DEĞİŞKEN YOĞUNLUKLU KAYNAK KISITLI PROJE ÇİZELGELEME İÇİN MATEMATİKSEL MODELLEME VE GENETİK ALGORİTMA YAKLAŞIMI

Mastaneh JOUSHANI

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DEĞİŞKEN YOĞUNLUKLU KAYNAK KISITLI PROJE ÇİZELGELEME İÇİN MATEMATİKSEL MODELLEME VE GENETİK ALGORİTMA YAKLAŞIMI

Mastaneh JOUSHANI
0000-0002-6133-3541

Prof. Dr. Erdal EMEL
(Danışman)

YÜKSEK LİSANS TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2022

TEZ ONAYI

Mastaneh JOUSHANI tarafından hazırlanan "DEĞİŞKEN YOĞUNLUKLU KAYNAK KISITLI PROJE ÇİZELGELEME İÇİN MATEMATİKSEL MODELLEME VE GENETİK ALGORİTMA YAKLAŞIMI" adlı tez çalışması aşağıdaki jüri tarafından oy birliği ile Bursa Uludağ Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Prof. Dr. Erdal EMEL

Başkan:	Prof. Dr. Erdal EMEL 0000-0002-9220-7353 Bursa Uludağ Üniversitesi Mühendislik Fakültesi, Endüstri Mühendisliği Anabilim Dalı	İmza
Üye:	Prof. Dr. Müjgan SAĞIR 0000-0003-2781-658X ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ Mühendislik – Mimarlık Fakültesi, Endüstri Mühendisliği Anabilim Dalı	İmza
Üye:	Doç. Dr. Betül YAĞMAHAN 0000-0003-1744-3062 Bursa Uludağ Üniversitesi Mühendislik Fakültesi, Endüstri Mühendisliği Anabilim Dalı	İmza

Yukarıdaki sonucu onaylarım

Prof. Dr. Hüseyin Aksel EREN
Enstitü Müdürü
…/ 07 / 2022

B.U.Ü. Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

14 / 06 / 2022

Mastaneh JOUSHANI

ÖZET

Yüksek Lisans

DEĞİŞKEN YOĞUNLUKLU KAYNAK KISITLI PROJE ÇİZELGELEME İÇİN MATEMATİKSEL MODELLEME VE GENETİK ALGORİTMA YAKLAŞIMI

Mastaneh JOUSHANI

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Endüstri Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Erdal EMEL

Üretim planlaması için proje çizelgeleme yaklaşımları, ürün karmaşık ve yüksek oranda özelleştirilmiş olduğundan, genellikle sipariş üzerine üretim (MTO) veya sipariş üzerine mühendislik (ETO) sistemlerinde kullanılır. Bu sistemlerde, üretimin her aşamasının kendi karmaşıklıkları ve özellikleri vardır. Bu sebeple genel olarak üretimin tüm faaliyetleri bir projenin aşamaları olarak düşünülebilir. Bu tür üretimde, basit bitiş-başlangıç öncüllük ilişkileri gerçek üretim sürecini doğru bir şekilde temsil etmez, bu nedenle üretim süresini ve maliyetini en aza indirmek için faaliyetler arasındaki örtüşmeye izin verilmelidir. Bu çalışmada, değişken yoğunluk formülasyonu ve dört farklı öncüllük ilişkileri kullanılarak kaynak kullanımını dengelemek ve üretim süresini en aza indirmek için bir matematiksel model geliştirilmiştir. Bu modelde tüm proje faaliyetleri değişken yoğunluk formülüne göre gerçekleştirilmektedir. Bu, belirli bir süre içinde tamamlanan bir faaliyetin yüzdesinin, o anda gereken kaynak tahsisi miktarına bağlı olduğu anlamına gelir. Bu model, NP (non-deterministic polynomial) zor problemler sınıfına aittir; bu nedenle, bahsedilen problemdeki uygun çözümleri hesaplamak için bir genetik algoritma yeni bir kromozom yapısı ile birlikte önerilmiştir. Genetik algoritmanın parametreleri üç seviyeli deneyler üzerinden optimize edilmiştir. Sonuçların, matematiksel model ve genetik algoritma için küçük problemlerde aynı olduğu, orta ve büyük problemlerde ise matematiksel modelin sonuç bulamadığı görülmektedir.

Anahtar Kelimeler: Kaynak kısıtlı proje çizelgeleme, siparişe göre üretim, siparişe göre mühendislik, değişken yoğunluk formülasyonu, genetik algoritma, kaynak dengeleme.
2022, vii + 98 sayfa.

ABSTRACT

MSc Thesis

MATHEMATICAL MODELING AND GENETIC ALGHORITHM APPROACH FOR VARIABLE DENSITY RESOURCE CONSTRAINED PROJECT SCHEDULING

Mastaneh JOUSHANI

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Industrial Engineering

Supervisor: Prof. Dr. Erdal EMEL

The use of project scheduling approaches for production planning is often used in make-to-order (MTO) or engineer-to-order (ETO) systems because the product is complex and highly customized. In these systems, each stage of production has its own complexities and features. Therefore, in general, all activities of production can be considered as stages of a project. In this type of production, simple finish-start precedence relationships do not accurately represent the actual production process; so overlap between activities must be allowed to minimize the production time and cost. In this study, a mathematical model is developed to balance the resource usage and minimize the production time by using a variable intensity formulation and four different precedence relationships. In this model, all project activities are carried out according to the variable intensity formula. This means that the percentage of an activity completed in a given time period depends on the amount of resource allocation required at that time. This model belongs to the class of NP hard (non-deterministic polynomial) problems, therefore, a genetic algorithm is proposed to calculate suitable solutions in the aforementioned problem. A novel chromosome structure is proposed for this genetic algorithm, and a three-level combinatorial searchis applied to adjust the parameters of the genetic algorithm. The solution obtained by the mathematical model and the genetic algorithm are same for small problems but for medium and large datasets only the genetic algorithm provides sub_optimal solutions.

Key words: Resource-constrained project scheduling, manufacturing-to-order, engineering-to-order, variable intensity formulation, genetic algorithm, resource levelling.
2022, vii + 98 pages.

