedir.

4.4.4.3. Turnuva seçim yöntemi

Turnuva seçim yöntemi, isminden de anlaşılacağı gibi bireylerin birbirleri ile rekabeti üzerine kurulmuştur. Bir turnuva büyüklüğü belirlenerek, popülasyon içerisinden her defasında bu sayıda birey turnuvaya katılmaktadır. Örneğin turnuva büyüklüğü 3 ise, bu 3 bireyden en iyi uyum değerine sahip olan birey yeni popülasyona kopyalanmaktadır. Bu işlem popülasyon büyüklüğü sayısı kadar yinelenerek yeni popülasyon oluşturulmaktadır.

4.4.4.4. Denge durumu seçim yöntemi

Denge durumu seçim yönteminde, her bir nesil için popülasyonda bulunan bireylerden belirlenen sayıdaki birey yok edilirken, yerine belirlenen sayıda yavru birey geçmektedir. Bu belirlenen sayı dışında popülasyonda bulunan diğer bireyler, bir sonraki nesilde de varlıklarını sürdürmektedirler.

4.4.4.5. Rastgele seçim yöntemi

Bu yöntemde bireylerin seçiminde rassal sayı üretiminden faydalanıldığı belirtilmektedir. Her bir birey 1'den başlayarak, popülasyondaki bireylerin toplam sayısına kadar sıralanmaktadır. Ardından popülasyon büyüklüğü kadar sayıda rassal sayı üretilip, her bir rassal sayı, popülasyon büyüklüğü sayısı ile çarpılmaktadır (örneğin popülasyon büyüklüğü 20 ise 20 adet rassal sayı üretilir ve her bir sayı 20 ile çarpılır). Çarpım sonucunda elde edilen değerler, yukarı yuvarlama formülü ile yuvarlanarak bir üst tam sayı elde edilmektedir. Bu şekilde işlem yapılarak, eşleştirmeler belirlenmektedir [36].

Örneğin; popülasyon büyüklüğü 4 olan bir popülasyonda, eşleşecek olan bireyler şu şekilde belirlenmektedir. 0.0061, 0,9871, 0.0489, 0.5562 olmak üzere 4 rassal sayı belirlenir. Bu sayıların her biri 4 ile çarpıldıktan sonra, bir üst tamsayıya yuvarlanarak; 1, 4, 2 ve 3 değerleri bulunur. Buna göre $dizi_1$-$dizi_4$ ile, $dizi_2$-$dizi_3$ ile eşleştirilir.

4.4.4.6. Elitizasyon yöntemi

Bir popülasyonda elde edilen uyum değeri en yüksek bireylerin sonraki nesillerde yok olmasını engellemek amacıyla kullanılmaktadır. Özellikle genetik operasyonlar uygulandıktan sonra en iyi uyum değerine sahip bireylerin yok olma riskinin önüne geçmektedir. Popülasyon içerisinde bir oran verilip, bu oranda bireyin sonraki nesle herhangi bir genetik operatör işlemi uygulanmadan aktarılması sağlanmaktadır.

"Elitizim operatörü ile, t. nesildeki en iyi α çözümün doğrudan bir sonraki topluluğa kopyalanması sağlanabilir. Böylece en iyi α çözümün yaşaması garanti altına alınır [34]."

Elitizm popülasyon oluşturmak için kullanılan seçilim yöntemleri içerisinde anlatılmış olsa da aslında bir genetik operatördür. Bir seçim mekanizması sadece elitizm işlemi ile tamamlanamamaktadır. Bu işlemi gerçekleştirmek diğer genetik operatörlere göre daha kolaydır. Ancak kaç tane α çözümün yaşatılacağının iyi belirlenmesi gerekmektedir. Eğer α değeri 0 seçilir ise, elitizm uygulanmayacaktır. Ancak pek çok

problem ile yapılan denemelerde optimizasyon üzerinde pozitif etkileri olduğu bilinmektedir. Bunun yanı sıra eğer değer büyük seçilir ise bu da yerel optimumlarda yakınsama riski oluşturmaktadır. "N topluluk büyüklüğü olmak üzere, $0 \leq \alpha \leq 0.1N$ olarak seçilebilir [34]."

4.4.5. Genetik algoritmalarda kullanılan genetik operatörler

GA'larda en iyi çözüme ulaşabilmek için çözüm uzayında yapılan aramanın güçlü olması gerekmektedir. Çözüm uzayındaki aramayı güçlendiren en önemli faktörlerden biri de her bir nesilde çeşitliliğin sağlanmasıdır. Nesillerde çeşitlilik sağlanabilmesi için popülasyonlar içerisinde yer alan bireylere çaprazlama ve mutasyon operatörleri belirlenen oranlarda uygulanmaktadır. Mutasyon operatörü literatürde karşımıza değişim operatörü ya da dönüşüm operatörü isimleriyle de çıkmaktadır.

Pek çok çaprazlama ve mutasyon operatörü türü bulunmaktadır. Hangilerinin kullanılacağına probleme göre karar verilmekle birlikte, bazı problemlerin çözümü için daha farklı, probleme özgü yöntemler de tercih edilebilmektedir.

4.4.5.1. Çaprazlama operatörü

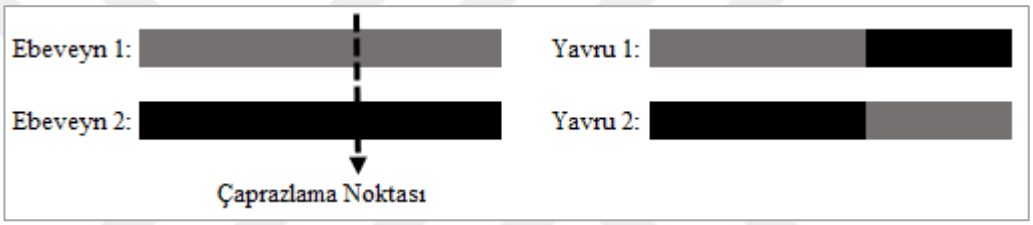
Çaprazlama operatörü, çözüm uzayında bulunmayan bireyleri üretmek için kullanılan önemli bir operatördür. Popülasyonda bulunan bireylerin karşılıklı olarak belirli metotlar çerçevesinde gen değişimi yapması işlemidir. Bu gen değişimleri sonucunda ortaya yeni bireyler çıkmaktadır. Hangi oranda çaprazlama yapılacağı da çaprazlama metodu kadar önemlidir. Çaprazlama oranı Pc olarak ifade edilmekte ve genele bakıldığında %55 ile %95 arasında bulunan değerler seçilerek uygulanmaktadır. Düşük çaprazlama oranı, en iyi sonuca ulaşma süresini uzatabilirken, yüksek çaprazlama oranı da en iyi çözümün kaçırılmasına yol açabilmektedir. Çaprazlama operatörlerinin çeşitleri şu şekildedir;

- Basit çaprazlama operatörü (tek noktalı ve iki noktalı)
- Uniform çaprazlama operatörü
- Kes-ekle çaprazlama operatörü
- Kısmi sıralı çaprazlama operatörü
- Sıralı çaprazlama operatörü
- Pozisyon tabanlı çaprazlama operatörü

- Sıra tabanlı çaprazlama operatörü
- Dairesel çaprazlama operatörü
- Sezgisel çaprazlama operatörü

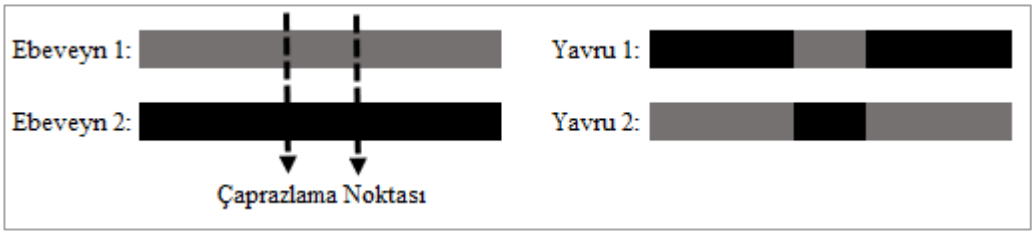
Basit çaprazlama operatörü

Belirlenen sayıdaki noktanın dizi üzerinde seçilerek gerçekleştirilen çaprazlama yöntemidir. En çok bir ve iki noktalı olanlar kullanılmaktadır. Bir noktalı çaprazlamada, iki ebeveyn dizi için belirlenen bir noktadan (iki gen arası) kesme işlemi yapılır. Dizilerin sağ tarafı sabit kalmak koşulu ile sol tarafları değiştirilerek çaprazlama gerçekleştirilir. Şekil 4.3.'te tek noktalı çaprazlama örneği görülmektedir.



Şekil 4.3. *Tek noktalı basit çaprazlama örneği*

İki noktalı çaprazlamada ise iki adet sabit nokta belirlenir ve ebeveyn genler bu noktalardan kesilerek, ortada kalan kısımlar birbirleri ile yer değiştirir.Şekil 4.4.'te iki noktalı çaprazlama örneği görülmektedir.



Şekil 4.4. *İki noktalı basit çaprazlama örneği*

Basit çaprazlama yöntemi genellikle ikili kodlanmış bireylerin çaprazlama işleminde kullanılmaktadır.

Tekdüze çaprazlama operatörü

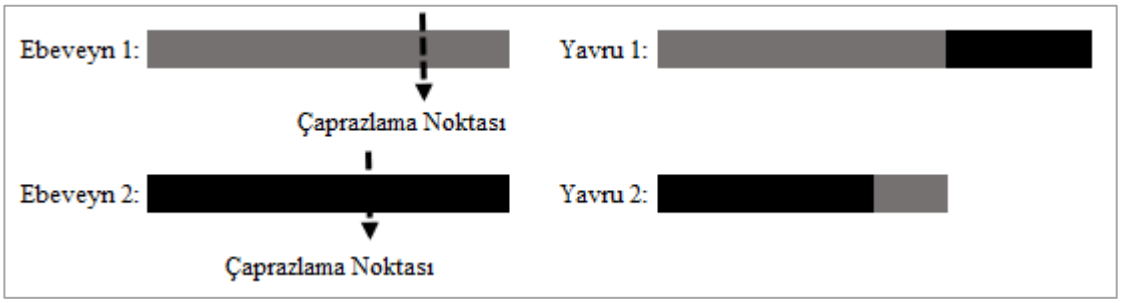
Bu yöntem ikili kodlama yapılmış bireylerin çaprazlanması için kullanılmaktadır. Çaprazlamanın gerçekleşmesi için iki ebeveynin yanında bir de maske birey bulunmaktadır. Bu maske bireyin dizi uzunluğu da ebeveyn genler ile aynı olmakla birlikte, genlerinin 0 ya da 1 olması durumuna göre yavru bireylerin hangi ebeveynden gen alacakları belirlenmektedir. Şekil 4.5.'te görüldüğü gibi, yavru1'in genleri, maskede ilgili sıradaki genin değeri 1 ise ebeveyn 1'den, 0 ise ebeveyn 2'den gen almaktadır. Yavru 2'nin genleri ise eğer maskede o gene karşılık gelen değer 0 ise ebeveyn 1'den, 1 ise ebeveyn 2'den gen almaktadır.

Ebeveyn 1:	1	1	0	0	0	Yavru 1:	1	0	0	0	1
Ebeveyn 2:	1	0	1	1	1	Yavru 2:	1	1	1	0	0
Maske:	1	0	1	1	0						

Şekil 4.5. *Uniform çaprazlama örneği*

Kes-ekle çaprazlama operatörü

Ebeveyn 1 ve 2'nin her biri için rassal olarak bir kesme noktası belirlenir ve noktanın sağında kalan genler değiştirilerek yavru bireyler oluşturulur. Bu çaprazlama yöntemi ile dizilerin boylarında değişme riski bulunmaktadır. Şekil 4.6.'da bir örnek gösterim bulunmaktadır.

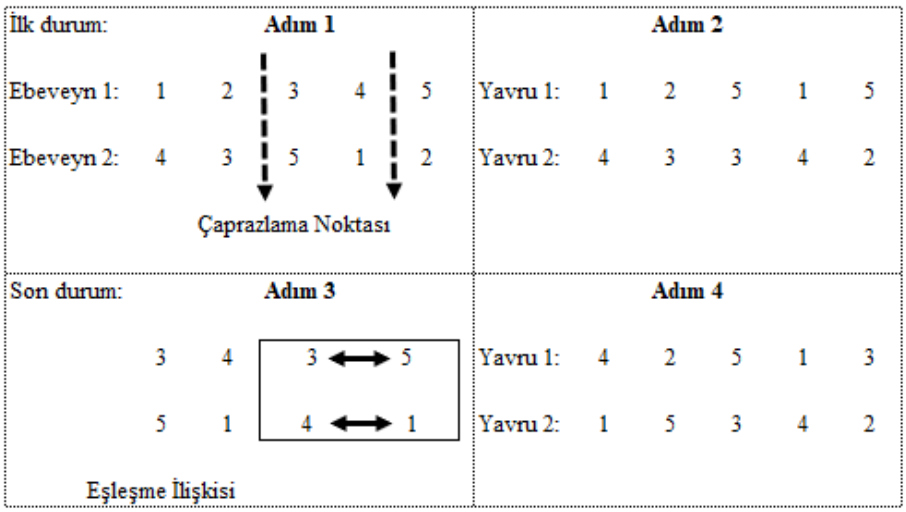


Şekil 4.6. *Kes-ekle çaprazlama örneği*

Kısmi sıralı çaprazlama operatörü

İki noktalı basit çaprazlamanın, ikili kodlama dışındaki yöntemlerle kodlanmış olan dizilerde kullanılan yöntemidir. Belirlenen iki nokta arasında kalan genler yer

değiştirdikten sonra, bir dizi üzerinde aynı genden iki tane bulunabilmektedir. Bu durum ile karşılaşıldığında ise, çaprazlama yapılırken yer değiştiren genlerin, ebeveyn dizilerdeki karşılıkları bulunup, çaprazlanmayan kısımlardaki genler ile karşılıklı değişimi sağlanır. Şekil 4.7.'de bir örnek görülmektedir.

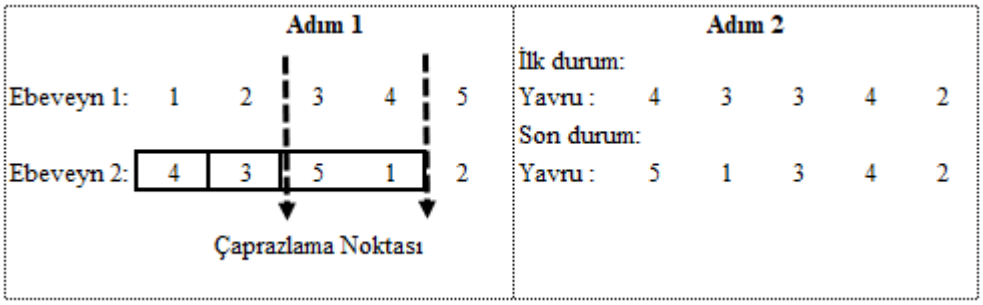


Şekil 4.7. *Kısmi sıralı çaprazlama örneği*

Sıralı çaprazlama operatörü

Kısmi sıralı çaprazlama operatörüne benzemektedir. Bu yöntemin farkı, ebeveyn 1'den yavruya aktarılan genlerin ebeveyn 2'den silindikten sonra aktarılmamış olan genlerin sırası ile yavruya kopyalanmasıdır.

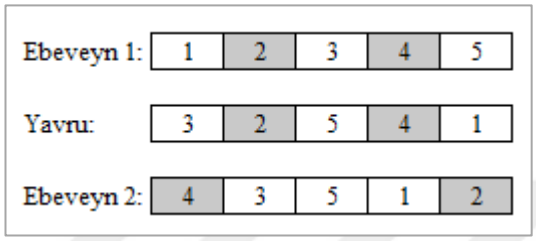
Şekil 4.8.'de görüldüğü gibi ebeveyn 1'den yavruya aktarılan 3 ve 4 genleri ebeveyn 2'den silinmiş, bunun yerine yavruda aynı bulunan 4 ve 3 genleri yerine sırası ile 5 ve 1 genleri kopyalanmıştır.



Şekil 4.8. *Sıralı çaprazlama örneği*

Pozisyon tabanlı çaprazlama operatörü

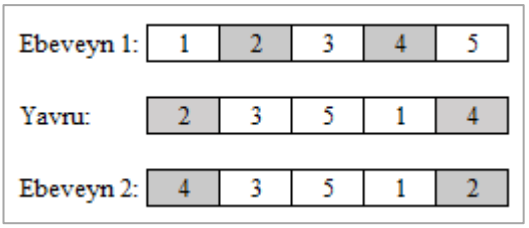
Bu çaprazlama türünde ebeveyn 1'den rassal sayıda rassal gen tutulmaktadır. Tutulan bu genler yavru dizide aynı gen konumuna denk gelecek şekilde kopyalanır. Ardından ebeveyn 1'den yavruya kopyalanan genler ebeveyn 2'den silinir ve ebeveyn 2'nin kalan diğer genleri yavru bireyde boş noktalara sırasıyla yerleştirilir. Bir örnek Şekil 4.9.'da gösterilmektedir.



Şekil 4.9. *Pozisyon tabanlı çaprazlama örneği*

Sıra tabanlı çaprazlama operatörü

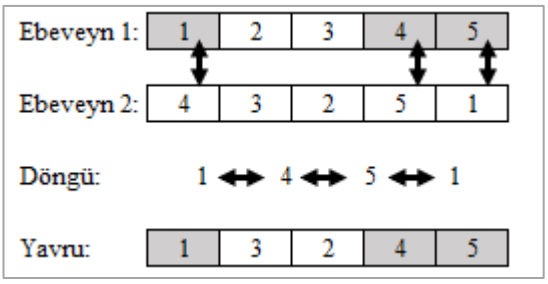
Pozisyon tabanlı çaprazlamanın farklı bir versiyonudur. Bu çaprazlamada ebeveyn 2'de silinmemiş olan genlerin dizi üzerindeki konumları önemlidir ve yavru dizide bu konum ve sıra korunarak çaprazlama işlemi gerçekleştirilmektedir. Şekil 4.10.'da bir örnek görülmektedir.



Şekil 4.10. *Sıra tabanlı çaprazlama örneği*

Dairesel çaprazlama operatörü

İki ebeveyn dizi arasında bir döngü oluşturduktan sonra gerçekleşen çaprazlama yöntemidir. Döngü her zaman birinci ebeveynden başlatılmaktadır. Döngünün başladığı gen bulunduğu anda durulmalıdır. Döngü içerisinde yer alan bütün genler ebeveyn 1'den yavruya kopyalanır. Ardından ebeveyn 2'de bulunan ve döngü dışında kalan genler aynı sıra ile yavru bireye kopyalanarak çaprazlama tamamlanır. Bir örneği Şekil 4.11.'de gösterilmektedir.



Şekil 4.11. *Dairesel çaprazlama örneği*

Sezgisel çaprazlama operatörü

Bir popülasyonda bulunan diziler ikili kodlama yöntemi ile kodlanmışsa kullanılabilmektedir. Genlerin bir kısmını bir ebeveynden, kalan kısmını da diğer ebeveynden almayı sağlar. Genlerin ne kadarının hangi ebeveynden alınacağına karar verirken 4.1 numaralı denklemi kullanılmaktadır. Bu denklem sayesinde dizilerin uzunlukları her zaman aynı kalmaktadır. Bunun sebebi ebeveyn 1'den alınacak gen oranı ile, ebeveyn 2'den alınacak gen oranlarının toplamının 1 olmasıdır. Yani her durumda oluşan yavru dizilerin uzunluğu ebeveyn diziler kadar olmaktadır. Denklemde bulunan α değeri [0,1] aralığındadır.

$$Yavru = \quad Ebeveyn1 * \alpha + \quad Ebeveyn2 * (1 - \alpha) \tag{4.1}$$

4.4.5.2. Mutasyon (dönüşüm/değişim) operatörü

Henüz gerekli çeşitlilik içerisinde arama yapılmadan GA'nın yakınsaması yerel minimumlara takılma riskini artırdığından, en iyi sonuca ulaşamama durumu ortaya çıkmaktadır. GA'ların gerekli nesil çeşitliliği sağlanmadan yakınsaması durumuna engel olması için kullanılan önemli bir operatördür.

Bu operatör popülasyon içerisindeki belirli orandaki bireyin bir ya da birden fazla genini rastgele değiştirerek yeni bireyler üretilmesine katkı sağlamaktadır. Operatörün uygulanma olasılığı Pm ile ifade edilirken, genel olarak çok küçük oranlarda uygulanmaktadır.

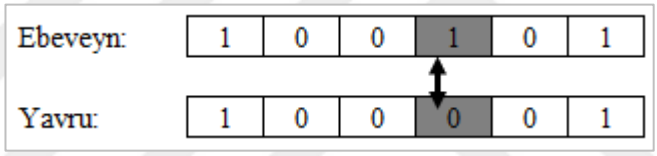
Çaprazlama operatörlerinde olduğu gibi, mutasyon operatörlerinin de çeşitleri bulunmaktadır. Başlıca mutasyon operatörleri şu şekildedir;

- Basit mutasyon
- Ekleme mutasyonu
- Ters çevirme mutasyonu

- Yer değiştirme mutasyonu
- Ters çevirmeli yer değiştirme mutasyonu
- Karşılıklı değişim mutasyonu

Basit mutasyon

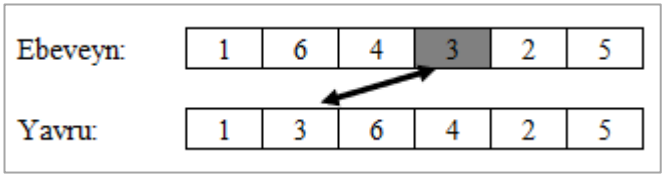
Genellikle ikili kodlama yöntemi ile kodlanmış diziler üzerinde tercih edilmektedir. Dizi üzerinde herhangi bir mutasyon noktası seçilerek, seçili gen 0 ise 1, 1 ise 0 yapılarak gerçekleştirilmektedir. Permütasyon kodlama ile kodlanmış dizilerde bu mutasyon yöntemi tercih edilmemektedir. Permütasyon kodlama ile kodlanan dizilerde ikili kodlama ile kodlanmış dizilerdeki gibi, seçilen gen yerine yavru gene atanacak, seçilen gen ile yer değiştirebilecek bir gen bulunmamaktadır. Şekil 4.12.'de basit mutasyona bir örnek gösterilmektedir.



Şekil 4.12. *Basit mutasyon örneği*

Ekleme mutasyonu

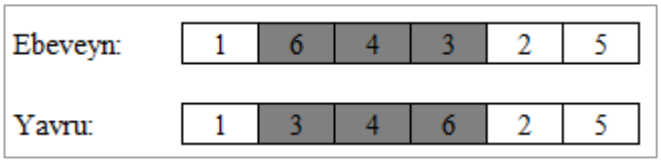
Dizi üzerinde seçilen rassal bir genin, yine dizi üzerinde rassal bir noktaya eklenmesi ile gerçekleştirilmektedir. İster ikili kodlama yöntemi ile kodlanmış dizi, isterse de permütasyon yöntemi kodlanmış dizi olsun, seçilmiş olan rassal bir gen ve kopyalanacak olan rassal aralık belirlendiğinde her iki kodlama yöntemi için de kullanılabilmektedir. Ekleme mutasyonuna bir örnek, Şekil 4.13.'te gösterilmektedir.



Şekil 4.13. *Ekleme mutasyonu örneği*

Ters çevirme mutasyonu

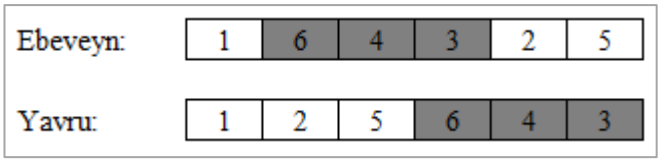
Bir dizi üzerinde iki tane mutasyon noktası belirlenip, bu iki nokta arasında kalan genlerin ters çevrilerek aynı konuma yerleştirilmesi ile gerçekleştirilen mutasyon çeşididir. Hem ikili kodlama yöntemi ile kodlanmış, hem de permütasyon yöntemi ile kodlanmış diziler için kullanılabilmektedir. Şekil 4.14.'te ebeveyn diziden alınıp, yavru diziye ters çevrilerek aynı noktaya kopyalanmış olan genler görülmektedir.



Şekil 4.14. *Ters çevirme mutasyonu örneği*

Yer değiştirme mutasyonu

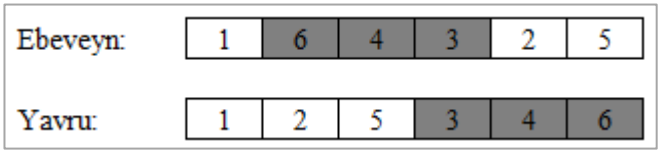
Dizi üzerinde belirlenen iki nokta arasındaki genlerin bir alt dizi şeklinde alınıp, aynı dizinin herhangi bir noktasına eklenmesi şeklinde gerçekleştirilmektedir. Şekil 4.15.'te bir örnek bulunmaktadır. Burada, dizinin hangi noktasına ekleme yapılacağı rastgele belirlenmektedir.



Şekil 4.15. *Yer değiştirme mutasyonu örneği*

Ters çevirmeli yer değiştirme mutasyonu

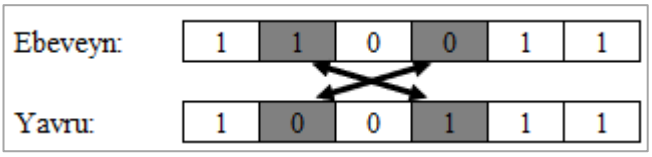
Dizi üzerinde belirlenen iki nokta arasındaki genlerin bir alt dizi şeklinde alınıp, aynı dizinin rastgele seçilen herhangi bir noktasına ters çevrilerek eklenmesi şeklinde gerçekleştirilmektedir. Şekil 4.16.'da örnek gösterilmektedir.



Şekil 4.16. *Ters çevirmeli yer değiştirme mutasyon örneği*

Karşılıklı değişim mutasyonu

Dizi üzerinde rassal olarak iki genin seçilerek, karşılıklı olarak birbirlerinin yerine geçmeleri ile gerçekleştirilmektedir. Oluşan yavru dizide diğer tüm genler konumunu ve kimliklerini korurken, sadece seçilmiş olan iki genin konumu ebeveynden farklıdır. Şekil 4.17.'de bir örnek uygulaması görülmektedir.



Şekil 4.17. *Karşılıklı değişim mutasyonu örneği*

4.4.6. Genetik algoritmaların kontrol parametreleri

GA'larda popülasyon büyüklüğü, çaprazlama olasılığı, mutasyon olasılığı, seçilim, çaprazlama ve mutasyon yöntemleri algoritmanın verimini etkilemektedir. Popülasyon büyüklüğünün yanı sıra, dizilerin uyumlarının arama uzayında ne şekilde arandığı da önemlidir. Popülasyon büyük ve dağılımı homojen gerçekleşmiş olsa bile, diğer parametrelerin etkisi ile optimum çözümün elde edilemediği durumlar da oluşmaktadır.

Çözüm uzayının birçok bölgesini temsil edebilecek çözümler üretmek için popülasyon büyüklüğünün çok büyük belirlenmesi iyi bir fikir olarak düşünülse de çalışma zamanını çok uzattığı için tercih edilmemektedir.

"Literatürde genel olarak mutasyon olasılığı 0.005-0.05 arası ve çaprazlama olasılığı 0.5-1 arası seçilmektedir (Satman, 2016)." Çaprazlama oranı %100 ve mutasyon oranı %0 iken, yeterli popülasyon büyüklüğü durumunda optimum sonucun elde edileceği beklense de bu çok iyimser bir yaklaşımdır. Çaprazlama olasılığı %100 olduğunda yeni oluşan popülasyonda tamamen yeni yavrular bulunacaktır. Bu noktada bir nesilde elde edilen iyi çözümlerin kaybolması ihtimali çok yüksektir. Gelecek nesillerde uyum oranı yüksek dizilerden yavru elde edilmesi olasılığı düşük kalmaktadır. Mutasyon olasılığı çok küçük seçildiğinde algoritma yerel optimumlara takılabilirken, büyük seçildiğinde ise yakınsaması zor olabilmektedir.

Bunun tersi düşünüldüğünde yaşanacak olan durum şu şekilde belirtilmektedir: "Çaprazlama olasılığı %0 ve mutasyon olasılığı %100 olarak seçildiğinde GA rassal arama algoritmasına dönüşür ki, tarihsel gelişimi tersine çevirmiş oluruz [34].

5. DİNAMİK TESİS YERLEŞİMİ PROBLEMİ ÜZERİNE BİR UYGULAMA

Bu bölümde uygulama yeri ve süreçlerden bahsedilecektir. DTYP ile ilgili problemin tanıtımı yapılıp, mevcut durum ile ilgili bilgi verilecektir. Problemin çözümü için kabul edilen varsayımlar ve çözüm yöntemi açıklandıktan sonra iyileştirme sonrası elde edilen öneri ve iyileşmeler aktarılacaktır.

5.1. Uygulama Yeri Hakkında Bilgi

Problem, üretim sektöründe seramik sağlık gereçleri üretimi yapan bir firmanın kalite ayrım sonrası operasyonlar bölümünde iyileştirme yapmak üzere ele alınmıştır. Firma, yapı ve inşaat sektörü ile, yapı malzemeleri satışı yapan perakende sektörü için üretim yapmaktadır.

Üretim süreçleri incelendiğinde genel olarak ağır makine, ekipman ve kurulduktan sonra yerinin değiştirilmesinin imkânsız olduğu büyük fırınlar sayesinde üretim faaliyetlerinin gerçekleştirildiği görülmektedir. Seramik sağlık gereçleri üretim operasyonları ana hatları ile göz önüne alındığında, şekillendirme, kurutma, sırlama ve pişirme süreçlerinde bulunan makine ve ekipmanların yer değiştirme maliyetleri çok yüksektir. Seramik sağlık gereçleri üretim sürecinin temel adımları şunlardır:

1. Şekillendirme: Ürün tasarım kriterlerine göre hazırlanmış kalıplara, hammadde değirmenlerinde üretilen çamurun dökülmesi ile şekillendirme operasyonu gerçekleştirilmektedir. Boru hatları ile yapılan çamur beslemeleri ve kalıpların bağlandığı tezgahların kurulum ve ayarlarının uzun süren zor operasyonlar sonucu kurulması nedeni ile şekillendirme hatlarının dinamik olarak yer değiştirmesi çok zordur.
2. Kurutma: Kalıplardan çıkan yaş yarı mamuller, çamurun içerisinde bulunan suyun belirli bir zaman ekseninde ürün bünyesinden atılması için kurutma fırınlarına alınarak kurutulmaktadır. Fırınların boyutlarının taşınmaya elverişli olmaması nedeni ile dinamik olarak yer değiştirmesi söz konusu değildir.
3. Sırlama: Kurutma operasyonunun tamamlanmasının ardından ürünlerin yüzeyi sırlama istasyonlarında sır ile kaplanmaktadır. Sırlama istasyonları bahsedilen makine, ekipman ve fırınlar kadar büyük ekipmanlar olmasa da kendinden önceki proseslere göre yerleşim planlaması yapıldığından, bu istasyonda genellikle çok sık yeniden düzenleme ihtiyacı bulunmamaktadır.

4. Pişirme: Sırlanan ürünler yüksek sıcaklıktaki tünel fırınlarda pişirilerek, ürün yüzeyinde bulunan sıra, hijyeni sağlayan camsı özellik kazandırılmaktadır. Bu proseste kullanılan fırınlar yaklaşık olarak 100 metre olup, herhangi bir duruş için söndürülmesi ve tekrar yakılması uzun zaman almaktadır. Herhangi bir yer değişikliği ihtiyacı nedeni ile fırının sökülüp yeniden inşa edilmesi ve bu süreçte yaşanacak üretim kaybının yaratacağı maliyete katlanmak, ürün taşıma maliyeti ile kıyaslandığında kesinlikle kabul edilemeyecek boyuttadır. Bu nedenle seramik üretimi yapılan işletmelerde en önemli ekipmanlardan olan fırınların yerleri ciddi çalışmalar ile belirlenmekte ve genellikle yer değişikliği yapılması söz konusu olmamaktadır.
5. Kalite ayrım: Fırından çıkan ürünler kalite ayrım operasyonunda kalite veya ıskarta etiketi almaktadır. Iskarta etiketi alan ürünler kırılıp imha edilmek için kırık alanlarına gönderilmektedir. Kalite etiketi alan ürünler ise, kalite ayrım sonrası operasyonlar bölümüne gönderilmektedir.
6. Kalite ayrım sonrası operasyonlar: Ürünün ihtiyacı olan kozmetik düzeltmeler ve montaj işlemlerinin tamamlanarak, ürünler paketlemeye hazır hale gelmektedir. Seramik sağlık gereçleri üretiminin diğer adımları ile kıyaslandığında, çok küçük iş istasyonlarında operasyonlar gerçekleştirilmektedir. İş istasyonları küçük olsa da gerçekleştirilen operasyonlar ürünün müşterinin istediği özellikleri kazanması için önemlidir.
7. Paketleme: Sevkiyata hazır ürünler, ürün özelliklerine göre paketlenerek, depoya veya sevkiyat aracına gönderilmektedir. Ürün çeşitliliğinin fazla olması ve bir çeşit ürün üzerinde farklı özellikler bulundurması nedeni ile paketleme prosesi karmaşıktır ve geniş bir alana ihtiyaç duymaktadır.

Yer değiştirme maliyetlerinin nispeten daha düşük olduğu ve tezgahlar arasındaki taşımaların daha çok önem taşıdığı işletmelerde belli dönemlerde yerleşim düzenlemesi sayesinde daha yüksek kazançların sağlanması mümkündür [12]. Bu nedenle çalışma, makine ve ekipmanların yer değiştirme maliyetlerinin çok düşük olduğu, makine ve ekipmanların kolay hareket edebildiği, ürün taşımanın fazla, ürünler ve proseslerde dinamikliğin yoğun olduğu kalite ayrım sonrası operasyonlar bölümü için gerçekleştirilmiştir.

İlgili işletmenin kalite ayrım sonrası operasyonlar bölümündeki operasyonlar incelendiğinde, zımpara, iç yüzey tamiri, yüzey tamiri, delik tamiri, taşlama, yüzey

kaplama, vakum testi, perde kesme, boru montajı, koku testi ve montaj olmak üzere 11 operasyon olduğu görülmektedir. Kalite ayrım sonrası operasyonlar:

1. Zımpara: Ürünlerin sifon bölgelerinde bulunan yüzey pürüzlerini gidermek için gerçekleştirilmektedir. Bu işlem sayesinde montaj sırasında yüzeylerin birbirleri ile tam örtüşmesi sağlanmaktadır.
2. İç yüzey tamiri: Kalite almış ürünlerin kozmetik yüzey hatalarının giderilmesi için sırsız yüzeylerde gerçekleştirilen tamir işlemidir.
3. Yüzey tamiri: Kalite almış ürünlerin kozmetik yüzey hatalarının giderilmesi için sırlı yüzeylerde gerçekleştirilen tamir işlemidir.
4. Delik tamiri: A tipi klozetlerin temiz su giriş ve kirli su çıkış delik çevrelerinde bulunan sırsız yüzeylerinin tamir işlemidir.
5. Taşlama: Kalite almış olan A tipi klozetlerin sırt yüzeylerinin duvara sıfır dayanması, B tipi klozetlerin ise tabanlarının yerde hareket etmemesi için, yüzeylerinin taşlanarak tüm yüzeyin eşit hale getirilmesini sağlamaktadır.
6. Yüzey kaplama: Önceden üretim sürecinde bulunmamasına rağmen, hijyene verilen önemin artması sonucu değişen müşteri talepleri doğrultusunda kalite ayrım sonrası operasyonlarına eklenmiştir. Bu süreçte ürünlerin yüzeyleri sıvı bir madde ile kaplanarak yüzey gerilimleri düşürülerek temas açısı küçültülmektedir. Kazandırılan bu özellik sayesinde ürün yüzeyinde sıvılar tutunamamakta, kirler sıvılar ile yüzeyden kayarak akıp gitmektedir.
7. Vakum test: Üretilmiş olan seramik sağlık gereçlerinin herhangi bir noktasında çatlak olup olmadığının kontrolünü gerçekleştirmek için tüm ürünlere uygulanmaktadır. Ürünün içerisine negatif basınç uygulanması ile gerçekleştirilen test sayesinde müşteriye hatalı ürün gönderilmesinin önüne geçilmektedir.
8. Perde kesme: B tipi klozetlerde klozetin arka alt bölümünde bulunan seramik perde parçasının kesilmesi için gerçekleştirilmektedir.
9. Boru montajı: Ürünün kullanıma hazır olması için gerekli su borularının ürün üzerine monte edilmesi için gerçekleştirilen bir operasyondur. Ürünün özelliklerine göre monte edilen boruların türleri değişmektedir.
10. Koku testi: Vakum testi mantığı ile gerçekleşse de negatif basınç ürünün tamamına değil, temiz su giriş ve pis su çıkış delikleri arasına uygulanmaktadır. Vakum test makinesinden farklı tasarlanmış olan başka bir makinede gerçekleştirilmektedir.

11. Montaj: Tüm ürünlerin fonksiyon testlerinin uygulandığı ve eğer ürünün su kanalı bulunmuyor ise suyu dağıtan rozetin montajının yapıldığı operasyondur.

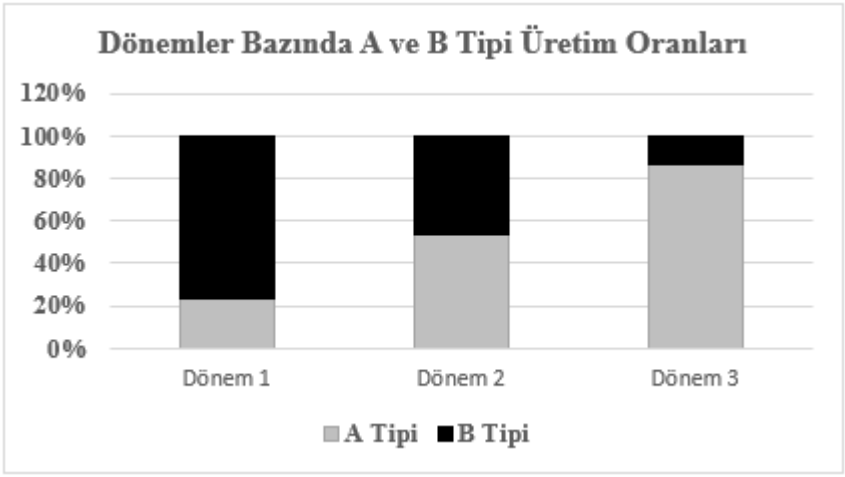
5.2. Problemin Açıklanması ve Mevcut Durum Analizi

Uygulamanın yapıldığı yerde, belirli dönemlerde müşteri talepleri doğrultusunda ürün tasarımları değişmektedir. Bu tasarım değişiklikleri, ilgili dönemin dekorasyon modasını yakalamak ve müşterinin hayatını kolaylaştıracak ürünlerin üretimini sağlamak için gerçekleştirilmektedir. Kalite ayrım sonrası operasyonlar bölümünde müşteri talepleri doğrultusunda değişen ürün tasarımlarına bağlı olarak üretim proseslerine yenileri eklenebildiği gibi bazı proseslere de zaman içerisinde gerek kalmadığı görülmektedir. Operasyonların gerçekleştiği iş istasyonlarının sayıları; makine kapasiteleri, ürün talepleri ve yönetim kararları doğrultusunda değişiklikler gösterebilmektedir. Ürün grupları arasındaki talep yoğunluklarındaki değişikliklere göre bazı makine ve ekipmanlar diğer ürün grubu için de kullanılabilmektedir.

İşletme, büyüklüğü nedeni ile, üretilen ürün grupları ve üretim yöntemleri gibi çeşitli özellikler göz önüne alınarak daha küçük tesislere ayrılmıştır. İşletmeye yeni bir tesis daha açılması, belirli ürün gruplarının ağırlıklı olarak farklı tesislerde üretilmesi kararını ortaya çıkartmıştır. Yeni açılan tesiste üretim kapasitesinin fazlar halinde artırılmasıyla birlikte, çalışmanın yapılmış olduğu tesisten bir ürün grubunun üretiminin aşamalı bir şekilde yeni tesise aktarılması, diğer ürün grubunun da kalan kapasiteyi ağırlıklı olarak tek başına kullanması durumu ortaya çıkmıştır. Bu nedenle dönemsel olarak tesis içerisindeki istasyonlar arası akış sayılarının değişiklik göstereceği bilinmektedir. Problemin ana sebebi bu geçiş sürecinde iş istasyonları arasındaki malzeme akışlarının değişkenlik gösterecek olmasıdır.