TEŞEKKÜR

Öncelikle, Bursa Uludağ Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği Bölümü'nde yüksek lisans eğitimine başladığım günden beri bana yeni bir bakış açısı kazandıran ve tez önerisi için yardımcı olan değerli hocam merhum Prof. Dr. Cenk ÖZMUTLU 'yı saygı ve hürmetle anmak istiyorum.

Prof. Dr. Erdal Emel'e yüksek lisans ve tez geliştirmemin her aşamasında verdiği bilgi, rehberlik, eğitim olanakları ve bana ayırdığı değerli zamanının yanı sıra yoğun programlarına rağmen tez danışmanım olmayı kabul ettiği için teşekkür ederim.

Öncelikle kız kardeşim Bahareh Joushani'ye ve eşim Reza Karimi'ye, bana verdikleri emeğin karşılığını ödeyemeyeceğimi bildiğim ve beni cesaretlendiren annem ve babama teşekkür ederim.

Mastaneh JOUSHANI

14/ 06 / 2022

İÇİNDEKİLER

Sayfa

KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
Cmax	Maximum Completion Makespan
CtS	%Completed-to-Start
CtF	%Completed-to-Finish
ETO	Engineer-to-order
FtC	Finish-to-%Completed
GA	Genetik Algoritma
GPR	Generalized Precedence Relations
KKPÇP	Kaynak kısıtlı proje çizelgeleme problemi
MTO	Make-to-order
StC	Start-to-%Completed

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

1. GİRİŞ

1.1. Çalışmanın Amacı

Düşük hacimli ve uzun tedarik süreleri ile yüksek oranda özelleştirilmiş ürünler sağlayan üretim sistemleri, kesikli üretim sistemleri olarak sınıflandırılır. Bu ürünler benzersizdir ve seri üretim sistemlerinde üretilemez. Bu üretim ortamları için tipik üretim stratejileri, Siparişe Göre Üretim (MTO) ve Siparişe Göre Mühendislik (ETO) şeklindedir. Bu stratejilerin karakteristik faktörü, üretim sürecinden önce ve müşteri sipariş onayından sonra başlayan müşteri sipariş noktasıdır. MTO ve ETO stratejileri, üretim tesislerinin yüksek esnekliği, yüksek belirsiz işlem süresi, ekipmanın işlevsel yerleşimi ve tasarım, üretim ve teknolojik gereksinim açısından benzersiz ürünler olarak ve geniş bir ürün yelpazesini karşılamaya yönelik öncüllük kısıtlamaları ile karakterize edilir (Zennaro ve ark. 2019).

Bu sistemlerde üretim planlaması, yüksek düzeyde üretim çeşitliliği nedeniyle karmaşıktır. Bu nedenle teslimat tarihleri, kapasite, sipariş kabulü ve kaynak dengesi etkin bir şekilde yönetilmelidir. Her siparişin kendi teslim tarihi olması ve siparişin tamamlanması için bir dizi faaliyetin gerekli olması nedeniyle üretim planlaması için proje planlama yaklaşımı uygun görülmektedir (Ghiyasinasab ve ark. 2021).

Bu üretim türünde, çok karmaşık bir özel ürünün üretimi bir projenin yürütülmesi olarak modellenebilir. Farklı ürünlerin üretimi için aynı anda yürütülen farklı projeler aynı üretim kaynakları (makineler, işçiler vb.) için rekabet edilebilirler.

Önerilen modellerin çoğu, projedeki her aktivitenin önceden belirlenmiş bir şekilde gerçekleştirilmesi gerektiğini varsaymaktadır. Bu, kaynakların faaliyetlere sabit bir oranda uygulandığı ve faaliyetlerin süresinin sabit olduğu anlamına gelir.

Literatürde kaynak kısıtlı çizelgeleme probleminin değişken yoğunluk formülü, değişken kaynakların kullanımı ve bu faaliyetlerin performans ilişkisini ele almak için önerilmiştir. Bu formül, her bir zaman periyodunda yürütülen bir aktiviteye ayrılan çabayı tanımlamak için kullanılan bir yoğunluk değişkeninin girilmesine dayanmaktadır. Kaynaklar sürekli dağıtılabilir olarak kabul edilir ve faaliyetleri tamamlamak için zaman içinde değişebilecek miktarlarda kullanılır (Kis 2005). Örneğin, üretim hattı için bazı cihazların

kablolanması ve bu cihazın durumu nedeniyle günde 15 adam-saat tahsis edilebileceğini varsayalım. Faaliyet 110 adam-saat gerektiriyorsa ve atölye kapasitesi şimdi 10 adam-saat ve 5 gün sonra 15 adam-saat olabiliyorsa, proje yöneticisi ilk 5 gün için 10 adam-saat ve altıncı günden itibaren 15 adam-saat ayırarak işi 9 günde tamamlayabilir. Değişken yoğunluk formülasyonu kullanılmadan, 15 adam-saat mevcut olana kadar bekleme gerekmektedir. Aktivite altıncı güne kadar, yeterli kaynak mevcut olmadığı için başlayamayacaktır ve altıncı gün başladıktan sonra kablolamanın bitirilmesine kadar 15 adam-saat her gün tahsis edilmesi gerekmektedir. Ayrıca, değişken yoğunluk formülasyonu kullanırken bazen aktivite başladıktan sonra işi durdurmak ve daha sonra başka bir dönemde tekrar tamamlamak mümkündür.

Literatürde değişken yoğunluk formülasyonu ve 4 farklı öncüllük ilişkisi kullanılarak üretim planlaması için kesin çözüm yaklaşımı geliştirilmiştir (Alfieri ve ark. 2011). Söz konusu çalışmada amaç fonksiyonu, faaliyetlerin tamamlanma süresini en aza indirmeye odaklanmıştır. Ancak, model NP zor olduğu için orta ve büyük ölçekli projelerde kesin çözüm bulunamamaktadır.