Uygulamanın gerçekleştirildiği tesiste klozet üretilmektedir. İşletmedeki ürün sayısı çok fazla olduğundan ürün bazında yerleşim yapmak imkânsız görülmüştür. Üretilen tüm ürünler ve prosesleri incelenmiş, üretimi yapılan ürünler özelliklerine göre A tipi ve B tipi olarak sınıflandırılmıştır. Tüm ürünler bu iki ürün gurubu içerisinde yer almaktadır. Dönemler bazında her iki ürün grubunun talep tahminleri ile ilgili analizler sonucunda A tipi ve B tipi ürünlerin üretim oranları arasında geçişler olduğu görülmektedir. Şekil 5.1.'de yapılan analiz sonucunda elde edilen grafik bulunmaktadır. Grafikte de görüldüğü gibi A tipi ürünün üretim hacmi birinci dönemde %20 iken, ikinci dönemde %55'i, üçüncü dönemde ise %80'i geçecektir. Bunun bir sonucu olarak, B tipi

ürün grubuna göre yapılmış olan tesis yerleşiminin değiştirilmeden kullanılmaya devam etmesi, taşıma maliyetlerini artıracaktır.



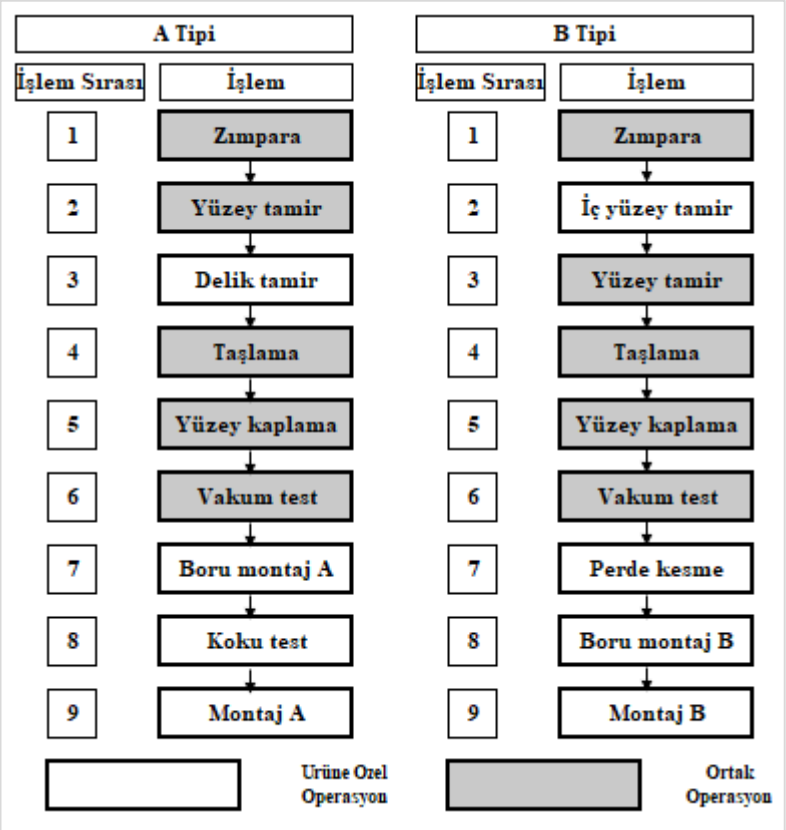
Şekil 5.1. *A ve B tipi ürünlerin dönem bazlı üretim oranları*

Bölümde toplam 27 tane iş istasyonu bulunmaktadır. Bunlardan bir kısmı her iki ürün grubu için kullanılmakta olup, bazıları sadece bir ürün grubuna özel iş istasyonlarıdır. Bölümün toplam üretim kapasitesi dönemler arasında çok değişmemekle birlikte, mevcut durumda ürünlerin üretim adetlerindeki değişimlere göre makine kapasiteleri incelenmiş ve yeterli olduğu görülmüştür. Ürün gruplarına özel iş istasyonlarının bazılarında kullanılmayan kapasiteler mevcuttur. Bu iş istasyonları her vardiyada çalışmamaktadır. Çalışma yapılmayan vardiyalarda bu iş istasyonunda işlenecek olan ürünler, yarı mamul arabalarında tezgâhın yanında ara stok olarak bekletilmektedir. Vardiya başladığında hattan gelen ürünlerle birlikte, stoklar da tüketilmektedir. Bu durum göz önüne alındığında yeni iş istasyonu ya da ekipman yatırımına ihtiyaç bulunmamaktadır.

Bazı proseslerin ürün akışından çıkartılması gerektiği durumlarda bu iş istasyonundan arta kalan boş alanlar nedeni ile diğer iş istasyonları arasında sağlanan malzeme akışı için bu boş mesafenin kat edilmesi gerekmektedir. Bir iş istasyonuna zaman içerisinde ihtiyaç kalmaması sonucunda diğer iki istasyon arasındaki taşıma mesafesinin yaratacağı taşıma maliyetinin azaltılması için dinamik tesis yerleşimi önem kazanmaktadır. Bazı iş istasyonlarının eklenmesi gerektiği durumlarda ise, bu iş istasyonları için kullanılabilir bir alana gereksinim duyulmaktadır. Ele alınan bu problemde de sayıca çok az ürüne deneme ya da özel istek sonucu uygulanan yüzey

kaplama prosesinin, yapılan maliyet iyileştirme çalışmaları sonucunda tüm ürünlere uygulanması kararı, bu iş istasyonunun da prosese eklenmesi gerekliliğini doğurmuştur. Gelecek dönemlerde bu istasyona gerçekleşecek iş akışlarının artacak olması nedeni ile, iş istasyonunun yeni ve diğer iş istasyonlarına daha yakın bir konuma yerleştirilmesi beklenmektedir.

Ayrılmış olan ürün gruplarına göre kalite ayrım sonrası operasyonlar bölümündeki üretim akışları Şekil 5.2.'de gösterilmiştir. Burada gri renkli olan operasyonlar her iki ürün grubu için ortak iken, beyaz renkli olan operasyonlar sadece ilgili ürün grubu için gerçekleştirilmektedir. Beyaz renk ile gösterilen ve ismi aynı olan operasyonlar ise, her bir ürün grubu için farklı özellik gösterdiği için ayrılmaktadır. Her iş istasyonunda belirtilen işlem yapılmakla birlikte, çalışmada operasyonlar arası öncelik ilişkisi göz ardı edilmiştir.



Şekil 5.2. *Kalite ayrım sonrası operasyonlar bölümü ürün akış şemaları*

Çalışma için, öncelikle ürünlerin talep tahminleri ve üretim planlama kararları incelenmiştir. Ürün gruplarının işletmedeki üretim adetlerindeki değişiklikler göz önüne alındığında, tesis planlamasında değişiklik yapılması gerektiği görülmüştür. Gelecek

dönemlerde üretim hacmi artan A tipi ürün için ekipman kapasitesinin yetersiz gelmeye başlaması nedeni ile A tipi ürün grubunun, B tipi ürün grubu için kullanılan makine kapasitesinden faydalanması gerekmektedir. Bu durumun bir sonucu olarak, mevcut durumda bazı ekipmanlar için ürüne göre yerleşim yapılmış olduğundan taşıma mesafesinin uzaması, bunun sonucunda da taşıma maliyetinin artmasının kaçınılmaz olduğu görülmüştür.

Problem, ürün tipleri arasındaki geçişler göz önüne alınarak üç dönem olarak ele alınmış olup, bir dönem bölümdeki bir yıllık üretimi temsil etmektedir. İki ürün grubunun yıllık toplam talep miktarları göz önünde bulundurularak her bir dönem için makineler ve ekipmanlar arası malzeme akış miktarları belirlenmiştir. Üretim sırasında yeniden işleme yapılmadığı varsayılmıştır.

Mevcuttaki yerleşim hiç değiştirilmeden gelecek üç dönem boyunca da üretim faaliyetlerine bu yerleşim planı ile devam edildiği durum için, her bir döneme ait taşıma maliyetleri ve toplam taşıma maliyeti hesaplanmıştır. Maliyetler Tablo 5.1.'de görülmektedir. Hesaplama manuel olarak excelde gerçekleştirilmiştir. Her bir dönem için akış miktarları talep tahminlerinden alınmıştır. Mevcut durumdaki yerleşimde iş istasyonları arası mesafeler ölçülmüştür. Akış miktarları ve mesafeler dönemler için ayrı ayrı çarpılarak dönem bazlı toplam maliyet elde edilmiştir. Ardından 3 döneme ait maliyetler toplanarak, toplam taşıma maliyeti hesaplanmıştır.

Tablo 5.1. *Bölümün mevcut durum taşıma maliyeti*

Dönem	Taşıma Maliyeti	Toplam Maliyet
1	46.275.000	201.325.000
2	49.650.000	
3	105.400.000	

Tablo 5.1. incelendiğinde; ikinci dönem taşıma maliyetlerinin birinci döneme göre %7, üçüncü dönem taşıma maliyetlerinin ise ikinci döneme göre %112 arttığı görülmektedir. Birinci dönem ile üçüncü dönem A ve B tipi ürünlerin üretim hacimleri oranı göz önüne alındığında; birinci dönemde A tipi ürünün üretim hacmi %20, B tipi ürünün üretim hacmi %80 iken, üçüncü dönemde bu durum neredeyse tam tersine dönmektedir.

Birinci dönemde yerleşim planı, halihazırda üretim hacmi yüksek olan B tipi ürüne göre düzenlenmiş olduğundan taşıma maliyeti 46.275.000 olarak hesaplanmıştır. Üçüncü

dönemde ise üretim hacminin büyük bir bölümünü A tipi ürünün oluşturması ve yerleşim planının sabit kalması sonucu A tipi ürünün istasyonlar arası kat edeceği mesafe B tipi ürüne göre daha fazla olduğundan, taşıma maliyeti iki katını da aşarak 105.400.000 olarak hesaplanmıştır. Hacimce fazla üretilen ürünün, az üretilen ürüne göre planlanmış yerleşimi kullanması ve daha fazla mesafe kat etmesi bu sonucu doğurmaktadır.

Makine ve ekipmanların birbirleri arasındaki uzaklık her bir dönem için iş istasyonlarının dizilimine göre değişiklik gösterecektir. Malzeme akış maliyeti her bir ürün grubu için eşit olup, bir ürünün birim taşıma maliyeti 1 TL olarak belirlenmiştir.

Ekipmanların yer değiştirme maliyetleri ise, yer değiştirme sürecinde yaşanacak olan üretim kaybı, işçilik saati, ekipman taşıma ve montajı için gerekli giderler göz önüne alınarak tüm ekipmanlar için ortalama bir değer olarak belirlenmiştir.

Mevcut tesis yerleşim planı EK-2'de gösterilmektedir. Mevcut durum makine ve ekipman yerleşimi incelendiğinde, ürüne göre ve prosese göre yerleşimin bir arada kullanıldığı, ancak ekipmanlar arasında gereksiz boşluklar bulunduğu gözlemlenmiştir. Tesisin kurulduğu zamanlardaki kapasite ihtiyacının yeterli gelmesi ve alanın kısıtlı olmaması makine ve ekipmanların rahat ve dağınık bir şekilde yerleştirilmesine neden olmuştur. Mevcut işletmenin kapasite ihtiyaçları ve yeni tesis yatırımı yapıldığı göz önünde bulundurulduğunda, gelecekte yine tesis ve alan ihtiyacı ortaya çıkabileceğini düşündürmektedir. Yeni tesis inşa edilmesinin getireceği maliyet nedeni ile, mevcut alanın optimum şekilde kullanılması da önem kazanmaktadır.

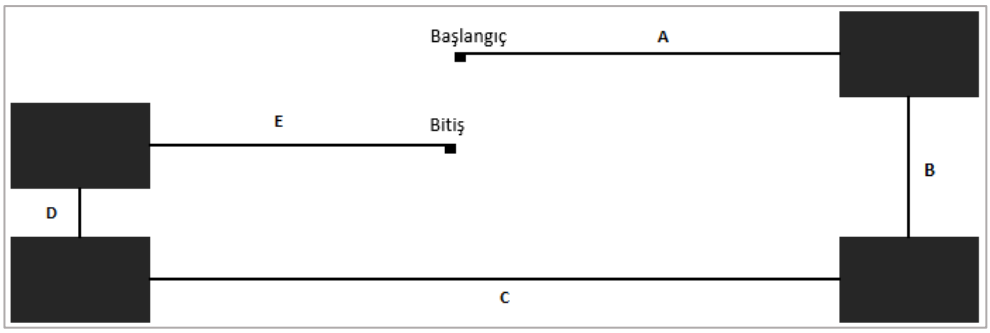
Bölümde üretilen ürünler, iki gruba ayrılmış olsa da kendi içlerinde tasarım ve özellik farkları bulunmaktadır ve çok çeşitlidirler. Bu nedenle hattan çıkan ürünler paketleme ya da sevkiyat bölümüne gitmeden önce paletler üzerinde özelliklerine göre gruplandırılmakta ve paletlerin palet standartlarına uygun bir şekilde dolması beklenmektedir. Ürün çeşitliliğine ek olarak, bazı müşterilerin istedikleri özel palet şekilleri nedeni ile de farklı paletler alanda bekleme yapmaktadır. İş istasyonlarına en yakın olacak şekilde geniş bir alan bu gruplama işi için gerekmektedir.

Ürünlerin akışları incelendiğinde bazı iş istasyonlarına her iki ürün grubunun da akışı olduğu, bazılarına ise sadece tek tip ürün grubunun akışı olduğu görülmektedir. Bu nedenle aralarındaki akış miktarının fazla olduğu istasyonların birbirine yakın olması, paketleme için bekleyecek paletler için gereken alan ve bölümün düzensiz şekilli olduğu düşünüldüğünde U tipi yerleşime benzeyen bir yerleşim şekli belirlenmesine ve iş istasyonlarının bu yerleşim şeklinin orta noktasından geçen bir hat üzerine ağırlık

merkezleri gelecek şekilde yan yana dizilmesine karar verilmiştir. Bunun ilk sebebi; iş istasyonları arası malzeme akışlarının, iş istasyonlarının ağırlık merkezleri arasında gerçekleşiyor olmasıdır. Bir diğer sebebi ise ekipmanlarda bulunan bağlantı noktalarının standart hale getirilerek, güç kaynağı, toz emişi ve atık su giderlerinin herhangi bir tadilat gerektirmeden her noktaya bağlanabilmesi ihtiyacıdır. Bunun sonucunda da iş istasyonu yer değişikliğinin daha kolay ve az maliyet ile gerçekleştirilmesi sağlanacaktır.

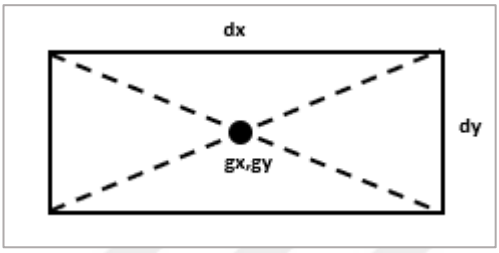
Belirlenen yerleşim şeklinin ana hatları ile gösterimi Şekil 5.3.'teki gibidir. Tüm ürünler başlangıç noktası olarak gösterilen noktadan kalite ayrım sonrası operasyonlar bölümüne giriş yapmaktadır. Bu noktadan bölüme giriş yapan ürünler, ürün grubunun üretim akışına göre sırasıyla ilgili istasyonlara uğramaktadır. Kalite ayrım bölümünden çıkıp, kalite ayrım sonrası operasyonlar bölümüne gelen ürün akışının başlangıç noktası ve bölümün şekli düşünüldüğünde belirlenen U şekilli yerleşimin bir kenarının diğerinden kısa olmasına ve ihtiyaç halinde yerleşimin sonunun, başı ile birleşmeyecek şekilde U'nun içine doğru kıvrılmasına olanak verecek şekilde planlanmıştır. Ayrıca ürünlerin ağırlıkları göz önüne alındığında, yeni yerleşim düzenlemesi yapılırken bir ergonomik ürün taşıma sisteminin de kurulabileceği gündeme gelmiştir.

Yerleşimin köşelerinde dikdörtgenler ile gösterilen yerler ise malzeme stoklamak için kullanılacak olup, bu alanlara istasyon ataması yapılmayacaktır. Stokta, montaj için gereken parçalar, yüzey kaplama malzemeleri ve yedek parçalar gibi malzemeler bulundurulacaktır. Bunun amacı iş istasyonlarında gereksiz malzeme bulundurulmasını engellemek ve bu yardımcı malzemelerin iş istasyonlarına tedarik sürecinin hızlı olmasını sağlamaktır. Maliyet hesaplaması yapılırken, stokta bulunan ürünler için gerçekleşen akışların ihmal edileceği varsayılmıştır.



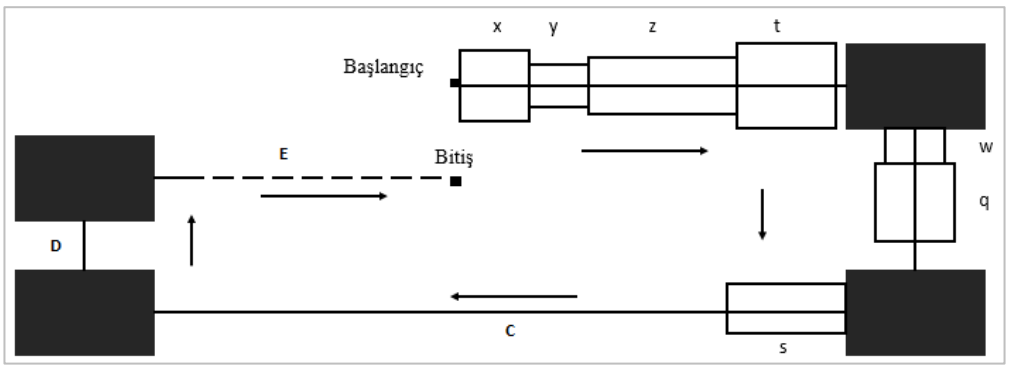
Şekil 5.3. *Yerleşimin görüntüsü*

Makine, ekipman ve çalışanların çalışma anında ihtiyaç duyacağı alanlar da göz önüne alınarak her bir iş istasyonun alan ve kenar uzunlukları belirlenmiş, iş istasyonlarının kenar uzunlukları dx ve dy olarak tanımlanmıştır. Şekil 5.4.'de bir iş istasyonunun dx ve dy uzunlukları gösterilmektedir. İş istasyonlarının dx kenarı her zaman x eksenine, dy kenarı ise y eksenine paralel olarak konumlanacaktır.



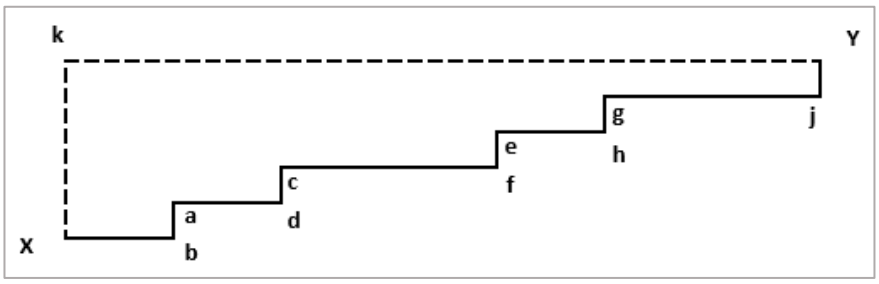
Şekil 5.4. *Bir iş istasyonunun kenar uzunlukları ve ağırlık merkezi*

Birinci sıraya gelecek olan iş istasyonunun bir kenarı başlangıç noktasına gelecek şekilde ağırlık merkezi hat üzerinde konumlanmaktadır. Ardından gelen iş istasyonunun başlangıç noktası ise önceki iş istasyonunun bitiş noktası olacak şekilde yerleştirilmektedir. Stok alanlarına iş istasyonu atanmayacağı için Şekil 5.3.'teki hattın A bölümü tamamlandıktan sonra, ilk stok alanı başlangıç noktası olarak düşünülerek B bölümü için iş istasyonu ataması başlamaktadır. Atama işlemi saat yönünde gerçekleştirilmektedir. Şekil 5.5.'te sırası ile x, y, z, t, w, q ve s istasyonlarının ne şekilde atandığı görülmektedir. Burada t iş istasyonu ile stok alanı arasında bir boşluk görülmektedir. Atama sırası gelen w ve s iş istasyonlarının kenar uzunlukları t ve q iş istasyonları ile stok alanları arasında kalan boşluğa sığmadığı için bu kısım boş bırakılmak zorundadır.



Şekil 5.5. *İstasyon dizilimleri*

Bu şekilde A, B, C, D ve E bölümlerine sırası ile tüm istasyonlar yerleşene kadar atamalar yapılıp, ardından istasyonlar arası dik doğrusal mesafeler hesaplanacaktır. Dik doğrusal mesafenin hesaplanması için bir örnek Şekil 5.6.'da gösterilmektedir.



Şekil 5.6. *Dik-doğrusal uzunluk*

Buradaki dik-doğrusal mesafeler denklem (5.1) yardımı ile hesaplanmaktadır.

$$|Xk| + |kY| =$$

$$|Xb| + |ba| + |ad| + |dc| + |cf| + |fe| + |eh| + |hg| + gj| + |jY| \quad (5.1)$$

Ayrıca, i ve j olarak tanımlanan iki iş istasyonu arasındaki mesafe hesaplanırken, (5.2) numaralı denklem kullanılmaktadır.

$$|gx_i - gx_j| + |gy_i - gy_j| \quad (5.2)$$

Özetlemek gerekir ise, öncelikle problemin mevcut durumu analiz edilmiş ve ortaya konmuştur. Buna göre, 27 iş istasyonlu ve 3 dönemli problemin çözümü için aşağıdaki varsayımlar kabul edilmiştir:

- Bölümler yerleşim üzerine dizilirken her sıraya yerleştirilebilmektedir.
- Yeniden işleme göz ardı edilmiştir.
- Her iki ürün grubu için 1 birim taşıma maliyeti 1 TL olarak belirlenmiştir.
- Ekipmanların yer değiştirme maliyeti her bir ekipman için eşit olarak belirlenmiştir ve yıllar arasında değişmeyeceği varsayılmıştır.
- Belirlenen stok alanlarından iş istasyonlarına malzeme tedarik için gerçekleştirilen akışlar göz ardı edilmiştir.
- Planlanan faz geçişleri, dolayısıyla ürünlerin üretim miktarlarındaki değişikliklerin plana uygun gerçekleşeceği varsayılmıştır.

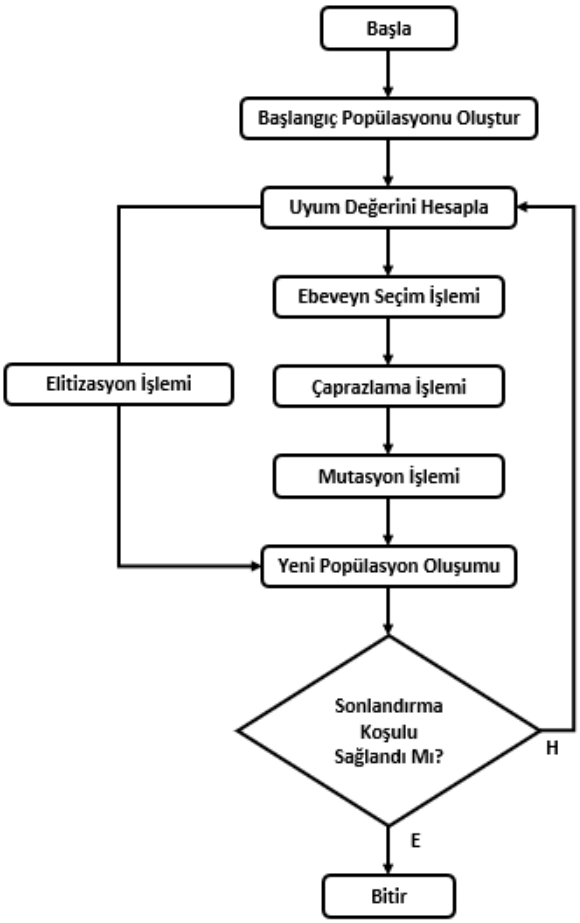
- Kalite ayrım sonrası operasyonlar departmanına ürünler kalite ayrım departmanından gelmektedir. Kalite ayrım departmanı, tünel fırınlardan çıkan ürünleri direk olarak kontrol masalarına yüklemektedir. Tünel fırınların yeri kesinlikle değişemediği için başlangıç noktası sabittir.
- Köşelerde bulunan siyah ile belirtilmiş alanlarda havalandırma ekipmanları, yük taşıma cihazı kolonları bulunduğundan bu noktaların yerleri değiştirilememekte ve istasyon ataması yapılamamaktadır. Bu noktalarda kullanılabilir alanlar yardımcı malzeme vb stok alanı olarak kullanılacaktır.
- İş istasyonları, kullanılacak olan taşıma sisteminin hareketi ve ortak altyapı hattı nedeni ile tek sıra olarak ve yan yana dizilmek zorundadır. İki iş istasyonunun birbirinin önüne ya da arkasına gelmesi durumunda ürünün iş istasyonuna bu sistem ile taşınması ve ortak hattı kullanımı imkânsız olacağından tüm istasyonlar tek sıra olarak konumlanacaktır.
- U benzeri hat planlanıp orta kısmın boş bırakılmasının sebebi ise ürünlerin palete yüklenmesi işleminin bu alanda yapılacak olması ve bu esnada taşıma sisteminden de faydalanılacak olmasıdır.

5.3. Problemin Çözüm Yöntemi ve Programın Tanıtılması

Bu bölümde 3 dönemli DTYP'nin çözümü ve 3 dönem için toplam en düşük maliyetli yerleşimin önerilmesi için GA kullanılarak hazırlanan ve C# ile programlanan algoritma anlatılacaktır. Geliştirilen GA'da, probleme özgü olan U tipi benzeri yerleşim şekli üzerine, eşit olmayan alanlı iş istasyonlarının atanarak çözüm elde edilmesi sağlanmıştır. Çalışmanın sonucunda işletme için en düşük maliyetli yerleşim planının önerilmesi amaçlanmıştır.

5.3.1. Problemin çözümü için geliştirilen algoritma ve özellikleri

Probleminin çözümü için klasik bir GA kullanılmıştır. Algoritmanın adımları Şekil 5.7.'deki gibidir. Burada bulunan uyum fonksiyonunun hesaplanması için bir model değil probleme özgü ayrı bir algoritma hazırlanarak bu algoritma kullanılmıştır.



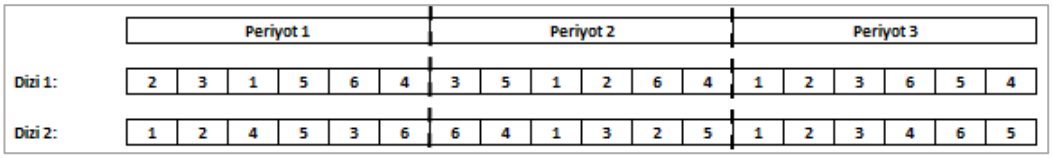
Şekil 5.7. *Problemin algoritması*

Problem bir TYP olduğundan, dizi gösteriminin permütasyon yöntemi ile gerçekleştirilmesine karar verilmiştir. Bu yöntem incelendiğinde, her bir iş istasyonu bir gene karşılık gelmektedir. Bu iş istasyonlarına numara verilmiş olup, ilgili iş istasyonu dizi üzerinde bu numara ile gösterilmektedir. İş istasyonuna ait numara, dizi üzerinde kaçıncı sırada bulunuyor ise, iş istasyonu yerleşimde bu sırada yer alıyor demektir.

Problem 3 dönemli olduğu için, bir dizi 3 parçadan oluşmaktadır. Birinci parça t_1, ikinci parça t_2 ve üçüncü parça t_3 dönemine karşılık gelmektedir. İş istasyonları her bir dönem için dizi üzerinde sıralanmaktadır. Her bir dönem için iş istasyonunun bulunduğu sıra aynı kalabildiği gibi, farklı bir sırada da yer alabilmektedir. Tek dönemli TYP'lerde, n tane iş istasyonu bulunan bir tesis için n tane geni olan dizi oluşturulmaktadır. Problem birden fazla yani m tane dönemden oluştuğunda ise, n*m tane gen dizi üzerine kodlanmaktadır.

Şekil 5.8.'de 3 dönemli ve 6 istasyonu olan örnek dizi gösterimi bulunmaktadır. Burada $dizi_1$'de t_1 döneminde 2 numaralı iş istasyonunun ilk sırada, 3 numaralı iş

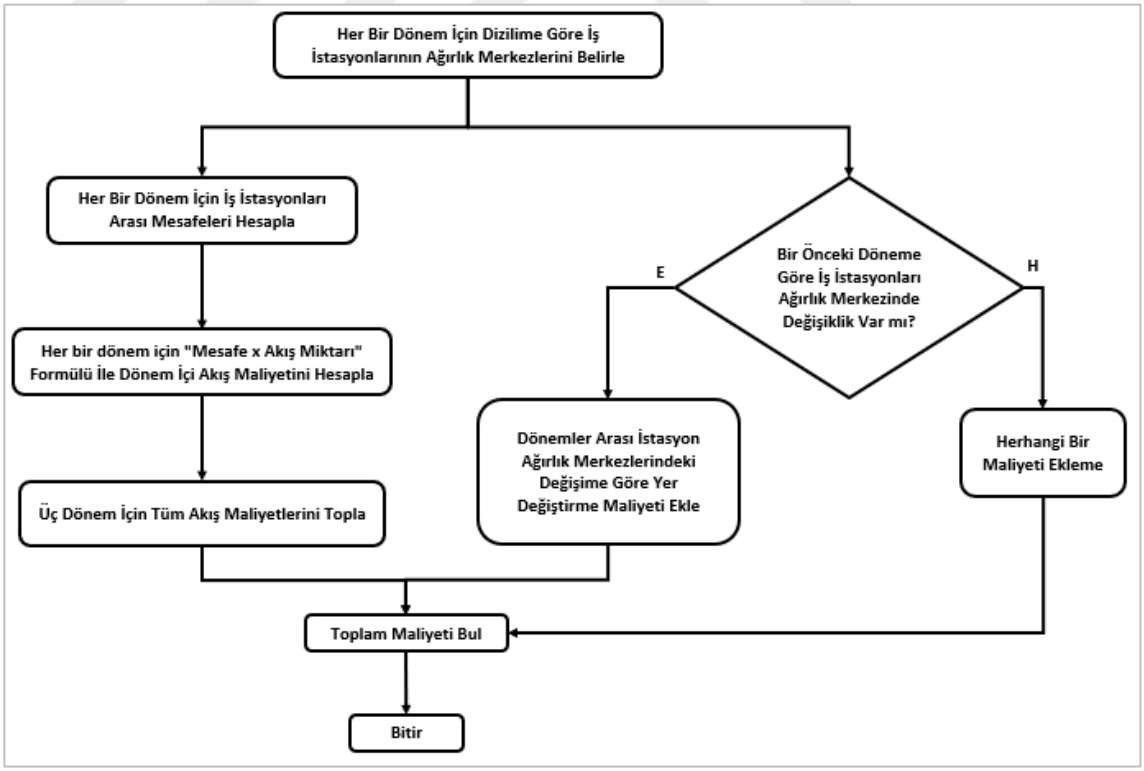
istasyonunun ise belirlenen yerleşimin ikinci sırasında yer aldığı görülmektedir. Aynı şekilde $dizi_2$ için t_1 döneminde ilk sırada 1 numaralı iş istasyonunun, ikinci sırada ise 2 numaralı iş istasyonunun bulunduğu görülmektedir. t_2 dönemi incelendiğinde ise $dizi_1$ için ilk sırada 3 numaralı iş istasyonunun, son sırada ise 4 numaralı iş istasyonunun bulunduğu görülmektedir.



Şekil 5.8. *Genlerin diziye kodlanması*

Gerçek problemde ise 3 dönem ve 27 iş istasyonundan oluşan bir dizi kodlaması yapılmış, toplamda 81 adet gen bir dizi üzerine kodlanmıştır.

DTYP'nin KAP'larda olduğu gibi, bu problemde de amaç taşıma maliyetinin en küçüklenmesidir. Uyum fonksiyonunu algoritması ise Şekil 5.9.'da gösterilmektedir.



Şekil 5.9. *Uyum fonksiyonu algoritması*

Uyum fonksiyonu algoritmasında öncelikle, her bir dönem için belirlenecek istasyon dizilimine göre istasyonlar U tipi yerleşime benzeyen yerleşim şekli üzerine yerleştirilmekte ve iş istasyonunun ağırlık noktası koordinatları yardımı ile istasyonlar arası mesafe hesaplanmaktadır. Hesaplanan mesafeler, ilgili dönem için iki iş istasyonu arasındaki akış miktarları ile çarpılarak taşıma maliyeti hesaplanmaktadır. Burada hesaplama model üzerinden değil, bir algoritma ile gerçekleştirilmektedir.

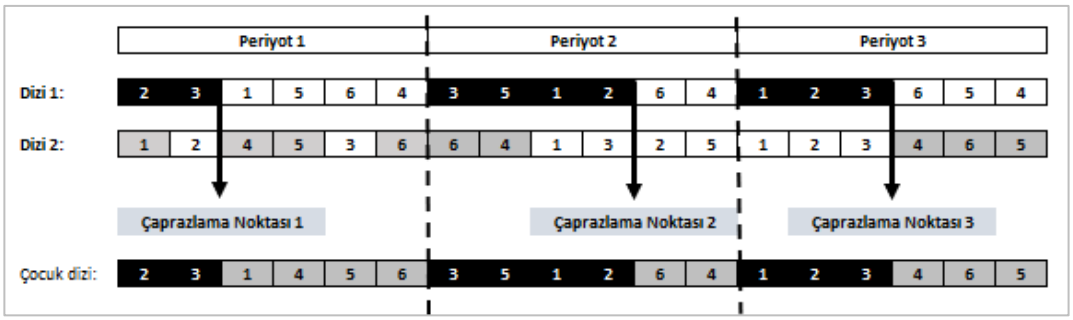
Buna ek olarak, ilk döneme göre ikinci dönemde ya da ikinci döneme göre üçüncü dönemde yer değiştirmiş iş istasyonu var ise, her bir iş istasyonunun yer değiştirme maliyeti de taşıma maliyetine eklenerek toplam taşıma maliyeti elde edilmektedir. Yani her bir nesilde elde edilen dizilerin uygunluk değeri; toplam taşıma maliyeti ve iş istasyonlarının yer değiştirme maliyetleri toplanarak bulunmaktadır. Bu değerin en küçüklenmesi amaçlanmaktadır.

Başlangıç popülasyonu oluşturulurken literatürle çözülen problemler incelenmiş, iş istasyonu sayısı ve çözüm süresi göz önünde bulundurularak, popülasyonların hızlı hesaplanması için 25 adetlik bir popülasyon kullanılmasına karar verilmiştir. Literatürde yaygın olarak örnekleri ile karşılaşılan bir yöntem olarak rastgele popülasyon oluşturulması tercih edilmiştir. Popülasyon üretimi yapılırken bir kısmının iyi sonuçlar verecek diziler olmasına önem verilmiştir. Bu nedenle GA programı hazırlanmadan önce, çeşitli dizilerin girilerek toplam taşıma maliyetinin hesaplanacağı bir program hazırlanmıştır. Dizilerin bir kısmı bu programda denenerek popülasyona eklenmiştir.

Bir popülasyonda bulunan en iyi çözümü kaybetmemek ve bir sonraki popülasyona aktarılmasını garanti altına almak için elitizm fonksiyonu kullanılmıştır. Elitizasyon için α çözüm sayısı $\alpha \leq 0.1*25$ denklemine göre 2 olarak belirlenmiştir.

Çaprazlama operatörü uygulanacak olan bireylerin seçim yöntemi için sıralı seçim mekanizması kullanılmıştır. Oransal seçilim ve rulet tekerleği seçilim yöntemlerinde dizilerin seçilme olasılıkları, ilgili dizilerin uygunluk değerleri ile popülasyonun toplam uygunluk değerine oranlanması ile bulunmaktadır. Yüksek uygunluklu dizilerin bir süre sonra seçilme olasılıkları 1'e yaklaşabileceğinden, diğer dizilerin bir sonraki popülasyona aktarılma olasılığını ortadan kaldırabilmektedir. Bu da yerel optimumlarda yakınsama yaşanmasına sebep olabilmektedir. Bu nedenle sıralı seçim mekanizması tercih edilmiştir.

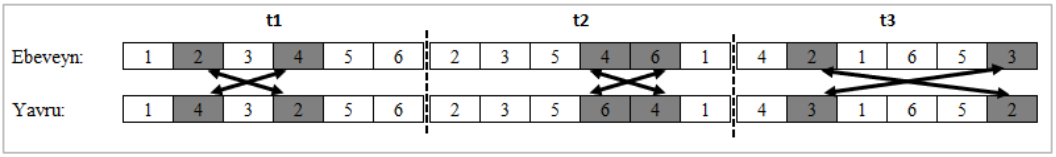
Çaprazlamanın ne şekilde gerçekleştirildiği Şekil 5.10.'da görsel olarak ifade edilmektedir.



Şekil 5.10. *Çaprazlama operatörü*

Çaprazlama için oluşturulan eşleştirme havuzundan iki dizi rassal olarak ebeveyn seçilir. Seçim işleminin ardından, her bir dönem için rassal olarak bir çaprazlama noktası belirlenir. Belirlenen bu noktanın sağ tarafında kalan genler birinci ebeveyn diziden yavru diziye kopyalanır. Geriye kalan atanmamış genler ise ikinci ebeveyn dizide dizili oldukları sıra ile yavru diziye aktarılmaktadır. Örneğin, t_1 dönemi için çaprazlama noktası 2 ve 3. sıra arasında belirlenmiştir. Çaprazlama noktasının sol tarafında kalan birinci dizinin ilk iki geni 2 ve 3 numaralı genler, yine aynı şekilde 1 ve 2. sırada olmak üzere yavru diziye kopyalanır. Ataması yapılmamış olan 1, 5, 6 ve 4 genleri ise, ikinci dizide bulunduğu sıra ile 1, 4, 5, 6 olarak yavru diziye kopyalanır. Aynı işlemler t_2 ve t_3 dönemleri için de gerçekleştirilir ve sonucunda yeni birey oluşturulmaktadır.

Mutasyon genetik operatörü, algoritmanın yakınsamasını önlemek için eklenmiştir. Gerçekleştirilen mutasyon işleminde girilen mutasyon olasılığı oranına göre işlem gerçekleştirilmektedir. Mutasyona uğrayacak olan dizi üzerindeki her bir dönem için öncelikle rassal olarak iki gen, yani iş istasyonu seçilmektedir. Seçilen bu genler her bir dönem kendi içerisinde olmak üzere karşılıklı olarak yer değiştirmektedir. Yani bir dizi üzerinde üç ayrı dönem için "karşılıklı yer değiştirme" mutasyonu uygulanmaktadır. Mutasyonun gerçekleşmesi için diziye atanan rassal sayının, belirlenen mutasyon oranından küçük olması gerekmektedir. Gerçekleşen mutasyon işlemi Şekil 5.11.'de gösterilmektedir.



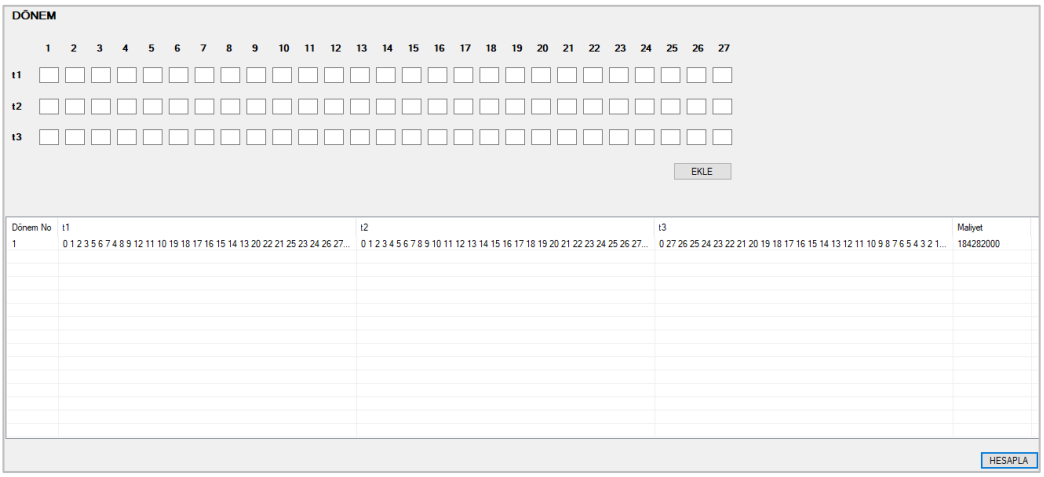
Şekil 5.11. *Mutasyon operatörü*

Nesil sayısına kullanıcı tarafından karar verilmektedir. Problemin en iyi çözümünün elde edilebilmesi için diğer parametreler göz önünde bulundurularak nesil sayısı denenerek belirlenmektedir. Parametrelerde değişiklikler yapıldığında en iyi sonucu verecek olan nesil sayısı program yardımı ile denenerek bulunmaktadır.