Bu çalışmada ise literatürden alınan söz konusu problemin amaç fonksiyonuna kaynak dengeleme de eklenmiştir. Bu şekilde geliştirilen modelin amacı, kaynakları dengelerken proje tamamlanma süresini en aza indirmektir. Ayrıca, yeni bir kromozom yapısı ile orta ve büyük boyutlu problemler için bir genetik algoritma önerilmiştir.

1.2. Tezin İçeriği

Tez üç ana bölümden oluşmaktadır. Kaynak araştırması bölümünde, seçilen konunun temelleri ve benzer özelliklere sahip problemler üzerine yapılan çalışmalar 4 farklı grupta özetlenmiştir: 1. Kaynak Kısıtlı Proje Çizelgeleme, 2. Çizelgeleme ve Değişken Yoğunluk Formülasyonu, 3. Kaynak Dengeleme, 4. Kaynak Kısıtlı Proje Çizelgelemesi için Sezgisel Algoritmalar.

Materyal bölümünde, problemler ve hedefler ayrıntılı olarak tanımlandıktan sonra, çözümün temelini oluşturacak yöntemler genel hatlarıyla tanıtılmaktadır. Yöntemler bölümünde geliştirilen matematiksel model, önerilen genetik algoritma ve parametre

optimizasyonu anlatılmaktadır. Bulgular ve Tartışma bölümünde uygulamada kullanılan veriler ve elde edilen sonuçlar sunulmuş ve analiz edilmiştir.

2. KAYNAK ARAŞTIRMASI

Bu bölüm dört alt başlık altında incelenmektedir: Kaynak kısıtlı proje çizelgeleme, değişken yoğunluklu formülasyon, kaynak kullanımının dengelemesi, kaynak kısıtlı proje çizelgeleme için sezgisel algoritmalar.

2.1. Kaynak Kısıtlı Proje Çizelgeleme

Kaynak kısıtlı proje çizelgeleme problemi (KKPÇP), projenin toplam süresini en aza indirirken, öncüllük kısıtlamaları ve kaynak kısıtlamaları ile ilişkili bir dizi faaliyet veya görevi ifade eden en genel çizelgeleme problemlerinden biridir (Liess ve Michelon 2008).

Üretim planlaması için proje çizelgeleme yaklaşımlarının kullanımı bilimsel kaynaklarda sıklıkla ele alınmıştır (Kleiny 2000; Vanczal ve Kis, 2003).

Karmaşık ve son derece özelleştirilmiş ürünlerin üretimi, sipariş üzerine üretim veya siparişe göre mühendislik sistemlerinde, her bir öğe genellikle belirli bir müşteri için özel olarak tasarlanmış ve kendi özelliklerine sahiptir. Her ürünün tasarımı, üretimi ve müşteriye teslimatı, bir projenin tamamlanması için kolayca modellenebilecek faaliyetlerdir. Üretim planlamasında, proje çizelgeleme yaklaşımları, gelecekteki faaliyetlerin yeterli bir görünümünü sağlamak için genellikle orta veya uzun bir zaman ufkunu ifade etmektedir (Alfieri ve ark. 2012).

Farklı üretim işlemleri toplu faaliyetler olarak gruplandırılır. Makineler ve işçiler, üretim kaynaklarını oluşturmak için birlikte değerlendirilir. Toplu faaliyetler genellikle tüm üretim aşamalarını temsil eder ve süreleri önemli sayılır. Ayrıca, ayrıntılı bir üretim süreci bilinmediğinde veya üretim planlama düzeyinde tanımlanması zor olduğunda, toplam düzeyde planlama yapmak kaçınılmaz bir seçimdir (Alfieri ve ark. 2011). Ulusoy (2015) üretim planlaması için proje çizelgeleme yaklaşımını kullanarak, faaliyetler arasındaki öncüllük ilişkilerini zaman bazında belirlemiştir.

Literatürde yaygın olarak tek bir öncüllük ilişkisi kullanılmaktadır (Kis 2005). Tek bir üretim operasyonu düşünüldüğünde, kaynak kısıtlamalarını temsil eden bir "bitişiyle başla" öncüllük ilişkisi tanımlamak kolaydır. Ancak operasyonlar toplu aktiviteler olarak gruplandırıldığında, "bitişiyle başla" öncüllük ilişkileri artık gerçek üretim sürecini doğru

şekilde temsil etmeyebilir. Bu durumda, öncüllük ilişkilerini daha doğru bir şekilde modellemeye yönelik ortak bir yaklaşım, faaliyetler arasında belirli bir miktarda örtüşmeye izin veren, genelleştirilmiş öncelik ilişkilerini kullanmak olabilir (Riane ve ark. 1998; Kis 2005).

Panwalkar ve Koulamas (2014) üretim planlaması için proje çizelgeleme yaklaşımını kullandı ve ayrıca iki ardışık iş arasındaki zaman gecikmelerini tanımladı. Tanımlanan zaman gecikmeleri minimum, maksimum veya tam olabilmektedir.

2.2. Çizelgeleme ve Değişken Yoğunluklu Formülasyon

Çoğu literatürde, kısıtlı kaynak proje çizelgeleme probleminde, her bir faaliyeti gerçekleştirirken sabit faaliyet süreleri ve sabit bir kaynak kullanım oranı öngörülmektedir (Brucker ve ark. 1998; De Reyck ve Herroelen 1998).

Literatürde, değişken yoğunluk formülasyonu, her bir faaliyetin dönem içinde değişebileceği ve kaynak kullanım yüzdesi ile orantılı olduğu varsayımıyla tanımlanmıştır. Bu formülasyonda, her bir dönemde her bir faaliyetin tamamlanma yüzdesini tanımlamak için bir yoğunluk değişkeni eklenmiştir. Değerlendirilen kaynaklar sürekli olarak bölünebilir ve gereken dönemlerde faaliyetlere tahsis edilebilir (Kis 2005).

NP_zor bir proje planlamada, her aktivite belirli bir miktarda yenilenebilir kaynak gerektirir ve faaliyetlerin her periyodundaki tamamlanma oranı, tahsis edilen kaynak ihtiyacının yüzdesine bağlıdır (Hung ve Leachman 1996).