5.3.2. Problemin çözümü için geliştirilen program

DTYP'nin çözümü için C# programı ile öncelikle uyum fonksiyonu için bir program hazırlanmıştır. Programa her bir dönem için sırası ile istasyon girişleri 1 numaralı kutucuktan başlayıp, 27 numaralı kutucuğa kadar gerçekleştirilmektedir. Üç dönem için de istasyonları belirten sayıların ilgili alanlara girişi tamamlandıktan sonra bu kutucukların hemen altında bulunan "Ekle" butonuna basılarak, taşıma maliyetinin hesaplanmasını istediğimiz dizi oluşturulmaktadır.

Maliyet hesaplaması yapmak için art arda birden fazla dizi oluşturulabilmektedir. Ardından oluşturulan bu dizilerin içerisinden maliyetini hesaplamak istediğimiz dizinin üzerine tıkladıktan, yani dizinin bulunduğu satırı seçtikten sonra, bu bölümün altında bulunan "Hesapla" butonuna basılmaktadır. Şekil 5.12.'de uyum fonksiyonu programının arayüzü görülmektedir.



Şekil 5.12. *Uyum fonksiyonu programı arayüzü*

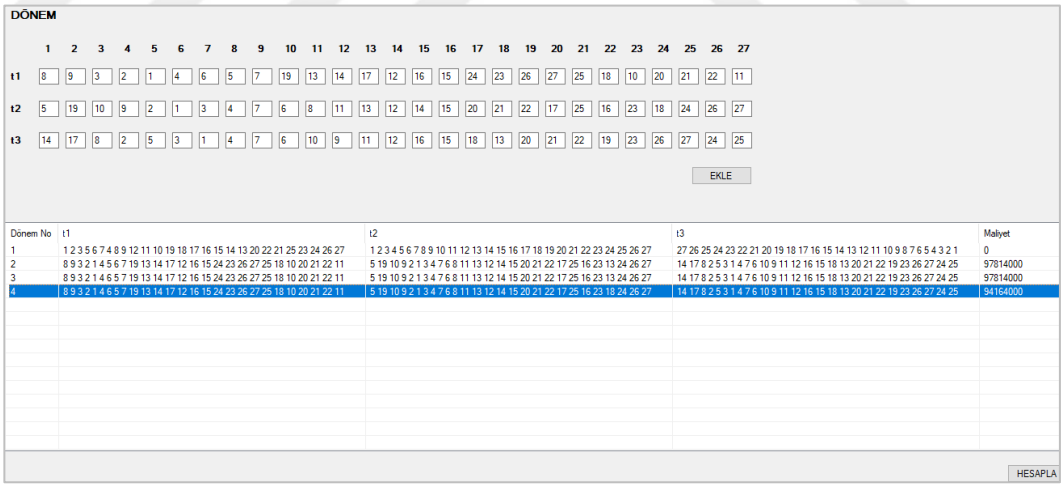
Hesapla butonuna basıldıktan sonra program çalışmaktadır. Program çalışırken, uyum fonksiyonu bölümünde anlatılmış olan algoritmayı uygulayarak girişi yapılmış olan dizinin maliyet hesabını yapmaktadır. Hesaplama işlemi bittikten sonra ise, ilgili dizinin bulunduğu satırın karşısındaki sütuna sonucu yazmaktadır.

İlgili iş istasyonlarının kenar uzunlukları ve her bir dönem için gerçekleşen akışlar program içerisinde tanımlanmıştır. Bir iş istasyonu eklemek/çıkartmak, akış değişikliği ya da iş istasyonunun boyutlarını değiştirmek için programın yazılım kısmında değişiklik yapılması gerekmektedir.

Bu program uyum fonksiyonunun doğru çalışıp çalışmadığını kontrol etmek amacı ile hazırlanmıştır. Programda hesaplatılan 5 farklı dizinin toplam taşıma maliyetinin sonucu, araştırmacı tarafından hesaplanarak doğrulanmıştır.

Programın uyum fonksiyonu kısmı hazırlandığında ilk olarak sadece iş istasyonları arasında gerçekleşen bölüm içerisindeki akışlar göz önüne alınmıştır. Ancak işletmedeki özel durum göz önüne alındığında ürünlerin bu bölüme girişlerinin gerçekleşeceği noktanın değiştirilme olanağının bulunmadığı görülmüştür. Bu nedenle akışların, ürünlerin bölüme girdikleri noktanın da dikkate alınarak hesaplanması gerekmektedir.

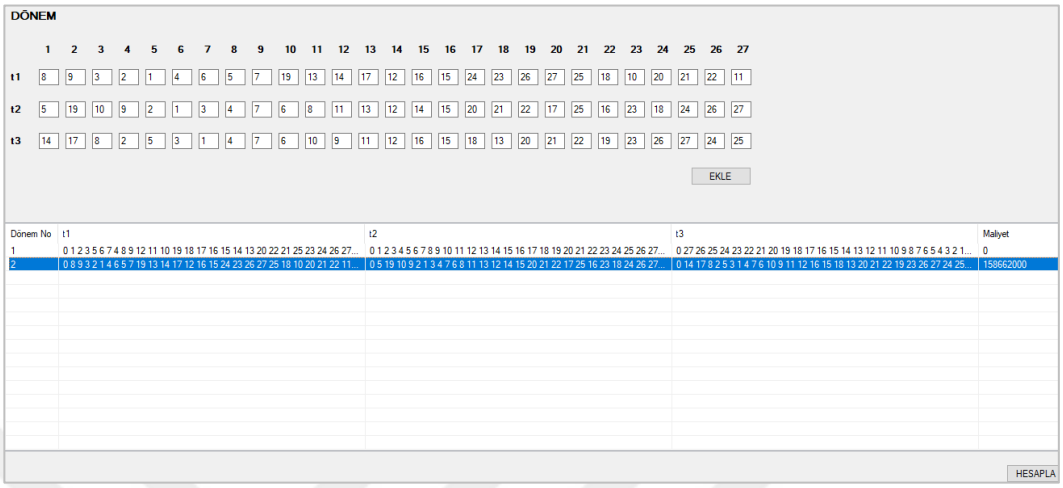
Program bu durum göz önünde bulundurularak tekrar düzenlenmiştir. Şekil 5.13.'te ürünlerin bölüme giriş noktalarından birinci iş istasyonuna olan akışlar göz önünde bulundurulmadığı durum için bir yerleşim planının taşıma maliyeti hesaplanmıştır. Hesaplamanın sonucunda 94 milyon gibi bir taşıma maliyeti elde edilmektedir.



Şekil 5.13. *Bölüme gelen akış dikkate alınmadığında taşınma maliyeti*

Şekil 5.14.'te ise, aynı yerleşim planı için aynı kısıtlar altında ürünlerin bölüme giriş noktasından itibaren gerçekleşecek taşımalar da göz önüne alındığında taşıma maliyeti 158 milyon olarak bulunmaktadır. Yani burada görülmektedir ki, ürünlerin bölüme giriş noktasından, ilk iş istasyonuna olan taşımalar göz ardı edildiğinde toplam

taşıma maliyeti 94 milyon olarak hesaplansa da bu taşımalar göz önüne alındığında gerçek maliyetin 158 milyon olduğu görülmektedir.

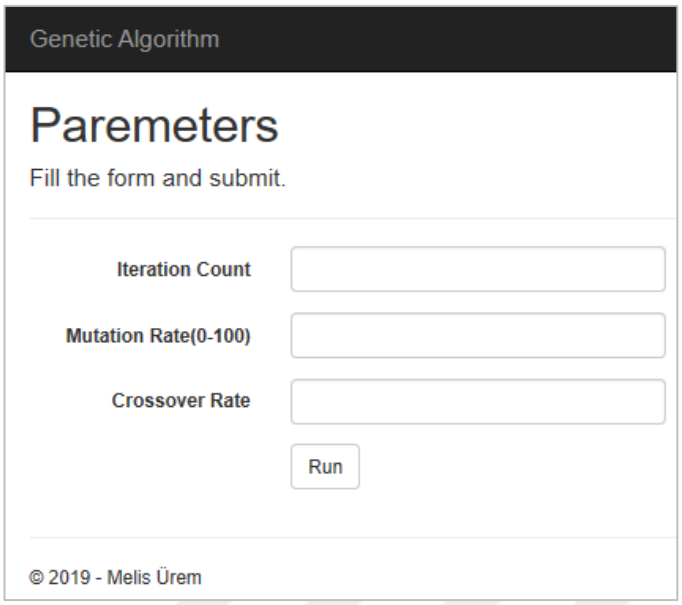


Şekil 5.14. *Bölüme gelen akış dikkate alındığında taşınma maliyeti*

Bu da gerçek maliyetin ilk hesaplanan maliyetten aslında %68 daha fazla olduğunu gözler önüne sermektedir. Bu nedenle, gerçek problemler ile ilgili uygulamalar yapılırken karara etki edebilecek tüm parametrelerin göz önüne alınması çok önemlidir.

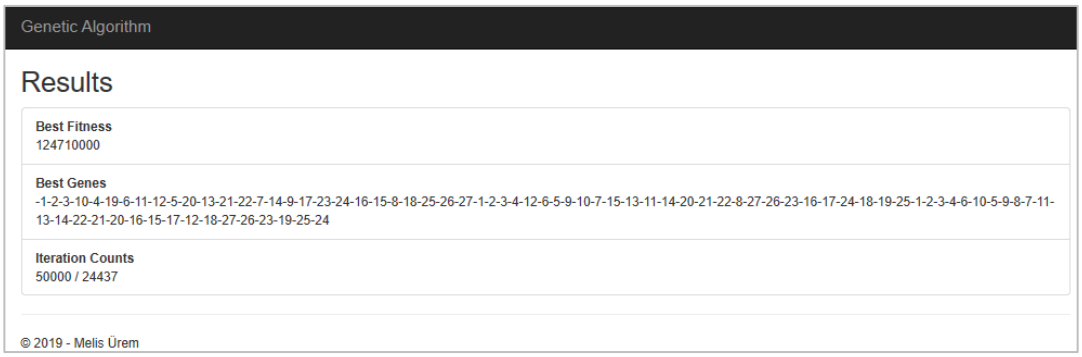
Uyum fonksiyonu programının doğru çalıştığı tespit edildikten sonra, GA ile en iyi sonucun aranacağı program için ikinci aşamaya geçilmiştir. Daha önce belirtilen parametreler doğrultusunda program hazırlanmıştır. Program içerisinde GA'ya tabi tutulan diziler 81 genden oluşmaktadır. Ancak bölüme giriş noktası düşünüldüğünde, istasyona gelene kadar gerçekleşen akış da düşünüldüğünde dizinin her dönemine gen eklemek yerine, giriş noktasının ilk iş istasyonuna olan uzaklığının her defasında hesaplanarak maliyet fonksiyonuna yansıtılması sağlanmıştır. Böylece dizi üzerinde genetik operatörler uygulandığında giriş noktasının yerinin değişmesinin önüne geçilmiştir.

Hazırlanan GA programında çaprazlama oranı, mutasyon oranı ve iterasyon sayısı programa girilerek arama yapılması sağlanmaktadır. Programa giriş ekranı Şekil 5.15.'te görülmektedir. Literatürde bulunan diğer örnekler gibi akışlar programa başka bir dosyadan okutulmamaktadır. Program, probleme özel olarak hazırlandığından, akışlar program içerisinde tanımlanmıştır. Mesafeler ise uyum fonksiyonu algoritması içerisinde her diziye göre hesaplanarak, maliyet hesaplama kısmına dahil olmaktadır.



Şekil 5.15. *Genetik algoritma parametre giriş ekranı*

Programa parametreler girilip "Run" butonuna basıldıktan sonra, çözüm uzayında en iyi uyum fonksiyonuna sahip dizi aranmaya başlanmaktadır. Program çalışmasını tamamladıktan sonra, bulduğu en iyi sonucu, yerleşim sıralamasını ve en iyi sonuca kaçıncı iterasyonda ulaşıldığını göstermektedir. Programın çalışmasını tamamladıktan sonra açılan sonuç arayüzü Şekil 5.16.'da görülmektedir. Program kodu ise Ek-3'te gösterilmektedir.



Şekil 5.16. *Genetik algoritma programı sonuç ekranı*

Bu arayüzde 50000 iterasyon içerisinde en iyi uyum fonksiyonuna sahip dizinin 24437. iterasyonda elde edildiği ve bu dizi için dönemler bazında iş istasyonlarının dizilimi görülmektedir. Bu en iyi dizinin uyum fonksiyonunun sonucu ise 124.710.000 olarak belirtilmektedir.

5.4. İyileştirme Sonrası Önerilen Durum

En ucuz maliyetli iyileştirme önerisini elde etmek için belirli parametreler altında program çalıştırılmıştır. Elde edilen en iyi değer ve bu değerin elde edildiği parametreler Tablo 5.2.'de gösterilmektedir.

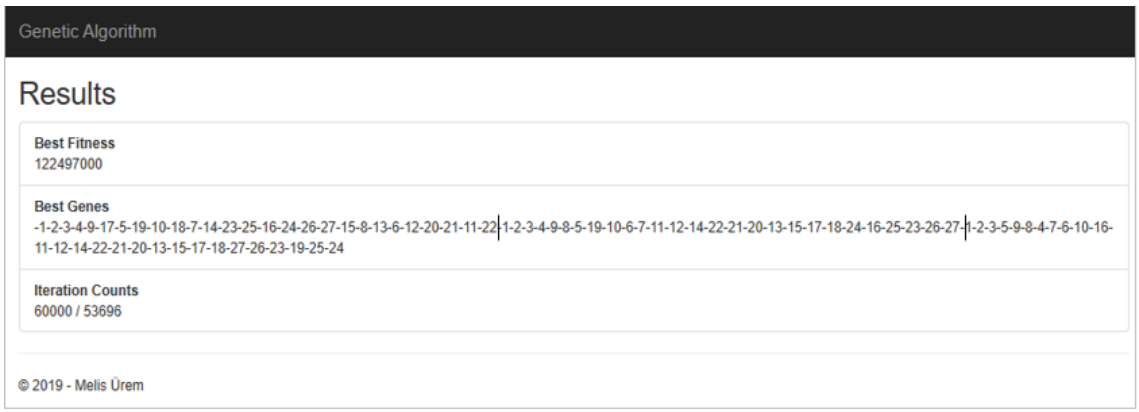
Tablo 5.2. *Elde edilen en iyi değerin parametreleri*

Popülasyon büyüklüğü	**: 25**
Pc	**: 90%**
Pm	**: 2%**
İterasyon sayısı	**: 60000**

Literatürden elde edilen bilgiler ışığında popülasyon büyüklüğünün 25 olmasına karar verilmiştir. Bu popülasyon büyüklüğü ile çeşitliliği sağlamak için pek çok çaprazlama oranı denenmiş, popülasyon büyüklüğünün %85'in altında kaldığı durumlarda sonuçların iyileşme göstermediği görülmüştür. En iyi sonuçlar ise çaprazlama oranının %85-90 olduğu durumlarda elde edilmiştir.

Bunlara ek olarak, mutasyon oranı %0 ya da %1 olarak belirlendiğinde sonuçların çabuk yakınsadığı, çeşitliliğin sürdürülemediği görülmüştür. %2'nin üzerinde mutasyon oranı uygulandığında da yine çabuk yakınsama durumu ile karşılaşılmıştır. İyi genlerin muhtemelen bir sonraki nesillere aktarılamadan bozulduğu düşünülmektedir.

İterasyon sayısı, 10.000 ve katları olarak 160.000'e kadar denenmiştir. İterasyon sayısının 50.000'den küçük olduğunda en iyi diziyi bulmakta yetersiz kaldığı, 70.000'den büyük olduğunda ise herhangi bir iyileştirme elde edilemediği, sadece çözüm süresinin uzadığı görülmüştür. Elde edilen en iyi çözüm ise Şekil 5.17.'de gösterilmektedir.



Şekil 5.17. *Elde edilen en iyi çözüm*

Elde edilen en iyi çözüm incelendiğinde, en iyi dizinin 53.696. iterasyonda elde edildiği görülmektedir. Dönemler bazında en iyi dizideki istasyonların sıralamaları gösterilmekte ve sonucu 122.497.000 olarak belirtilmektedir.

En iyi dizide, her iki ürün tipinin de uğradığı iş istasyonlarının başlangıç noktasına en yakın yere konumlandığı ve dönemler arasında yerlerinin neredeyse değişmediği görülmektedir. Buna ek olarak, sadece B tipi ürünlerin üretiminde kullanılan iş istasyonlarının ilk dönemden son döneme doğru giderek bölüm içerisinde en uzak noktaya doğru konumunun değiştiği görülmektedir. Aynı şekilde A tipi ürünlerin üretiminde kullanılan iş istasyonları ise, ilk dönemden son döneme doğru giderek akışın yoğun olduğu diğer iş istasyonlarına yakın olarak konumlanmaktadır.

Programdan elde edilen en iyi çözümün dönemler bazında dizilimi Tablo 5.3.'te gösterilmektedir. Mevcut durumda taşıma maliyeti 201.325.000, yer değiştirme maliyeti ise 0 idi. Önerilen dizilim sonucunda taşıma maliyeti ve yer değiştirme maliyetlerinin toplamı 122.497.000 olarak hesaplanmıştır. Bu sonuca göre yerleşim yeniden ele alındığında %39 oranında ciddi bir iyileşme sağlandığı görülmektedir.

Tablo 5.3. *Dönemler bazında elde edilen en iyi dizilim*

t1:	1	2	3	4	9	17	5	19	10	18	7	14	23	25	16	24	26	27	15	8	13	6	12	20	21	11	22
t2:	1	2	3	4	9	8	5	19	10	6	7	11	12	14	22	21	20	13	15	17	18	24	16	25	23	26	27
t3:	1	2	3	5	9	8	4	7	6	10	16	11	12	14	22	21	20	13	15	17	18	27	26	23	19	25	24

Elde edilen en iyi yerleşimin üçüncü ve son döneme ait olan yerleşim planı Ek-4'te gösterilmektedir. Mevcut durum yerleşiminde B tipi ürünün üretim hacminin fazla olduğu ve yerleşim planının da ağırlıklı olarak bu ürün referans alınarak oluşturulduğu bilinmekteydi. Üçüncü dönem için oluşan yerleşim planı da incelendiğinde yine üretim hacmi en fazla olan, yani A tipi ürüne göre oluşturulduğu görülmektedir.

Tablo 5.4. *Bölümün iyileştirme sonrası taşıma maliyeti*

Dönem	**Taşıma Maliyeti**	**Yer Değiştirme Maliyeti**	**Toplam Maliyet**
1	35.920.000	-	122.497.000
2	41.974.000	17.000	
3	44.574.000	12.000	

Tablo 5.4.'te ise, dönemler bazında taşıma ve yer değiştirme maliyetleri bulunmaktadır. Birinci dönem incelendiğinde toplam taşıma maliyetinin 35.920.000 olduğu, yer değiştirme maliyetinin ise bulunmadığı görülmektedir. İkinci dönemde taşıma maliyeti yaklaşık %16 bir artış göstererek 41.974.000'ye ulaşmıştır. Buna ek olarak birinci döneme göre yer değiştirmiş olan iş istasyonlarının yer değiştirme maliyetinin de 17.000 olduğu görülmektedir. Üçüncü dönem taşıma maliyeti ise ikinci döneme göre %6 bir artış ile 44.574.000 ve ikinci döneme göre yer değiştiren iş istasyonlarının yer değiştirme maliyeti ise 12.000 olarak belirtilmektedir.

Maliyetlerin dönemsel olarak sebepleri incelendiğinde birinci dönemdeki yerleşimin, üretim hacmi o dönemde en yüksek olan B tipi ürüne göre planlanmış olması nedeni ile düşük olduğu görülmektedir. İkinci dönemde ise A ve B tipi ürünlerin üretim hacimleri neredeyse birbirine eşit olduğundan, herhangi bir ürün tipine göre yerleşim yapılamamakta, her durumda maliyet artışı yaşanmaktadır. Ancak, üçüncü dönemin yerleşimi incelendiğinde, üretim hacmi yüksek olan A tipi ürüne göre planlanmış olduğu görülse de üretilmekte olan ürün adedi önceki dönemlere göre az da olsa fazla olduğundan, bu dönemde de taşıma maliyetlerinde artış olduğu görülmektedir.

Sonuç olarak iyileştirme önerisi sonrasındaki durum mevcut durum ile kıyaslandığında yapılan çalışma sonucunda %39 gibi büyük bir oranda taşıma maliyetlerinde iyileşme gerçekleştiği görülmektedir.

6. SONUÇLAR VE ÖNERİLER

Bu çalışmada birtakım işlevsel kısıtlar ve taşıma kısıtları nedeni ile belirlenmiş olan sabit bir yerleşim üzerinde eşit olmayan alanlı iş istasyonların atamasının yapılması sağlanmıştır. Çalışma, bu noktada diğer DTYP çalışmalarından farklılık göstermekte ve özel problemlerin çözümü konusunda literatüre farklı bir bakış açısı kazandırmaktadır. Literatürde sayıca az bulunan eşit olmayan alanlı iş istasyonlarının DTYP çalışmalarına yeni bir yaklaşım geliştirmiştir. Gerçek hayatta uygulanabilir bir problem seçilerek, probleme özel bir çalışma yapılmıştır. Bu şekilde hazır bir test problemi bulunmadığı için sadece gerçek problem ile çalışılmış, karşılaştırma yapılamamıştır.

Belirlenen bir yerleşim şeklinin orta noktasından geçen bir hat üzerine eşit olmayan alanlı iş istasyonlarının ağırlık merkezlerinin atanması probleminin çözümü için hazırlanan program, probleme özel olarak oluşturulmuş ve bu tip işletmelere özel problemlerin çözümüne bir örnek olmuştur.

Problem, eşit olmayan alanlı iş istasyonlarından oluştuğu için, eşit alanlı iş istasyonlarının bulunduğu problemlere göre daha zor bir yapıdadır. Buna ek olarak iş istasyonlarının ağırlık merkezlerinin sadece bir hat üzerine denk gelecek şekilde yan yana dizilmesinin amaçlanması da problemi zorlaştırmaktadır. Bu nedenle KAP ile modellenmemiş ve çözüm için özel bir uyum fonksiyonu algoritması geliştirilmiştir.

Eşit olmayan alanlı iş istasyonlarının, belirlenen bir alanda, belirli bir sabit yerleşim üzerine yerleştirilmesini amaçlayan problem çözülmüş ve taşıma maliyetlerinde %39 oranında önemli bir iyileştirme gerçekleşmiştir. Dinamikliğin yanı sıra iyileştirme oranının bu denli büyük olmasının bir sebebi de mevcut yerleşimde iş istasyonlarının uygunsuz biçimde birbirinden çok uzakta konumlanmış olmasıdır.

Genellikle literatürdeki TYP'lerde, ürünlerin tesise giriş çıkış noktaları ihmal edilmektedir. Depo gibi akışın yoğun olduğu bölümler genel olarak tesisin merkezine yerleştirilmekte ve bu nedenle çözüm direkt olarak gerçek hayatta uygulanamamaktadır. Bu çalışmada akışların verimli ve düşük maliyetli olarak gerçekleşmesi için sadece bölümdeki iş istasyonları arası akışlar değil, ürünlerin tesise giriş yaptıkları noktalar ve çıkış yapacakları noktalar da göz önünde bulundurulmuştur. Ürünlerin giriş noktası değiştirilememekte olduğundan bu noktanın koordinatı zaten bilinmektedir. İşletme yönetiminin kararı gereği işlem gören ürünler, bölümün ortasındaki alanda paletlere yüklenmektedir. Bunun için ise, ortadaki alanı temsil eden bir nokta, akışın son noktası

olarak belirlenmiştir. Böylece problemin çözümünün gerçek hayatta uygulanabilir ve karar vermeyi etkileyecek sonuçlarının gerçekçi olması amaçlanmıştır.

Bu çalışma sonrasında yapılabilir çalışmalar şu şekilde sıralanabilir;

- Gerçekleştirilen çalışmada dönemler bir yıl olarak belirlenmiş olsa da aylar bazında incelendiğinde talepler yıl içerisinde homojen dağılım göstermeyebilir yani dönemler birbirine eşit sürelere sahip olmayabilir. Birinci dönem bir yıl sürerken, ikinci dönem altı ay, üçüncü dönem ise iki yıl sürebilir. En optimum sonucun elde edilebilmesi için dönemlerin de dinamik olarak belirlenmesi maliyet iyileştirme çalışmalarına büyük oranda katkı sağlayacaktır.
- Probleme özel olarak hazırlanan yapı, benzer üretim süreçlerini içeren işletmelerde uygulanarak sanayide katma değersiz işlerin azaltılarak, işletme veriminin artması sağlanabilir.
- Gelecek çalışmalarda iş istasyonları arasındaki öncelik ilişkilerinin incelenebilir. Bu ilişkiler de göz önünde bulundurularak elde edilecek sonuçların gerçek hayat ile yüksek oranda örtüşmesini sağlayabilir.

KAYNAKÇA

[1] J. A. Tompkins, J. A. White, Y. A. Bozer ve J. M. A. Tanchoco, Facilities planning, John Wiley&Sons Inc., 1996.

[2] Merriam-Webster, «Merriam-Webster,» 05 03 2019. [Çevrimiçi]. Available: https://www.merriam-webster.com/dictionary/dynamic.

[3] H. Erkut ve M. Baskak, Stratejiden uygulamaya tesis tasarımı, İstanbul: İrfan Yayıncılık, 2003.

[4] S. A. Kumar ve N. Suresh, Operations Management, New Delhi: New Age International (P) Ltd. Publishers, 2009.

[5] N. Aras, *ENM411 - Tesis Yerleşimi,* Eskişehir, 2017.

[6] H. E. Şanlı, *An alternative approach to facility layout planning: spiral facility layout planning,* İstanbul, 2010.

[7] L. Chwif, M. R. P. Barretto ve L. A. Moscato, «A solution to the facility layout problem using,» *Computers in Industry,* no. 36, pp. 125-132, 1998.

[8] H. H. Nasab, S. Fereidouni, S. M. T. F. Ghomi ve M. B. Fakhrzad, «Classification of facility layout problems: a review study,» *Internationaj Journal od Advanced Manufacturing Technology,* Ağustos 2017.

[9] M. V. Pillai, I. B. Hunagund ve K. K. Krishnan, «Design of robust layout for dynamic plant layout problems,» *Computers & Industrial Engineering,* cilt 61, no. 3, pp. 813-823, Ekim 2011.

[10] S. Heragu, Facilities Design, Boston, USA: PWS Publishing, 1997.

[11] S. Kulluk ve O. Türkbey, «Tesis yerleşimi problemi için bir genetik algoritma,» %1 içinde *YA/EM'2004 - Yöneylem Araştırması/Endüstri Mühendisliği - XXIV Ulusal Kongresi*, Gaziantep-Adana, 2004.

[12] B. Ulutaş, *Dinamik yerlesim probleminin çözümü için bir klonal seçim algoritması ve uygulamaları,* Eskişehir, 2008.

[13] S. P. Singh ve R. R. K. Sharma, «A review of different approaches to the facility layout problems,» *Int J Adv Manuf Technol* , p. 425–433, Kasım 2006.

[14] B. Ulutaş ve A. A. İşlier, «Dynamic facility layout problem in footwear industry,» *Journal of Manufacturing Systems,* no. 36, pp. 55-61, 2015.

[15] O. Türkbey ve Ç. Alabaş, «Tesis yerleşimi problemi için bir bulanık-tabu arama yaklaşımı,» *Anadolu Üniversitesi Bilim ve Teknoloji Dergisi,* cilt 3, no. 1, pp. 77-88, 2002.

[16] J. Balakrishnan, R. F. Jacobs ve M. A. Venkataramanan, «Solutions for the constrained dynamic facility layout problem,» *European Journal of Operational Research,* pp. 280-286, 1992.

[17] D. G. Conway ve M. A. Venkataramanan, «Genetic search and the dynamic facility layout problem,» *Computers Operations Research,* cilt 21, no. 8, pp. 955-960, 1994.

[18] T. L. Urban, «Solution procedures for the dynamic facility layout problem,» *Annals of Operations Research,* no. 76, pp. 323-342, 1998.

[19] J. Balakrishnan, C. H. Cheng, D. G. Conway ve C. M. Lau, «A hybrid genetic algorithm for the dynamic plant layout problem,» *International Journal of Production Economics,* no. 86, pp. 107-120, 2003.

[20] A. Baykasoğlu, T. Dereli ve İ. Sabuncu, «An ant colony algorithm for solving budget constrained and unconstrained dynamic facility layout problems,» *Omega The International Journal of Management Science,* no. 34, pp. 385-396, 2006.

[21] R. Şahin, «Dinamik tesis düzenleme problemi için bir tavlama benzetimi sezgiseli,» *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi,* cilt 23, no. 4, pp. 863-870, 2008.

[22] R. Şahin, K. Ertoğral ve O. Türkbey, «A simulated annealing heuristic for the dynamic layout problem with budget constraint,» *Computers & Industrial Engineering,* no. 59, pp. 308-313, 2010.

[23] K. S. N. Ripon, K. Glette ve J. Torresen, «Dynamic facility layout problem under uncertainiyt: a Pareto-optimality based multi-objective evolutionary approach,» *Central European Journal of Computer Science,* cilt 1, no. 4, pp. 375-386, 2011.

[24] G. Moslemipour ve T. S. Lee, «Intelligent design of a dynamic machine layout in uncertain environment of flexible manufacturing systes,» no. 23, pp. 1849-1860, 2012.

[25] M. Mazinani, M. Abedzadeh ve N. Mohebali, «Dynamic facility layout problem based on flexible bay structure and solving by genetic algorithm,» *International Journal of Advanced Manufacturing Technologies,* no. 65, pp. 929-943, 2013.

[26] G. Y. Chen, «A new data structure of solution representation in hybrid ant colony optimization for large dynamic facility layout problems,» *International Journal of Production Economics,* no. 142, pp. 362-371, 2013.

[27] M. Kaveh, V. M. Dalfard ve S. Amiri, «A new intelligent algorithm for dynamic facility layout problem in state of fuzzy constrains,» *Neural Comput. & Applic.,* 2013.

[28] S. Wang, X. Zuo, X. Liu, X. Zhao ve J. Li, «Solving dynamic double row layout problem via combining simulated annealing and mathematical programming,» *Applied Soft Computing,* no. 37, pp. 303-310, 2015.

[29] A. Tayal, A. Gunasekaran, S. P. Singh, R. Dubey ve T. Papadopoulos, «Formulating and solving sustainable stochastic dynamic facility layout problem: a key to sustainable operations,» *Ann Oper Res,* no. 253, pp. 621-655, 2017.

[30] B. Turanoğlu ve G. Akkaya, «The dynamic facility layout problems with closeness rate: a fuzzy decision support system approach,» *Selçuk Üniversitesi Mühendislik Bilimleri ve Teknoloji Dergisi,* cilt 5, no. 3, pp. 300-312, 2017.

[31] C. Duman, *Genetik algoritma ile tesis yerleşimi tasarımı ve bir uygulama,* İstanbul, 2007.

[32] G. G. Emel ve Ç. Taşkın, «Genetik algoritmalar ve uygulama alanları,» *Uludağ Üniversitesi İktisadi ve İdari Bilimler Dergisi,* cilt 21, no. 1, pp. 129-152, 2002.

[33] R. Cheng, M. Gen ve Y. Tsujimura, «A tutorial survey of job-shop scheduling problems using genetic algorithms: I. presentation,» *Computers ind. Engng*, cilt 30, no. 4, pp. 983-997, 1996.

[34] M. H. Satman, Genetik Algoritmalar, İstanbul: Türkmen Kitabevi, 2016.

[35] A. Altay, *Genetik algoritma ve bir uygulama,* İstanbul, 2007.

[36] M. Çunkaş, *Genetik algoritmalar ve uygulamaları,* Konya, 2006.

EK-1. Dinamik tesis yerleşim problemi ile ilgili çalışmalar listesi

No	Yıl	Yazar Adı	Makale İsmi	Çözüm Stratejisi	Departman Sayısı	Dönem Sayısı
1	2011	Pillai V.M., Hunagund I.B., Krishnan K.K.	Dinamik Tesis Yerleşim Problemleri İçin Dayanıklı Yerleşim Düzeni Tasarımı	Tavlama benzetimi	6, 15, 30	5, 10
2	2016	Vitayasak S., Pongcharoen P., Hicks C.	Genetik Algoritma Ya Da Değiştirilmiş Geri İzleme Arama Algoritması Kullanarak Stokastik Talep ile İlgili Stokastik Dinamik Tesis Yerleşimi Sorunlarını Çözmek İçin Bir Araç	Genetik algoritma ve değiştirilmiş geri izleme arama algoritması	10	5
3	2013	Emami S., Nookabadi A.S.	Dinamik Tesis Yerleşimi Problemi İçin Yeni Çok Amaçlı Bir Modelin Yönetilmesi	Genetik algoritma	6	5
4	2013	Drira A., Pierreval H., Gabouj S.	Fazla Mesai Süresinde Artan Bilgi Belirsizliğine Sahip Sağlam Bir Düzen Tasarımı: Bulanık Evrimsel Bir Yaklaşım	Genetik algoritma	3,4,5	5,6,7
5	2009	Mckendall A.R., Hakobyan A.	Eşit Olmayan Alanlı Bölümler ile Dinamik Tesis Problemi İçin Sezgiseller	Tabu arama algoritması ve sınır araştırması		
6	2004	Mckendall A.R., Shang J.	Dinamik Tesis Yerleşim Problemi İçin Hibrit Karınca Sistemleri	Hibrit karınca koloni algoritması		
7	2013	Abedzadeh M., Mazinani M., Moradinasab N., Roghanian E.	Bulanık Çok Amaçlı Dinamik Tesis Yerleşimi Problemini Çözmek İçin Paralel Değişken Komşuluk Araması	Paralel değişken komşuluk araması	4,5	2,3
8	2015	Asl A.D., Wong K.Y.	Eşit Olmayan Alanlı Statik ve Dinamik Tesis Yerleşimi Problemlerini, Değiştirilmiş Parçacık Sürüsü Optimizasyonunu Kullanarak Çözme	Değiştirilmiş parçacık sürüsü optimizasyonu	6,12	6,4
9	2016	Mehdizadeh E., Rahimi V.	Operatör Ataması ve Hücre İçi / Dışı Hücre Düzenleri Dikkate Alınarak Dinamik Hücre Oluşum Problemini Çözmek İçin Entegre Bir Matematiksel Model	Çok amaçlı tavlama benzetimi ve çok amaçlı titreşim sönümlendirme optimizasyonu		
10	2016	Ghosh T., Doloil B., Dan P.K.	Hücresel Üretim Sistemindeki Dinamik Çok Amaçlı Düzen Problemlerinin Çözümünde Yapay Zeka İle Hesaplama Tekniklerinin Uygulanması	Geliştirilmiş genetik algoritma ve tavlama benzetimi sezgiseli	6	5
11	2015	Xu J., Song X.	Geçici İnşaat Tesisleri İçin Bulanık Belirsiz Ortam Altında Eşit Olmayan Alan Bölümleriyle Çok Amaçlı Dinamik Yerleşim Problemi	Parçacık sürüsü optimizasyonu	5	14
12	2015	Ulutaş B., İşlier A.A.	Ayakkabı Sektöründe Dinamik Tesis Yerleşim Problemi	Klonal seçim tabanlı algoritma	54	2,4
13	2015	Wanga S., Zuoa X., Liua X., Zhaoc X., Li J.	Dinamik Çift Sıra Yerleşim Problemini Simüle Edilmiş Tavlama Benzetimi ve Matematiksel Programlamayı Birleştirerek Çözme	Geliştirilmiş tavlama benzetimi		
14	2015	Kumar S.S., Cheng J.C.P	Sıkışık Şantiyeler İçin Yapı Bilgi Modellemesi Tabanlı Otomatik Yerleşim Planlama Çerçevesi	Genetik algoritma		
15	2015	Kulturel-Konak S., Konak A.	Döngüsel Tesis Yerleşimi Problemleri İçin Büyük Ölçekli Hibrit Tavlama Benzetimi Algoritması	Büyük ölçekli tavlama benzetimi algoritması		
16	2014	Bozorgi N., Abedzadeh M., Zeinali M.	Dinamik Tesis Yerleşiminde Sorun Araştırmasının Etkinliği İçin Sezgisel Tabu Arama ve Dinamik Tesis Yerleşiminde Tavlama Benzetimi	Tabu arama algoritması ve veri zarflama analizi	6	5
17	2014	Hosseini S., Khaled A.A., Vadlamani S.	Değişken Komşuluk Problemi İçin Hibrit Emperyalist Rekabet Algoritması	Hibrit emperyalist rekabet algoritması ve tavlama benzetimi		

EK-1. (devam) Dinamik tesis yerleşim problemi ile ilgili çalışmalar listesi

18	2014	Pourvaziri H., Naderib B.	Dinamik Tesis Yerleşimi Problemi İçin Karma Çok Popülasyonlu Genetik Algoritma	Genetik algoritma	5	3
19	2013	Mazinani M., Abedzadeh M., Mohebali N.	Esnek Bölme Yapısına Dayalı Dinamik Tesis Yerleşim Problemi ve Genetik Algoritma ile Çözümü	Genetik algoritma		
20	2014	Kaveh M., Dalfard V.M., Amiri S.	Bulanık Kısıtlamalar Altında Dinamik Tesis Yerleşim Problemi İçin Yeni Bir Akıllı Algoritma	Genetik algoritma ve tavlama benzetimi		
21	2013	Hosseini-Nasab H., Emami L.	Dinamik Tesis Yerleşim Problemi İçin Karma Parçacık Sürüsü Optimizasyonu	Parçacık sürüsü optimizasyonu ve tavlama benzetimi		
22	2013	Yu-Hsinchen G.	Dinamik Tesis Yerleşimi Problemlerinin Hibrit Karınca Kolonisi Optimizasyonu Çözüm Gösteriminde Yeni Bir Veri Yapısı	Karınca koloni algoritması		
23	2013	Azimi E.S.	Simülasyon Tekniği ve Parçacık Sürüsü Optimizasyonu Kullanarak Dinamik Tesis Yerleşim Problemi İçin Verimli Bir Hibrit Algoritması	Kesikli parçacık sürüsü optimizasyonu		
24	2013	Samarghandi H., Taabayan P., Behroozi M.	Eşit Olmayan Alan Kısıtlamaları ve Yakınlık Dereceleriyle Bulanık Dinamik Tesis Yerleşim Problemi İçin Meta-Sezgiseller	Bulanık meta-sezgisel algoritma		
25	2013	Azimi P., Charmchi H.R.	Bütçe Kısıtlı Dinamik Tesis Yerleşim Problemi İçin Simülasyon Yaklaşımı ile Yeni Bir Optimizasyon	Tam sayılı programlama		
26	2012	Jolai F., Tavakkoli-Moghaddam R., Taghipour M.	Eşit Olmayan Boyutlu Dinamik Tesis Yerleşimi Problemi İçin Alma / Bırakma Konumlarına Göre Çok Amaçlı Parçacık Sürüsü Optimizasyon Algoritması	Çok amaçlı parçacık sürüsü optimizasyonu		
27	2012	Moslemipour G., Lee T.S.	Esnek Üretim Sistemlerinin Belirsiz Ortamında Dinamik Bir Makine Düzeninin Akıllı Tasarımı	Tavlama benzetimi		
28	2011	Yang C., Chuang S., Hsu T.	Atölye İmalatında Dinamik Tesis Planlaması İçin Genetik Bir Algoritma	Genetik algoritma		
29	2011	Ripon K.S.N., Glette K., Hovin M., Torresen J.	Belirsizlik Altında Dinamik Tesis Yerleşim Problemi: Pareto-Optimallik Temelli Çok Amaçlı Evrimsel Bir Yaklaşım	Geriye doğru çift taraflı değişim sezgiseli		
30	2011	Ning X., Lam K., Lam M.C	Şantiye Yerleşim Planlaması İçin Karar Verme Sistemi	Karınca koloni algoritması		
31	2010	Şahin R, Ertoğral K., Türkbey O.	Bütçe Kısıtlı Dinamik Yerleşim Problemi İçin Tavlama Benzetimi Sezgiseli	Tavlama benzetimi		
32	2010	Bashiri M., Dehghan E.	Eşzamanlı Veri Zarflama Analizi Modellemesi Kullanarak Çok Kriterli Dinamik Tesis Yerleşimi Problemini Optimize Etmek	Veri zarflama analizi	3	4
33	2010	Ning X., Lam K., Lam M.C.	Max-Min Karınca Sistemini Kullanarak Dinamik Şantiye Yerleşim Planlaması	Karınca koloni algoritması		
34	2009	Dong M., Wua C., Hou F.	Dinamik İş Ortamı Altında Dinamik Tesis Yerleşim Problemi İçin En Kısa Yol Tabanlı Benzetilmiş Tavlama Algoritması	Tavlama benzetimi	50	7
35	2009	Ulutaş B.H., İşlier A.A.	Dinamik Tesis Yerleşimi Problemleri İçin Klonal Seçim Algoritması	Klonal seçim algoritması		
36	2006	Baykasoğlu A., Dereli T., Sabuncu İ.	Bütçe Kısıtlı ve Kısıtsız Dinamik Tesis Yerleşimi Sorunlarını Çözmek İçin Karınca Kolonisi Algoritması	Karınca koloni algoritması		