Shtub et al. (1996) ve Hung ve Leachman (1996), aynı kaynakları birkaç rakip projeye tahsis etmek için değişken yoğunluk formülasyonu ile proje faaliyetlerini yürütme fikrini kullanmış olup, makalede yol yapım projesi örnek olarak verilmektedir.

Kis (2005), her bir aktivitenin değişken kaynak kullanım yoğunluğuna bağlı olarak zaman içinde değişebildiği, kaynak kısıtlı proje çizelgeleme problemi üzerinde çalışmıştır. Problemi bir karışık tamsayı-doğrusal programlama modeli olarak formüle ederek çözümün NP-zor olduğunu kanıtlamıştır. Optimal çözümleri bulmak için bir dal-kesme algoritması önermektedir. Bianco ve Caramia (2011), değişken yoğunluk formülasyonu

ve öncüllük kısıtlamaları ile minimum proje süresini elde etmek için kaynak kısıtlı proje çizelgeleme problemini modellemiştir.

Alfieri ve ark. (2011), Kis (2005) tarafından önerilen farklı öncüllük ilişkilerini geliştirdiği modele ekleyerek, toplam dört farklı öncül ardıllık kısıtı içeren değişken yoğunlukla modellenen bir çizelgeleme problemi geliştirmişlerdir. Önerilen modelde her faaliyetin başlaması, bitmesi veya belirli bir yüzdesinin tamamlanması, öncül veya ardıl faaliyetin tamamlanma oranına bağlıdır.

2.3. Kaynak Kullanımının Dengelenmesi

Proje süresinin (C_{max}) minimizasyonu, kaynak kısıtlı proje planlama probleminde ve varyantlarında yaygın olarak ele alınan bir hedeftir (Yang ve ark. 2018). Kaynakların optimizasyonu ve iş dağılımı, başarı elde etmek ve zaman kaybetmemek için kritik kilit noktalardır. Birçok yazar, kaynak israfını en aza indirmeyi amaçlayan tasarım sürecinden türetilen uygun zamanlama yaklaşımını ve algoritmayı araştırmaktadır. Kaynak yönetimi, programlamanın önemli bir yönünü oluştururken (Damci ve ark. 2013), kaynak dengeleme, proje tamamlanma sırasında kaynak kullanımı dalgalanmalarını en aza indirmeyi amaçlamaktadır (Benjaoran ve ark. 2015; Li ve ark. 2018).

Proje süresini en aza indirmek, kaynak kısıtlı proje çizelgeleme problemlerinde yaygın olarak ele alınan bir hedef olsa da (Chen ve ark. 2017) proje yöneticilerinin genellikle zaman içinde kıt ve pahalı yenilenebilir kaynakların kullanımını dengelemesi gerekir. Profesyonel hizmet sektöründe işgücünün dengeli kullanımı, sık işe alım ve işten çıkarmalardan kaçınmak ve çalışanların moralini yüksek tutmak için çok önemlidir (Ponz-Tienda ve ark. 2017).

Diğer birçok proje çizelgeleme problemi gibi, kaynak dengeleme de sipariş üzerine üretim ve sipariş üzerine mühendislik sistemleri gibi üretim ortamında yaygın olarak uygulanmaktadır (Márkus ve ark. 2003; Neumann ve ark. 2006). Müşteri siparişleri tarafından yönlendirilen üretim planlamasında, üretim zaman çizelgesi, bitmiş ürün ve bileşenlerinin miktarı ile bir ana üretim çizelgesi oluşturulmalıdır. Sipariş üzerine mühendislik üretiminde kaynak dengelemesi uygulaması, 1960'lardan beri çalışılmaktadır. Kaynak dengeleme yaklaşımlarına dayalı olarak sipariş üzerine

mühendislik üretim ortamlarında insan gücünün kullanımını dengelemektedir (Volling ve ark. 2013).

2.4. Kaynak Kısıtlı Proje Çizelgeleme için Sezgisel Algoritmalar

Çizelgeleme, sıralamadan farklı olarak faaliyetlerin hangi sırada gerçekleştirilecekleri bilgisi dışında bu faaliyetlerin ne zaman başlayıp ne zaman biteceği bilgisini de içerir. Çizelgelenen faaliyetlerinin kullandığı kaynaklar kısıtlı olabildiği gibi, faaliyetler arasında öncüllük ilişkileri de olabilir. Kesin çözüm yöntemleri ile çözülebilen problemlerin sınırlı ve gerekli hesaplama süresinin uzun olması, araştırmacıları daha büyük projeler için optimal bir çözüm olmasa da hızlı bir şekilde çözüme ulaşan yöntemler üzerinde çalışmaya yöneltmiştir. Sezgisel yöntemlerin sonuçları da kesin çözüm yöntemleri için bir üst veya alt sınır oluşturması nedeniyle önemlidir (Ulusoy 2015) .

Genetik Algoritma, Tavlama Benzetimi ve Tabu Arama gibi birçok meta-sezgisel yaklaşımı, zor hibrit optimizasyon problemlerini çözmek için kullanılır. Meta-sezgisel algoritmalar bir başlangıç çözümü oluşturur ve mevcut çözümlerden yeni çözümler üretmek için farklı operatörler kullanır (Tritschler et al. 2017).

Kaynak kısıtlı proje çizelgeleme problemi, gerçek bir projedeki çok sayıda faaliyet ve karmaşık öncüllük ilişkileri nedeniyle NP-zor bir problemdir. Bu nedenle çözüm yöntemleri öncelikle sezgisel algoritmaların kullanılmasını öngörür. Son yirmi yılda, kaynak kısıtlı proje çizelgeleme problemini çözmek için meta-sezgisel yöntemlerden, genel olarak farklı meta-sezgisel stratejilere dayalı hibrit yöntemlere geçme eğilimi görülmektedir. Literatürde, Tavlama Benzetimi, Tabu Arama, Arı Kolonisi Optimizasyonu, Genetik Algoritmalar, Parçacık Sürü Optimizasyonu, Dağınık arama ve Karınca Kolonisi Optimizasyonu diğer meta-sezgisel yöntemler olarak sıklıkla kullanılmıştır (Pellerin ve ark. 2020).

Meta-sezgisel hibridizasyonunun en popüler yollarından biri, popülasyon tabanlı meta-sezgisellerin, yerel arama meta-sezgiselleriyle birleştirilmesidir. Bu kategori içinde iki tür hibrit ayırt edilebilir: tek çözümlü ve popülasyon tabanlı hibritler. Literatürde genetik algoritmalarda kullanılan çapraz geçiş işlemlerinin, yüksek kaliteli çözümün aralığını

genişletmek için tek bir çözüm meta sezgiseline gömülü olduğu bir hibrit önerilmiştir (Zamani 2011). Aslında literatürde, popülasyon tabanlı yöntemlerde genetik algoritmalar (Quintanilla ve ark. 2012), arı kolonisi algoritmaları (Ziarati ve ark. 2011) ve parçacık sürü optimizasyonu (Jia ve Seo 2013), yerel arama yöntemlerinin kullanılması kaynak kısıtlı proje çizelgeleme problemi için çok yaygındır.

Literatürde, Genetik Algoritma hesaplama süresini azaltmak için Tabu Arama ve Tavlama Benzetiminin popülasyonun bir kısmına uygulaması önerilmiştir. (Thammano ve Phu-ang 2012).

Li ve ark. (2018), genelleştirilmiş öncüllük ilişkileri ile kaynak dengeleme problemi için yeni bir genetik algoritma geliştirmiştir. Andrade ve ark. (2019), genel akış süresini en aza indirmek için genetik algoritmalar ve açgözlü yinelenen arama yöntemleri sunmuşlardır. Kazemi ve Davari-Ardakani (2020), proje tamamlama süresini ve proje yürütmenin toplam maliyetini en aza indirmeyi amaçlayan sezgisel model ile kaynak dengeleme ve malzeme siparişinin eşzamanlı planlanmasını gerçekleştirmiştir.

Snauwaert ve Vanhoucke (2021), kısıtlı kaynaklarla proje çizelgeleme probleminin çok yetenekli gelişimini verimli bir şekilde çözmek için bir genetik algoritma geliştirmiştir.

3. MATERYAL VE YÖNTEM

Bu bölümde, kaynak kısıtlı proje çizelgeleme problemi için yoğunluk değişkeni ve dört farklı öncüllük koşulu kullanılarak, amaç fonksiyonu C_{max} ve aynı zaman kaynak dengeleme olan bir matematiksel model ve bu matematiksel model için yeni bir genetik algoritmanın tasarımına yer verilmiştir.

3.1. Materyal

Bu bölümde öncelikle literatürdeki C_{max}'ı en küçükleyecek şekilde ve kaynak kısıtı altında proje çizelgeleme problemleri için yoğunluk değişkeni ve dört farklı öncül koşul kullanan model detaylandırılmış ve kodlanmıştır. Bir sonraki adımda, kaynak dengeleme hedefi ile bu modelin amaç fonksiyonu değiştirilmiş ve ardından bu model için yeni bir Genetik Algoritma tanıtılmıştır.

3.1.1. Değişken Yoğunluk Formülasyonu

Değişken Yoğunluk formülasyonunun farklı öncüllük ilişkilerini kullanarak kaynak kısıtlı proje çizelgeleme probleminin matematiksel modeli oluşturulabilir. Bu model aşağıdaki özelliklere sahiptir (Alfieri ve ark. 2011):

- Faaliyetlerin tamamlanması bir sürekli değişken ile izlenir. Bu değişken (x_{jt}) her dönemde (t) tamamlanan faaliyetin (j) yüzdesini temsil etmektedir. Bir faaliyeti tamamlamak için gereken sürenin önceden tanımlanmadığı kabul edilmektedir. Bu süre, her dönem yoğunluk değişkenlerinin değerine bağlı olup, bir minimum ve maksimum süre arasında değişebilmektedir. Minimum ve maksimum süreler, her dönemde her bir faaliyette tahsis edilmesi mümkün olan minimum ve maksimum kaynak miktarlarına bağlıdır.
- Faaliyetin sürdürülmesi için yoğunluk değişkeninin değerine göre farklı öncüllük ilişkileri tanımlanmaktadır. Bu öncüllük ilişkileri aşağıda belirlenmiştir:
- Başlama için %Tamamlanma ilişkisi, (*%Completed-to-Start* (CtS)): Ardıl (j) faaliyetin başlayabilmesi için öncül faaliyetin(i'nin) en az q_{ij} yüzdesi kadar tamamlanması gerekir (Şekil 3.1(a)).

- %Tamamlanmaya bağlı başlama ilişkisi (*Start-to-%Completed* (StC)): öncül faaliyet (i) başlamadan, ardıl faaliyetin (j' nin) tamamlanma yüzdesi en fazla g_{ij} kadar olabilir (Şekil 3.1(b)).
- Bitirme için %Tamamlanma ilişkisi (*%Completed-to-Finish* (CtF)): Ardıl (j) faaliyetin tamamlanabilmesi için öncül faaliyetin(i'nin) en az q_{ij} yüzdesi kadar tamamlanması gerekir (Şekil 3.1(c)).
- %Tamamlanmaya bağlı bitirme ilişkisi (*Finish-to-%Completed* (FtC)) : öncül faaliyet (i) tamamlanmadan, ardıl faaliyetin (j' nin) tamamlanma yüzdesi en fazla g_{ij} kadar olabilir (Şekil 3.1(d)).

Şekil 3.1. Ön şart tanımları (Alfieri ve ark. 2011)

Değişken yoğunluk formülünde, her bir periyotta her aktiviteyi yürütme için gösterilen çabaları tanımlamak üzere bir yoğunluk değişkeni tanımlanır (Hung ve Leachman 1996; Kis 2005).

Sürekli bölünebilen kaynaklar, faaliyetlere her dönem farklı miktarlarda tahsis edilir. Yoğunluk değişkeninin ($x_{jt} \in \mathbb{R}$) gerçel sayılardan değer alması nedeniyle, faaliyetler için sonsuz sayıda yürütme moduna izin verilir. Bu nedenle her bir aktivitenin tamamlanma süresi sabit değildir, ancak her periyottaki yoğunluk değişkenin miktarına bağlıdır (Tolio ve Urgo 2007; Kis 2005).