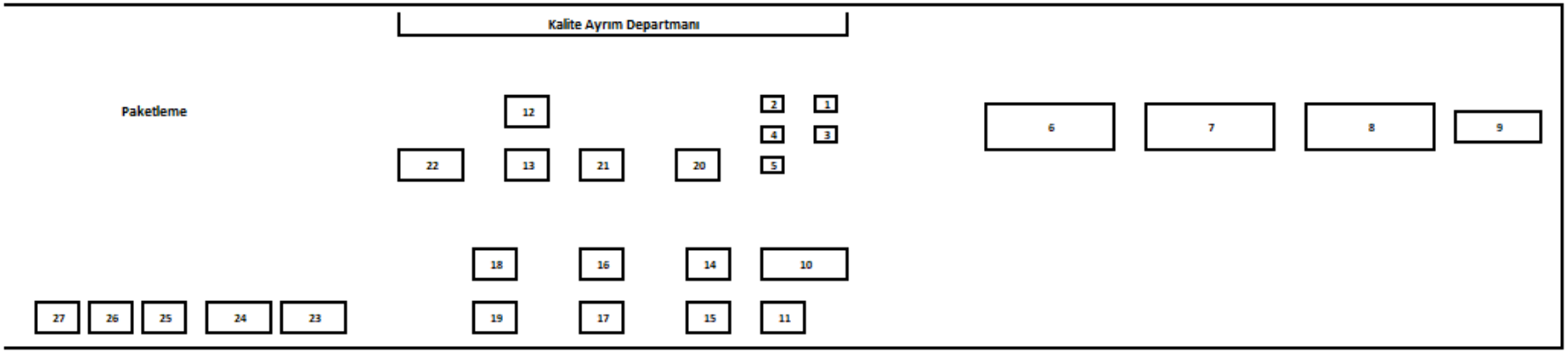
EK-1. (devam) Dinamik tesis yerleşim problemi ile ilgili çalışmalar listesi

37	2006	Mckendall A.R., Shanga J., Kuppusamy S.	Dinamik Tesis Yerleşim Problemi İçin Tavlama Benzetimi Sezgiseli	Tavlama benzetimi		
38	2005	Dunker T., Radons G., Westküamper E.	Dinamik Bir Tesis Yerleşim Problemini Çözmek İçin Evrimsel Hesaplama ve Dinamik Programlamayı Birleştirme	Evrimsel hesaplama ve dinamik programlama		
39	2003	Balakrishnan J., Cheng C.H., Conway D.G., Lau C.M.	Dinamik Tesis Yerleşim Problemi İçin Karma Bir Genetik Algoritma	Hibrit genetik algoritma		
40	1998	Yang T., Peters B.A.	Dinamik ve Belirsiz Üretim Ortamları İçin Esnek Makine Düzeni Tasarımı	Dinamik programlama		
41	1994	Conway D.G., Venkataramanan M.A.	Genetik Arama ve Dinamik Tesis Yerleşim Problemi	Genetik algoritma	9	5
42	1992	Balakrishnan J., Jacobs F.R., Venkataramanan M.A.	Kısıtlı Dinamik Tesis Yerleşimi Problemi İçin Çözümler	Dinamik programlama	6	5
43	2007	Urban T.L.	Dinamik Tesis Yerleşim Problemi İçin Sezgisel	Dinamik programlama		
44	1990	Urban T.L.	Dinamik Tesis Yerleşimi Problemi İçin Alt-Sınır Prosedürlerinin Hesaplanmış Performansı ve Etkinliği	Dinamik programlama		
45	1999	Kochhar J.S., Heragu S.S.	Değişken Bir Ortamda Tesis Yerleşim Tasarımı	Genetik algoritma	12	2
46	2009	Rezazadeh H., Ghazanfari M., Saidi-Mehrabad M., Sadjadi S.J.	Dinamik Tesis Yerleşim Problemi İçin Genişletilmiş Kesikli Parçacık Sürüsü Optimizasyon Algoritması	Parçacık sürüsü optimizasyonu		
47	2017	Hosseini-Nasab H., Fereidouni S., Ghomi S.M.T.F., Fakhrzad M.B	Tesis Yerleşim Problemlerinin Sınıflandırılması: Bir İnceleme Çalışması			
48	2017	Turanoğlu B., Akkaya G.	Yakınlık Oranıyla İlgili Dinamik Tesis Yerleşimi Problemleri: Bulanık Bir Karar Destek Sistemi Yaklaşımı	Bulanık karar destek sistemi	6	5
49	2007	Drira A., Pierreval H., Hajri-Gabouj S.	Tesis Yerleşim Problemleri: Bir Anket			
50	2016	Tayal A., Gunasekaran A., Singh S.P., Dubey R., Papadopoulos T.	Sürdürülebilir Stokastik Dinamik Tesis Yerleşim Problemini Formüle Etme ve Çözme: Sürdürülebilir Operasyonların Anahtarı	Tavlama benzetimi ve veri zarflama analizi	12	5
51	2016	Tayal A., Singh S.P.	Tesis Yerleşim Problemi İçin Büyük Veri Analitiği ve Melez Ateş Böceği-Kaotik Tavlama Benzetimi Yaklaşımı Entegrasyonu	Tavlama benzetimi	12	5
52	2004	Ertay T., Ruan D., Tuzkaya U.R.	Üretim Sistemlerinde Tesis Yerleşimi Tasarımı İçin Veri Zarflama Analizini ve Analitik Hiyerarşiyi Bütünleştirmek	Veri zarflama analizi ve analitik hiyerarşi		
53	2008	Şahin R.	Dinamik Tesis Düzenleme Problemi İçin Bir Tavlama Benzetimi Sezgiseli	Tavlama benzetimi		
54	2013	Bilişik O.G., Bilişik M.T.	Dinamik Tesis Yerleşim Düzenleme Probleminin Çok Ölçütlü Olarak Ele Alınması	Dinamik programlama ve TOPSİS		
55	1997	Lacksonen T.A	Statik ve Dinamik Tesis Yerleşimi Problemleri İçin Ön İşleme	Karışık tamsayılı programlama ve dal-sınır algoritması	12	3

EK-1. (devam) Dinamik tesis yerleşim problemi ile ilgili çalışmalar listesi

56	1998	Urban T.L.	Dinamik Tesis Yerleşim Problemi İçin Çözüm Prosedürleri	Dinamik programlama	9	6

EK-2. Kalite ayrım sonrası operasyonlar departmanı mevcut yerleşim



1- Zımpara
2- İç yüzey tamir
3- Yüzey tamir
4- Delik tamir
5- Delik tamir
6- Taşlama
7- Taşlama
8- Taşlama
9- Taşlama
10- Taşlama
11- Yüzey kaplama
12- Vakum test
13- Vakum test
14- Vakum test
15- Vakum test
16- Vakum test
17- Vakum test
18- Vakum test
19- Vakum test
20- Boru montaj (A tipi)
21- Koku test
22- Montaj (A tipi)
23- Perde kesme
24- Perde kesme
25- Perde kesme
26- Boru montaj (B tipi)
27- Montaj (B tipi)

EK-3. C# ile kodlanan programın kodu

```
using GenAlg.Models;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace GenAlg.Controllers
{
    public class HomeController : Controller
    {
        private ApplicationDbContext db = new ApplicationDbContext();
        public List<Donem> dönemList = new List<Donem>();
        public Dictionary<int, List<Akis>> t1Akış = new Dictionary<int, List<Akis>>();
        public Dictionary<int, List<Akis>> t2Akış = new Dictionary<int, List<Akis>>();
        public Dictionary<int, List<Akis>> t3Akış = new Dictionary<int, List<Akis>>();
        int count = 1;
        public const double X1 = 48;
        public const double X2 = 62;
        public const double X3 = 65;
        public const double X4 = 23;
        public const double X5 = 20;
        public const double Y1 = 17;
        public const double Y2 = 15.5;
        public const double Y3 = 8.5;
        public const double Y4 = 7;
        public const double Y5 = 10.5;
        public const double Y6 = 12;
        public ActionResult Genetic()
```

EK-3. (devam) C# ile kodlanan programın kodu

```
 {
        return View();
    }
    public double fitnessMe(int[] geneList)
    {
        Donem dondon = new Donem();
        List<Istasyon> t1List = new List<Istasyon>();
        List<Istasyon> t2List = new List<Istasyon>();
        List<Istasyon> t3List = new List<Istasyon>();
        t1List.Add(new Istasyon("0"));
        for (int i = 0; i < 27; i++)
        {
            t1List.Add(new Istasyon(geneList[i].ToString()));
        }
        t1List.Add(new Istasyon("28"));
        t2List.Add(new Istasyon("0"));
        for (int i = 27; i < 54; i++)
        {
            t2List.Add(new Istasyon(geneList[i].ToString()));
        }
        t2List.Add(new Istasyon("28"));
        t3List.Add(new Istasyon("0"));
        for (int i = 54; i < 81; i++)
        {
            t3List.Add(new Istasyon(geneList[i].ToString()));
        }
        t3List.Add(new Istasyon("28"));
        dondon.t1List = t1List;
        dondon.t2List = t2List;
```

EK-3. (devam) C# ile kodlanan programın kodu

```
            dondon.t3List = t3List;
            var totalCost = maliyetHesapla(dondon);
            return totalCost;
        }
        [HttpPost]
        public ActionResult RunAlg(int iterationCount, int[] initialArray, int
mutationRate,int crossOverRate)
        {
            ViewBag.iterationCount = iterationCount;
            World world = new World();
            Random rnd = new Random();
            createT1Akis();
            createT2Akis();
            createT3Akis();
            double bestFitnessPoint = double.MaxValue;
            int bestIterationCount = 0;
            int[] bestGeneList = new int[81];
            for (int a = 0; a < iterationCount; a++)
            {
                world.NextPopulation.Clear();
                foreach (var genomeee in world.Population)
                {
                    genomeee.FitnessPoint = fitnessMe(genomeee.Genes);
                }
                double newFitnessPoint = world.Population.Min(p => p.FitnessPoint);
                if(newFitnessPoint< bestFitnessPoint)
                {
                    bestFitnessPoint = newFitnessPoint;
                    bestIterationCount = a+1
```

EK-3. (devam) C# ile kodlanan programın kodu

```
    }
        bestGeneList = world.Population.Where(p => p.FitnessPoint ==
newFitnessPoint).FirstOrDefault().Genes;
        world.Population = world.Population.OrderBy(p => p.FitnessPoint).ToList();
        int bestGeneCount = Convert.ToInt32( 0.25 * (100 - crossOverRate));
        for (int i = 0; i < bestGeneCount; i++)
        {
           world.NextPopulation.Add(new Genome(world.Population[i].Genes));
        }
        for (int i = 0; i < (25-bestGeneCount); i++)
        {
           int select1 = 0;
           int select2 = 0;
           bool isEqual = false;  //DO WHILE ->
           int differenceCount = 0;
              do
              {
                 select1 = rnd.Next(0, 25);
                 select2 = rnd.Next(0, 25);
                 isEqual =
Enumerable.SequenceEqual(world.Population[select1].Genes,
world.Population[select2].Genes);
                 differenceCount++;
              if(differenceCount > 25)
              {
                 string result2 = "";
                 for (int ai = 0; ai < 81; ai++)
                 {
                    result2 = result2 + "-" + bestGeneList[i].ToString();
                 }
```

EK-3. (devam) C# ile kodlanan programın kodu

```
                    return Content("[Yakinsama]Iteration: " + a + " Best Fitness : " +
bestFitnessPoint  + " " + "Best Gene List : "  + result2);
                }
                } while (isEqual);
                int[] childGenome =
GeneretaChildGenom(world.Population[select1].Genes,
world.Population[select2].Genes, rnd, mutationRate);
                world.NextPopulation.Add(new Genome(childGenome));
            }
            for (int c = 0; c < 25; c++)
            {
                world.NextPopulation[c].Genes.CopyTo(world.Population[c].Genes, 0);
            }
        }
        string result = "";
        for (int i = 0; i < 81; i++)
        {
            result = result + "-" + bestGeneList[i].ToString();
        }
        AlgResult model = new AlgResult()
        {
            BestFitness = bestFitnessPoint,
            BestIterationCount = bestIterationCount,
            BestGenes = result
        };
        return View(model);
    }
    public int[] GeneretaChildGenom(int[] parent1, int[] parent2, Random rnd, int
mutationRate)
    {
```

EK-3. (devam) C# ile kodlanan programın kodu

```
int[] param1 = new int[27];
int[] param2 = new int[27];
int[] childGenom = new int[81];
for (int i = 0; i < 81; i++)
{
   param1[i % 27] = parent1[i];
   param2[i % 27] = parent2[i];
   int a = rnd.Next(0, 26);
   int b = rnd.Next(0, 26);
   int c = rnd.Next(0, 26);
   int needMutation = rnd.Next(0, 101);
   if (i == 26)
   {
      var result1 = CrossMe(param1, param2, a);
      Array.Clear(param1, 0, 26);
      Array.Clear(param2, 0, 26);
      if (needMutation <= mutationRate)
      {
         int d = rnd.Next(0, 26);
         int e = rnd.Next(0, 26);
         int first = result1[d];
         int second = result1[e];
         result1[d] = second;
         result1[e] = first;
      }
      result1.CopyTo(childGenom, 0);
   }
   if (i == 53)
   {
```

EK-3. (devam) C# ile kodlanan programın kodu

```
            var result2 = CrossMe(param1, param2, b);
            Array.Clear(param1, 0, 26);
            Array.Clear(param2, 0, 26);
            if (needMutation <= mutationRate)
            {
                int d = rnd.Next(0, 26);
                int e = rnd.Next(0, 26);
                int first = result2[d];
                int second = result2[e];
                result2[d] = second;
                result2[e] = first;
            }
            result2.CopyTo(childGenom, 27);
        }
        if (i == 80)
        {
            var result3 = CrossMe(param1, param2, c);
            Array.Clear(param1, 0, 26);
            Array.Clear(param2, 0, 26);
            if (needMutation <= mutationRate)
            {
                int d = rnd.Next(0, 26);
                int e = rnd.Next(0, 26);
                int first = result3[d];
                int second = result3[e];
                result3[d] = second;
                result3[e] = first;
            }
            result3.CopyTo(childGenom, 54);
```

EK-3. (devam) C# ile kodlanan programın kodu

```
            }
        }
        return childGenom;
    }
    public int[] CrossMe(int[] PeriodParent1, int[] PeriodParent2, int crossPoint1)
    {
        int[] childParent = new int[27];
        for (int i = 0; i < 27; i++)
        {
            if (i <= crossPoint1)
            {
                childParent[i] = PeriodParent1[i];
            }
        }
        for (int i = 0; i < 27; i++)
        {
            if (!childParent.Contains(PeriodParent2[i]))
            {
                childParent[crossPoint1 + 1] = PeriodParent2[i];
                crossPoint1++;
            }
        }
        return childParent;
    }
    public void MainFitness()
    {
        Donem dönem1 = new Donem();
        List<Istasyon> t1List = new List<Istasyon>();
        List<Istasyon> t2List = new List<Istasyon>();
```

EK-3. (devam) C# ile kodlanan programın kodu

```
List<Istasyon> t3List = new List<Istasyon>();
t1List.Add(new Istasyon("0"));
t1List.Add(new Istasyon("1"));
t1List.Add(new Istasyon("2"));
t1List.Add(new Istasyon("3"));
t1List.Add(new Istasyon("5"));
t1List.Add(new Istasyon("6"));
t1List.Add(new Istasyon("7"));
t1List.Add(new Istasyon("4"));
t1List.Add(new Istasyon("8"));
t1List.Add(new Istasyon("9"));
t1List.Add(new Istasyon("12"));
t1List.Add(new Istasyon("11"));
t1List.Add(new Istasyon("10"));
t1List.Add(new Istasyon("19"));
t1List.Add(new Istasyon("18"));
t1List.Add(new Istasyon("17"));
t1List.Add(new Istasyon("16"));
t1List.Add(new Istasyon("15"));
t1List.Add(new Istasyon("14"));
t1List.Add(new Istasyon("13"));
t1List.Add(new Istasyon("20"));
t1List.Add(new Istasyon("22"));
t1List.Add(new Istasyon("21"));
t1List.Add(new Istasyon("25"));
t1List.Add(new Istasyon("23"));
t1List.Add(new Istasyon("24"));
t1List.Add(new Istasyon("26"));
t1List.Add(new Istasyon("27"));
```

EK-3. (devam) C# ile kodlanan programın kodu

```
t1List.Add(new Istasyon("28"));
dönem1.t1List = t1List;
t2List.Add(new Istasyon("0"));
t2List.Add(new Istasyon("1"));
t2List.Add(new Istasyon("2"));
t2List.Add(new Istasyon("3"));
t2List.Add(new Istasyon("4"));
t2List.Add(new Istasyon("5"));
t2List.Add(new Istasyon("6"));
t2List.Add(new Istasyon("7"));
t2List.Add(new Istasyon("8"));
t2List.Add(new Istasyon("9"));
t2List.Add(new Istasyon("10"));
t2List.Add(new Istasyon("11"));
t2List.Add(new Istasyon("12"));
t2List.Add(new Istasyon("13"));
t2List.Add(new Istasyon("14"));
t2List.Add(new Istasyon("15"));
t2List.Add(new Istasyon("16"));
t2List.Add(new Istasyon("17"));
t2List.Add(new Istasyon("18"));
t2List.Add(new Istasyon("19"));
t2List.Add(new Istasyon("20"));
t2List.Add(new Istasyon("21"));
t2List.Add(new Istasyon("22"));
t2List.Add(new Istasyon("23"));
t2List.Add(new Istasyon("24"));
t2List.Add(new Istasyon("25"));
t2List.Add(new Istasyon("26"));
```

EK-3. (devam) C# ile kodlanan programın kodu

```
t2List.Add(new Istasyon("27"));
t2List.Add(new Istasyon("28"));
dönem1.t2List = t2List;
t3List.Add(new Istasyon("0"));
t3List.Add(new Istasyon("27"));
t3List.Add(new Istasyon("26"));
t3List.Add(new Istasyon("25"));
t3List.Add(new Istasyon("24"));
t3List.Add(new Istasyon("23"));
t3List.Add(new Istasyon("22"));
t3List.Add(new Istasyon("21"));
t3List.Add(new Istasyon("20"));
t3List.Add(new Istasyon("19"));
t3List.Add(new Istasyon("18"));
t3List.Add(new Istasyon("17"));
t3List.Add(new Istasyon("16"));
t3List.Add(new Istasyon("15"));
t3List.Add(new Istasyon("14"));
t3List.Add(new Istasyon("13"));
t3List.Add(new Istasyon("12"));
t3List.Add(new Istasyon("11"));
t3List.Add(new Istasyon("10"));
t3List.Add(new Istasyon("9"));
t3List.Add(new Istasyon("8"));
t3List.Add(new Istasyon("7"));
t3List.Add(new Istasyon("6"));
t3List.Add(new Istasyon("5"));
t3List.Add(new Istasyon("4"));
t3List.Add(new Istasyon("3"));
```

EK-3. (devam) C# ile kodlanan programın kodu

```
            t3List.Add(new Istasyon("2"));
            t3List.Add(new Istasyon("1"));
            t3List.Add(new Istasyon("28"));
            dönem1.t3List = t3List;
            dönemList.Add(dönem1);
            count++;
            createT1Akis();
            createT2Akis();
            createT3Akis();
            double totalMaliyet = maliyetHesapla(dönem1);
        }
        private double maliyetHesapla(Donem dönem)
        {
            double totalMaliyet = 0.0;
            dönem.t1List = spiralYerleştirme(dönem.t1List);
            dönem.t2List = spiralYerleştirme(dönem.t2List);
            dönem.t3List = spiralYerleştirme(dönem.t3List);
            double t1Maliyet = akisMaliyet(t1Akış, dönem.t1List);
            double t2Maliyet = akisMaliyet(t2Akış, dönem.t2List);
            double t3Maliyet = akisMaliyet(t3Akış, dönem.t3List);
            double t1DenT2eYerdeğiştirmeMaliyeti = yerdeğiştirmeMaliyeti(dönem.t1List,
dönem.t2List);
            double t2DenT3eYerdeğiştirmeMaliyeti = yerdeğiştirmeMaliyeti(dönem.t2List,
dönem.t3List);
            totalMaliyet = t1Maliyet + t2Maliyet + t3Maliyet +
t1DenT2eYerdeğiştirmeMaliyeti + t2DenT3eYerdeğiştirmeMaliyeti;
            return totalMaliyet;
        }
        private List<Istasyon> spiralYerleştirme(List<Istasyon> t)
        {
```