Kis (2005) tarafından önerilen Besleme Öncüllük İlişkisi kavramı, tam olarak yukarıda açıklanan zorlukların üstesinden gelmek için geliştirilmiştir. Bununla birlikte, Kis (2005), bir faaliyeti ancak önceki faaliyetin belirli bir yüzdesi tamamlandıktan sonra başlayacak şekilde sınırlamak için tek bir besleme öncüllük ilişkisi türü tanımlamıştır.

Alfieri ve ark. (2011)'in katkısı ise, faaliyetlerin yürütme modunun zaman içinde değişebileceği tüm genelleştirilmiş öncüllük ilişki türlerini göz önünde bulundurmak için diğer üç tür öncüllük ilişkisinin tanımına dayanır.

Önerilen öncüllük ilişkileri,

i. bir faaliyetin, ardıl faaliyetinin belirli bir yüzdesi tamamlanmadan önce başlamasını,

ii. bir faaliyetin, öncül faaliyetinin belirli bir yüzdesi tamamlandıktan sonra başlamasını (Kis 2005),

iii. bir faaliyetin ardıl faaliyetinin belirli bir yüzdesi tamamlanmadan önce bitmesini,

iv. bir faaliyetin öncül faaliyetinin belirli bir yüzdesi tamamlandıktan sonra bitmesini

tanımlar.

Bu ilişkilere besleme öncüllükleri olarak atıfta bulunulacaktır. Besleme öncüllükleri, hem başlangıç ve bitiş zamanlarını, hem de yürütmelerinin ilerlemesini göz önünde bulundurarak, faaliyet çiftleri arasındaki öncüllük ilişkilerinin rolüne farklı bir bakış açısı getirmektedir.

Alfieri et al. (2011)'a göre problemin matematiksel modeli aşağıda verilmiştir:

Parametreler

J	Faaliyetlerin kümesi
T	Proje dönemlerinin kümesi
K	Kaynakların kümesi
τ	Öncül ardıl ilişkiler kümesi
$\tau_1 \in \tau$	Öncüllük ilişkilerinin alt kümesi (*%Completed-to-Start* (CtS))
$\tau_2 \in \tau$	Öncüllük ilişkilerinin alt kümesi (*Start-to-%Completed* (StC))
$\tau_3 \in \tau$	Öncüllük ilişkilerinin alt kümesi (*%Completed-to-Finish* (CtF))
$\tau_4 \in \tau$	Öncüllük ilişkilerinin alt kümesi (*Finish-to-%Completed* (FtC))
B_j	j faaliyeti tarafından bir dönemde yapılabilecek maksimum iş yüzdesi

b_j j faaliyeti tarafından bir dönemde yapılabilecek minimum iş yüzdesi

i_p Öncül faaliyet $p \in \tau$

j_p Ardıl faaliyet $p \in \tau$

q_p j_p faaliyetinin belirli bir dönemde başlamasına veya bitmesine izin vermek için i_p faaliyeti için gereken tamamlanma yüzdesi, $p \in (\tau_1, \tau_3)$.

g_p j_p faaliyetinin belirli bir dönemde başlamış veya bitmiş olması halinde, i_p faaliyetinde elde edilebilecek iş yüzdesi, $p \in (\tau_2, \tau_4)$.

r_j j faaliyetinin başlayabilecek zamanı

d_j j faaliyetinin son tarihi

Q_{jk} j faaliyetini tamamen işlemek için gereken k kaynağın miktarı

R_{kt} t döneminde mevcut olan k kaynağının toplam miktarı

Değişkenler

$C_{\max}$ Maksimum tamamlanma zamanı, $C_{\max} \in \mathbb{R}^+$

f_{jt} $\begin{cases} 1, & j \text{ faaliyeti } t \text{ döneminde biterse} \\ 0, & d.d. \end{cases}$

s_{jt} $\begin{cases} 1, & j \text{ faaliyeti } t \text{ döneminde başlarsa} \\ 0, & d.d. \end{cases}$

x_{jt} t döneminde j faaliyetinin tamamlanma yüzdesini temsil eden sürekli pozitif değişken, $x_{jt} \in \mathbb{R}^+$

η_{jt} $\begin{cases} 1, & j \text{ faaliyeti t döneminde işlenirse} \\ 0, & d.d. \end{cases}$

Amaç fonksiyonu:

$$\text{Min } C_{\max} \tag{3.1}$$

Kısıtlar:

$$C_{\max} \geq \sum_{t=r_j}^{d_j} t \cdot f_{jt}, \quad \forall j \tag{3.2}$$

$$\sum_{t=r_j}^{d_j} s_{jt} = 1, \quad \forall j \tag{3.3}$$

$$\sum_{t=r_j}^{d_j} f_{jt} = 1, \quad \forall j \tag{3.4}$$

$$\sum_{t=r_j}^{d_j} t \cdot s_{jt} \geq r_j, \quad \forall j \tag{3.5}$$

$$\sum_{t=r_j}^{d_j} x_{jt} = 1, \quad \forall j \tag{3.6}$$

$$x_{jt} \leq B_j \cdot \eta_{jt}, \quad \forall j, t \tag{3.7}$$

$$x_{jt} \geq b_j \cdot \eta_{jt}, \quad \forall j, t \tag{3.8}$$

$$s_{jt} \leq \eta_{jt}, \quad \forall j, t \tag{3.9}$$

$$f_{jt} \leq \eta_{jt}, \quad \forall j, t \tag{3.10}$$

$$f_{jt} \leq \sum_{h=r_j}^{t} x_{jh}, \quad \forall j, t \tag{3.11}$$

$$\sum_{h=r_j}^{t} s_{jh} \geq x_{jt}, \quad \forall j, t \tag{3.12}$$

$$\sum_{j \in J} Q_{jk} \cdot x_{jt} \leq R_{kt}, \quad \forall k, t \tag{3.13}$$

$$s_{jt} \leq \sum_{h=1}^{t-1} x_{ih} - q_p + 1, \quad \forall p \in \tau_1, i = i_p, j = j_p, \forall t \tag{3.14}$$

$$\left(1 - \sum_{h=1}^{t} f_{jt}\right) \geq q_p - \sum_{h=1}^{t-1} x_{ih}, \quad \forall p \in \tau_3, i = i_p, j = j_p, \forall t \tag{3.15}$$

$$\sum_{h=1}^{t} x_{jh} \leq g_p + (1 - g_p) \sum_{h=1}^{t-1} s_{ih} \ , \quad \forall\, p \in \tau_2, i = i_p, j = j_p \ , \forall t \tag{3.16}$$

$$\sum_{h=1}^{t} x_{jh} \leq g_p + (1 - g_p) \sum_{h=1}^{t-1} f_{ih} \ , \quad \forall\, p \in \tau_4, i = i_p, j = j_p \ , \forall t \tag{3.17}$$

Kısıt (3.2), maksimum proje süresini en son tamamlanan faaliyetin tamamlanma zamanı olarak tanımlamıştır. Her aktivite mutlaka bir *t* anında başlar (3.3) ve biter (3.4). Ayrıca, hiçbir faaliyet başlangıç zamanından (3.5) önce başlayamaz ve başlangıç zamanı ile bitiş zamanı arasındaki sürede her bir faaliyetin tam olarak işlenmesi gerekmektedir (3.6).

Bir dönemde *j* faaliyetinin bir yüzdesi yapılırsa, bu oran en az b_j (3.8) ve en çok B_j (3.7) olabilir. Bu kısıtlamalar temel olarak teknolojik ve/veya ekonomik nedenlerden kaynaklanmaktadır; çünkü faaliyetin türüne ve gerekli kaynaklara bağlı olarak, bir faaliyeti belirli bir miktardan daha fazla veya daha az işlemek mümkün veya ekonomik olmayabilir. Kısıtlar (3.9), ve (3.10), bir faaliyetin işlenmediği bir zaman döneminde ($\eta = 0$) etkin bir şekilde başlamasını veya bitmesini (*s* ve *f*, 1'e eşittir) yasaklar. (3.11) ve (3.12), bir aktivitenin tamamlanmadığı takdirde bitemeyeceğini, başlatılmadığı takdirde de, işlenemeyecğini göstermektedir.

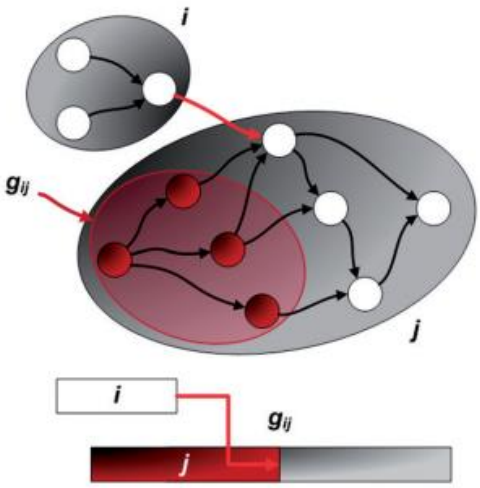
Faaliyet işleme tarafından kullanılan toplam kaynak miktarı, her kaynak için ve her dönemde mevcut olan kaynağın toplam miktarını aşmamalıdır. Bu, kısıtlama (3.13) ile sağlanmaktadır.

Kısıtlar (3.14) - (3.17) iki aktivite arasındaki besleme öncüllüklerini temsil eder. Kısıt (3.14), %*Completed-to-Start* ilişkilerini temsil eder. Belirli bir t döneminde tüm %*Completed-to-Start* öncüllük ilişkileri ($p \in \tau_1$) çerçevesinde, öncül faaliyet i_p kümülatif olarak en az yüzde q_p olmak üzere işlenene kadar ardıl faaliyet j_p başlatılamaz ($s_{jt} < 1$).

Kısıt (3.16), *Start-to-%Completed* önceliklerini temsil eder. Belirli bir dönemde (t), tüm "*Start-to-%Completed*" öncüllük ilişkisi olan aktivitelerin $(j = j_p \text{ ve } p \in \tau_2)$

tamamlanma oranı $\left(\sum_{h=1}^{t} x_{jh}\right)$, g_p'den daha büyük olması için öncül i faaliyetinin başlaması gerekmektedir; yani $\sum_{h=1}^{t-1} s_{ih} = 1$ olmalıdır.

Kısıtlar (3.15) ve (3.17), diğer ilişki türleri için benzer şekilde çalışır (önceki açıklamalarda sadece başlangıç ile bitişi değiştirerek). Kısıt (3.15) *%Completed-to-Finish* ilişkilerini temsil ederken, kısıt (3.17) *Finish-to-%Completed* ilişkilerini temsil etmektedir (Şekil 3.2).



Şekil 3.2. Toplu faaliyetlerde besleme önceliği (Alfieri ve ark. 2011)

3.1.2. Endüstriyel Uygulama Örneği

Bir işleme merkezi, palet veya parça taşıma için otomatik bir takım değiştirici ve ekipmanla entegre edilmiş bir CNC (Bilgisayar Sayısal Kontrollü) takım tezgahı olarak tanımlanabilir. Tipik olarak, çok eksenli bilgisayar kontrollü bir freze tezgahı ve farklı işlevler sağlayan bir dizi ek ekipmandan oluşur (örneğin, palet deposu, soğutma ve yağlama sağlayan ekipman, metal talaşların bertaraf için bir cihaz, yüksek derecede otomasyonu yönetmek için otomatik kontrolörler ve bilgisayarlar vb.). İşleme merkezi üreticileri, ürünleri için standart konfigürasyonlar sağlasa da müşteriler genellikle özel ihtiyaçlarına göre uyarlanmış modifikasyonlar ister. Bu, Avrupalı (ve özellikle İtalya) işleme merkezi üreticileri için yaygın bir uygulamadır (Schwindt ve Zimmermann 2015a).