EK-3. (devam) C# ile kodlanan programın kodu

```
double x1Length = X1;
double y1Length = Y2;
double x2Length = X2;
double y2Length = Y3;
double x3Length = X4;

for (int i = 0; i < t.Count-1 ; i++)
{
   Istasyon ist = t[i];
   x1Length += ist.X;
   if (x1Length <= X2)
   {
      ist.agırlıkX = x1Length - (ist.X / 2.0);
      ist.agırlıkY = Y1;
   }
   else
   {
      y1Length -= ist.Y;
      if (y1Length >= Y3)
      {
         ist.agırlıkX = X3;
         ist.agırlıkY = y1Length + (ist.Y / 2.0);
      }
      else
      {
         x2Length -= ist.X;
         if (x2Length >= X4)
         {
            ist.agırlıkX = x2Length + (ist.X / 2.0);
```

EK-3. (devam) C# ile kodlanan programın kodu

```
                    ist.agırlıkY = Y4;
                }
                else
                {
                    y2Length += ist.Y;
                    if (y2Length <= Y5)
                    {
                        ist.agırlıkX = X5;
                        ist.agırlıkY = y2Length - (ist.Y / 2.0);
                    }
                    else
                    {
                        x3Length += ist.X;
                        if (x3Length <= X1)
                        {
                            ist.agırlıkX = x3Length - (ist.X / 2.0);
                            ist.agırlıkY = Y6;
                        }
                    }
                }
            }
        }
        t[i] = ist;
    }
    return t;
}
private double akisMaliyet(Dictionary<int, List<Akis>> akis, List<Istasyon> list)
{
    double maliyet = 0.0;
```

EK-3. (devam) C# ile kodlanan programın kodu

```
        foreach (KeyValuePair<int, List<Akis>> item in akis)
        {
            int key = item.Key;
            List<Akis> akisList = item.Value;
            Istasyon keyIstasyon = new Istasyon();
            foreach (var ist in list)
            {
                if (ist.id == key) keyIstasyon = ist;
            }
            foreach (var akıs in akisList)
            {
                foreach (var ist in list)
                {
                    if (akıs.id == ist.id)
                    {
                        double x = Math.Abs(keyIstasyon.agırlıkX - ist.agırlıkX);
                        double y = Math.Abs(keyIstasyon.agırlıkY - ist.agırlıkY);
                        double istMaliyet = (x + y) * akıs.maliyet;
                        maliyet += istMaliyet;
                    }
                }
            }
        }
        return maliyet;
    }
    private double yerdeğiştirmeMaliyeti(List<Istasyon> list1, List<Istasyon> list2)
    {
        double maliyet = 0.0;
        for (int i = 0; i < 27; i++)
```

EK-3. (devam) C# ile kodlanan programın kodu

```
        {
            Istasyon ist1 = list1[i];
            Istasyon ist2 = list2[i];
            if (ist1.id != ist2.id) maliyet += 1000;
        }
        return maliyet;
    }
    private void createT1Akis()
    {
        t1Akış.Clear();
        List<Akis> baslangıc = new List<Akis>();
        baslangıc.Add(new Akis(1, 650000));
        t1Akış.Add(0, baslangıc);
        List<Akis> akis1 = new List<Akis>();
        akis1.Add(new Akis(2, 500000));
        akis1.Add(new Akis(3, 150000));
        t1Akış.Add(1, akis1);
        List<Akis> akis2 = new List<Akis>();
        akis2.Add(new Akis(3, 500000));
        t1Akış.Add(2, akis2);
        List<Akis> akis3 = new List<Akis>();
        akis3.Add(new Akis(4, 150000));
        akis3.Add(new Akis(7, 150000));
        akis3.Add(new Akis(8, 150000));
        akis3.Add(new Akis(9, 150000));
        akis3.Add(new Akis(10, 50000));
        t1Akış.Add(3, akis3);
        List<Akis> akis4 = new List<Akis>();
        akis4.Add(new Akis(6, 150000));
```

EK-3. (devam) C# ile kodlanan programın kodu

```
t1Akış.Add(4, akis4);
List<Akis> akis5 = new List<Akis>();
t1Akış.Add(5, akis5);
List<Akis> akis6 = new List<Akis>();
akis6.Add(new Akis(12, 100000));
akis6.Add(new Akis(13, 50000));
t1Akış.Add(6, akis6);
List<Akis> akis7 = new List<Akis>();
akis7.Add(new Akis(13, 50000));
akis7.Add(new Akis(14, 100000));
t1Akış.Add(7, akis7);
List<Akis> akis8 = new List<Akis>();
akis8.Add(new Akis(15, 100000));
akis8.Add(new Akis(16, 50000));
t1Akış.Add(8, akis8);
List<Akis> akis9 = new List<Akis>();
akis9.Add(new Akis(16, 50000));
akis9.Add(new Akis(17, 100000));
t1Akış.Add(9, akis9);
List<Akis> akis10 = new List<Akis>();
akis10.Add(new Akis(18, 50000));
t1Akış.Add(10, akis10);
List<Akis> akis11 = new List<Akis>();
t1Akış.Add(11, akis11);
List<Akis> akis12 = new List<Akis>();
akis12.Add(new Akis(20, 100000));
t1Akış.Add(12, akis12);
List<Akis> akis13 = new List<Akis>();
akis13.Add(new Akis(20, 50000));
```

EK-3. (devam) C# ile kodlanan programın kodu

```
akis13.Add(new Akis(23, 50000));
t1Akış.Add(13, akis13);
List<Akis> akis14 = new List<Akis>();
akis14.Add(new Akis(23, 100000));
t1Akış.Add(14, akis14);
List<Akis> akis15 = new List<Akis>();
akis15.Add(new Akis(23, 30000));
akis15.Add(new Akis(24, 70000));
t1Akış.Add(15, akis15);
List<Akis> akis16 = new List<Akis>();
akis16.Add(new Akis(24, 100000));
t1Akış.Add(16, akis16);
List<Akis> akis17 = new List<Akis>();
akis17.Add(new Akis(24, 10000));
akis17.Add(new Akis(25, 90000));
t1Akış.Add(17, akis17);
List<Akis> akis18 = new List<Akis>();
akis18.Add(new Akis(25, 50000));
t1Akış.Add(18, akis18);
List<Akis> akis19 = new List<Akis>();
t1Akış.Add(19, akis19);
List<Akis> akis20 = new List<Akis>();
akis20.Add(new Akis(21, 150000));
t1Akış.Add(20, akis20);
List<Akis> akis21 = new List<Akis>();
akis21.Add(new Akis(22, 150000));
t1Akış.Add(21, akis21);
List<Akis> akis22 = new List<Akis>();
akis22.Add(new Akis(28, 150000));
```

EK-3. (devam) C# ile kodlanan programın kodu

```
        t1Akış.Add(22, akis22);
        List<Akis> akis23 = new List<Akis>();
        akis23.Add(new Akis(26, 180000));
        t1Akış.Add(23, akis23);
        List<Akis> akis24 = new List<Akis>();
        akis24.Add(new Akis(26, 180000));
        t1Akış.Add(24, akis24);
        List<Akis> akis25 = new List<Akis>();
        akis25.Add(new Akis(26, 140000));
        t1Akış.Add(25, akis25);
        List<Akis> akis26 = new List<Akis>();
        akis26.Add(new Akis(27, 500000));
        t1Akış.Add(26, akis26);
        List<Akis> akis27 = new List<Akis>();
        akis27.Add(new Akis(28, 500000));
        t1Akış.Add(27, akis27);
        List<Akis> bitis = new List<Akis>();
        t1Akış.Add(28, bitis);
    }
    private void createT2Akis()
    {
        t2Akış.Clear();
        List<Akis> baslangıc = new List<Akis>();
        baslangıc.Add(new Akis(1, 650000));
        t2Akış.Add(0, baslangıc);
        List<Akis> akis1 = new List<Akis>();
        akis1.Add(new Akis(2, 300000));
        akis1.Add(new Akis(3, 350000));
        t2Akış.Add(1, akis1);
```

EK-3. (devam) C# ile kodlanan programın kodu

```
List<Akis> akis2 = new List<Akis>();
akis2.Add(new Akis(3, 300000));
t2Akış.Add(2, akis2);
List<Akis> akis3 = new List<Akis>();
akis3.Add(new Akis(4, 350000));
akis3.Add(new Akis(8, 100000));
akis3.Add(new Akis(9, 150000));
akis3.Add(new Akis(10, 50000));
t2Akış.Add(3, akis3);
List<Akis> akis4 = new List<Akis>();
akis4.Add(new Akis(6, 150000));
akis4.Add(new Akis(7, 150000));
akis4.Add(new Akis(8, 50000));
t2Akış.Add(4, akis4);
List<Akis> akis5 = new List<Akis>();
t2Akış.Add(5, akis5);
List<Akis> akis6 = new List<Akis>();
akis6.Add(new Akis(11, 150000));
t2Akış.Add(6, akis6);
List<Akis> akis7 = new List<Akis>();
akis7.Add(new Akis(11, 150000));
t2Akış.Add(7, akis7);
List<Akis> akis8 = new List<Akis>();
akis8.Add(new Akis(11, 150000));
t2Akış.Add(8, akis8);
List<Akis> akis9 = new List<Akis>();
akis9.Add(new Akis(11, 150000));
t2Akış.Add(9, akis9);
List<Akis> akis10 = new List<Akis>();
```

EK-3. (devam) C# ile kodlanan programın kodu

```
akis10.Add(new Akis(11, 50000));
t2Akış.Add(10, akis10);
List<Akis> akis11 = new List<Akis>();
akis11.Add(new Akis(12, 100000));
akis11.Add(new Akis(13, 100000));
akis11.Add(new Akis(14, 100000));
akis11.Add(new Akis(15, 100000));
akis11.Add(new Akis(16, 100000));
akis11.Add(new Akis(17, 100000));
akis11.Add(new Akis(18, 50000));
t2Akış.Add(11, akis11);
List<Akis> akis12 = new List<Akis>();
akis12.Add(new Akis(20, 100000));
t2Akış.Add(12, akis12);
List<Akis> akis13 = new List<Akis>();
akis13.Add(new Akis(20, 100000));
t2Akış.Add(13, akis13);
List<Akis> akis14 = new List<Akis>();
akis14.Add(new Akis(20, 100000));
t2Akış.Add(14, akis14);
List<Akis> akis15 = new List<Akis>();
akis15.Add(new Akis(20, 50000));
akis15.Add(new Akis(23, 50000));
t2Akış.Add(15, akis15);
List<Akis> akis16 = new List<Akis>();
akis16.Add(new Akis(23, 100000));
t2Akış.Add(16, akis16);
List<Akis> akis17 = new List<Akis>();
akis17.Add(new Akis(23, 30000));
```

EK-3. (devam) C# ile kodlanan programın kodu

```
akis17.Add(new Akis(24, 70000));
t2Akış.Add(17, akis17);
List<Akis> akis18 = new List<Akis>();
akis18.Add(new Akis(24, 50000));
t2Akış.Add(18, akis18);
List<Akis> akis19 = new List<Akis>();
t2Akış.Add(19, akis19);
List<Akis> akis20 = new List<Akis>();
akis20.Add(new Akis(21, 350000));
t2Akış.Add(20, akis20);
List<Akis> akis21 = new List<Akis>();
akis21.Add(new Akis(22, 350000));
t2Akış.Add(21, akis21);
List<Akis> akis22 = new List<Akis>();
akis22.Add(new Akis(28, 350000));
t2Akış.Add(22, akis22);
List<Akis> akis23 = new List<Akis>();
akis23.Add(new Akis(26, 180000));
t2Akış.Add(23, akis23);
List<Akis> akis24 = new List<Akis>();
akis24.Add(new Akis(26, 120000));
t2Akış.Add(24, akis24);
List<Akis> akis25 = new List<Akis>();
t2Akış.Add(25, akis25);
List<Akis> akis26 = new List<Akis>();
akis26.Add(new Akis(27, 300000));
t2Akış.Add(26, akis26);
List<Akis> akis27 = new List<Akis>();
akis27.Add(new Akis(28, 300000));
```

EK-3. (devam) C# ile kodlanan programın kodu

```
        t2Akış.Add(27, akis27);
        List<Akis> bitis = new List<Akis>();
        t2Akış.Add(28, bitis);
    }
    private void createT3Akis()
    {
        t3Akış.Clear();
        List<Akis> baslangıc = new List<Akis>();
        baslangıc.Add(new Akis(1, 750000));
        t3Akış.Add(0, baslangıc);
        List<Akis> akis1 = new List<Akis>();
        akis1.Add(new Akis(2, 100000));
        akis1.Add(new Akis(3, 650000));
        t3Akış.Add(1, akis1);
        List<Akis> akis2 = new List<Akis>();
        akis2.Add(new Akis(3, 100000));
        t3Akış.Add(2, akis2);
        List<Akis> akis3 = new List<Akis>();
        akis3.Add(new Akis(4, 350000));
        akis3.Add(new Akis(5, 300000));
        akis3.Add(new Akis(10, 100000));
        t3Akış.Add(3, akis3);
        List<Akis> akis4 = new List<Akis>();
        akis4.Add(new Akis(6, 150000));
        akis4.Add(new Akis(7, 150000));
        akis4.Add(new Akis(8, 50000));
        t3Akış.Add(4, akis4);
        List<Akis> akis5 = new List<Akis>();
        akis5.Add(new Akis(8, 100000));
```

EK-3. (devam) C# ile kodlanan programın kodu

```
akis5.Add(new Akis(9, 150000));
akis5.Add(new Akis(10, 50000));
t3Akış.Add(5, akis5);
List<Akis> akis6 = new List<Akis>();
akis6.Add(new Akis(11, 150000));
t3Akış.Add(6, akis6);
List<Akis> akis7 = new List<Akis>();
akis7.Add(new Akis(11, 150000));
t3Akış.Add(7, akis7);
List<Akis> akis8 = new List<Akis>();
akis8.Add(new Akis(11, 150000));
t3Akış.Add(8, akis8);
List<Akis> akis9 = new List<Akis>();
akis9.Add(new Akis(11, 150000));
t3Akış.Add(9, akis9);
List<Akis> akis10 = new List<Akis>();
akis10.Add(new Akis(11, 150000));
t3Akış.Add(10, akis10);
List<Akis> akis11 = new List<Akis>();
akis11.Add(new Akis(12, 100000));
akis11.Add(new Akis(13, 100000));
akis11.Add(new Akis(14, 100000));
akis11.Add(new Akis(15, 100000));
akis11.Add(new Akis(16, 100000));
akis11.Add(new Akis(17, 100000));
akis11.Add(new Akis(18, 100000));
akis11.Add(new Akis(19, 50000));
t3Akış.Add(11, akis11);
List<Akis> akis12 = new List<Akis>();
```

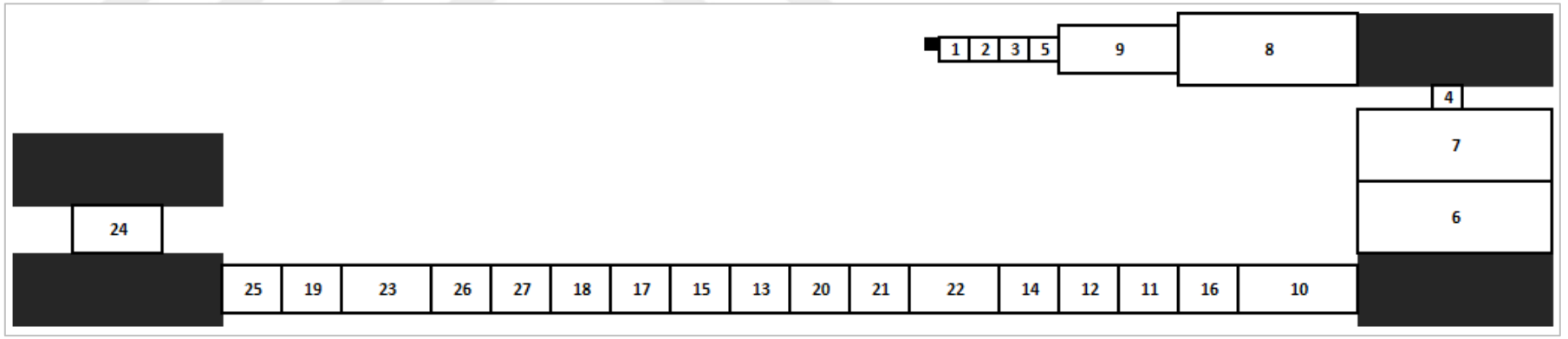
EK-3. (devam) C# ile kodlanan programın kodu

```
akis12.Add(new Akis(20, 100000));
t3Akış.Add(12, akis12);
List<Akis> akis13 = new List<Akis>();
akis13.Add(new Akis(20, 100000));
t3Akış.Add(13, akis13);
List<Akis> akis14 = new List<Akis>();
akis14.Add(new Akis(20, 100000));
t3Akış.Add(14, akis14);
List<Akis> akis15 = new List<Akis>();
akis15.Add(new Akis(20, 100000));
t3Akış.Add(15, akis15);
List<Akis> akis16 = new List<Akis>();
akis16.Add(new Akis(20, 100000));
t3Akış.Add(16, akis16);
List<Akis> akis17 = new List<Akis>();
akis17.Add(new Akis(20, 100000));
t3Akış.Add(17, akis17);
List<Akis> akis18 = new List<Akis>();
akis18.Add(new Akis(20, 50000));
akis18.Add(new Akis(23, 50000));
t3Akış.Add(18, akis18);
List<Akis> akis19 = new List<Akis>();
akis19.Add(new Akis(23, 50000));
t3Akış.Add(19, akis19);
List<Akis> akis20 = new List<Akis>();
akis20.Add(new Akis(21, 650000));
t3Akış.Add(20, akis20);
List<Akis> akis21 = new List<Akis>();
akis21.Add(new Akis(22, 650000));
```

EK-3. (devam) C# ile kodlanan programın kodu

```
            t3Akış.Add(21, akis21);
            List<Akis> akis22 = new List<Akis>();
            akis22.Add(new Akis(28, 650000));
            t3Akış.Add(22, akis22);
            List<Akis> akis23 = new List<Akis>();
            akis23.Add(new Akis(26, 100000));
            t3Akış.Add(23, akis23);
            List<Akis> akis24 = new List<Akis>();
            t3Akış.Add(24, akis24);
            List<Akis> akis25 = new List<Akis>();
            t3Akış.Add(25, akis25);
            List<Akis> akis26 = new List<Akis>();
            akis26.Add(new Akis(27, 100000));
            t3Akış.Add(26, akis26);
            List<Akis> akis27 = new List<Akis>();
            akis27.Add(new Akis(28, 100000));
            t3Akış.Add(27, akis27);
            List<Akis> bitis = new List<Akis>();
            t3Akış.Add(28, bitis);
        }
    }
}
```

EK-4. Kalite ayrım sonrası operasyonlar departmanı iyileştirme sonrası t_3 dönemi yerleşimi



ÖZGEÇMİŞ

Adı Soyadı : Melis ÜREM

Yabancı Dil : İngilizce

Doğum Yeri ve Yılı : İzmir-Ödemiş/1990

E-Posta : uremmelis@gmail.com

Mesleki Geçmişi:

- 2019 – Halen, Yalın Üretim Mühendisi, Kordsa Teknik Tekstil A.Ş., Operasyonel Mükemmellik Müdürlüğü
- 2017 – 2019, Süreç Geliştirme Uzmanı, Eczacıbaşı Yapı Gereçleri A.Ş., Süreç Geliştirme Departmanı
- 2015 – 2016, Planlama ve Stok Kontrol Mühendisi, Ege Fren A.Ş., Üretim Planlama ve Kontrol Departmanı

Eğitim Geçmişi:

- 2016 – 2019, Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Endüstri Mühendisliği Anabilim Dalı
- 2009 – 2014, Eskişehir Anadolu Üniversitesi, Mühendislik Fakültesi, Endüstri Mühendisliği Bölümü