Bir işleme merkezinin imalatı, tipik olarak proje çizelgeleme yaklaşımlarıyla ele alınan karmaşık, türünün tek örneği bir süreçtir. Özelleştirilmiş spesifikasyonlar tasarlandıktan sonra, önemli bir bileşen setinin üretimi, harici tedarikçilere atanırken, yalnızca kritik bileşenler için yüksek hassasiyetli üretim faaliyetleri dâhili olarak gerçekleştirilir.

Şekil 3.3. İşleme merkezi (Schwindt ve Zimmermann 2015a)

Diğer tüm bileşenler ve parçalar ana yapının etrafına monte edilir ve birbirine bağlanır. Nihai montaj (Şekil 3.3) test edilir ve ardından müşteriye teslim edilmek üzere kısmen demonte edilir.

Ayrıntılı üretim süreci, her biri belirli bir bileşen setinin montajı, kablolaması veya test edilmesiyle ilgilenen yüzlerce üretim faaliyeti göz önünde bulundurularak düşünülmüştür. Toplu faaliyetlerin tanımı, işleme merkezinin malzeme listesine göre yapılmıştır. Bileşenler fonksiyonel birimler halinde gruplandırılır ve her grup için bir seri üretim veya montaj faaliyeti tanımlanır. Toplu faaliyetler tanımlandıktan sonra, tipik bir işleme merkezinin üretimi sekiz ana aşamadan oluşur (Şekil 3 ve 4). Bunlar,

i. Yapı Hazırlığı. Montaj aşaması için makine merkezin yapısı hazırlanır. (A01)

ii. Palet Hazırlama. Paletler montaj aşamasına hazırlanır. (A02)

iii. Yapı Boyama. Makinenin merkezi yapısı boyanır. (A03)

iv. Otonom Bileşenlerin Birleştirilmesi. İşleme merkezine kurulacak özerk bileşenler, (İş mili kafası, makine masası, elektrik panosu, vb.) ayrı ayrı monte edilir. (A04)

v. Montaj. Makine merkezi yapısı montaj alanına yerleştirilir ve tüm bileşenler kurulur. (A05)

vi. Kablolama. Tüm kurulu bileşenler ve kontrol sistemi için elektrik bağlantısı sağlanır. (A06)

vii. Test aşaması. Ana işlevler, ana düzenlemelere ve dâhili standartlara göre test edilir. Makine merkezi doğruluğu, beyan edilen yeteneklerine ve müşterinin özelliklerine göre test edilir. (A07)

viii. Sökme ve Teslimat. İşleme merkezi kısmen demonte edilerek müşteriye teslim edilir. (A08)

Üretim sürecini doğru bir şekilde modellemek için öncüllük ilişkileri kullanılır. Montaj aşaması; elektrik kabini, iş mili kafası, çalışma masası gibi bağımsız bileşenlerin montajına ayrılmış birden fazla alt adımdan oluşur. Bu münferit bileşenlerin işleme merkezi yapısına montajı gerekir. Ancak montaj aşaması başladığında, fiilen tam olarak işlenmeleri gerekmez. Aksine, ayrıntılı üretim süreci göz önüne alındığında, ancak belirli bir dizi başka montaj işlemi tamamlandıktan sonra işleme merkezinin yapısına monte edilebilirler. Aynı zamanda, en geç işleme merkezi üzerlerine kurulmaya hazır hale gelmeden önce tamamlanmalıdır.

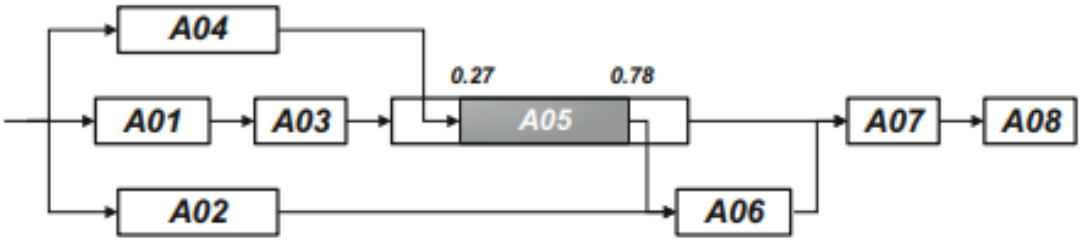
Bu gibi durumlarda, farklı bileşenlerin montajının en geç işleme merkezi montajının belirli bir yüzdesi gerçekleştirildikten sonra tamamlanmasına izin vermek için *Finish-to-%Completed* kısıtlaması kullanılabilir. Bu yüzde, dikkate alınan alt montaj henüz işleme merkezine kurulmaya hazır olmasa bile gerçekleştirilebilecek montaj faaliyetinin yüzdesini temsil eder.

Mekanik montaj ile kablolama arasındaki ilişkilere atıfta bulunarak benzer bir değerlendirme yapılabilir. Kablolamaya başlamak için tüm mekanik montaj aşamasının

tamamlanması beklenmemelidir. Kablolama, birbirine bağlanması gereken bileşenler kurulur kurulmaz başlayabilir. Bu durumda, montaj faaliyetinin belirli bir yüzdesi tamamlandıktan sonra kablolama faaliyetinin erken başlamasına izin verilmelidir.

Dolayısıyla, kablolama aşamasının birlikte kablolaması gereken bileşenler işleme merkezine monte edilir edilmez başlamasına izin vermek için %*Completed-to-Start* kısıtlaması kullanılabilir. Bu aşamalar esas olarak mavi yaka çalışanlar tarafından yönetilir. Söz konusu çalışanlar, özel becerilerine göre yedi farklı tipte gruplandırılmıştır ve her üretim aşaması yalnızca bir tür vasıflı işçi gerektirir.

Bir ekipteki işçiler, aynı toplam faaliyete ait farklı üretim operasyonlarında çalışabilirler ve bazıları aynı anda üretilen farklı işleme merkezlerinde çalışan farklı ekiplere aktarılabilir. Davranışları, değişken bir kaynak kullanımına izin veren değişken yoğunluk formülasyonu kullanılarak doğru bir şekilde modellenebilir. Gerçek endüstriyel ortamda, kaynak kullanılabilirliği, aynı anda üretimde olabilecek diğer siparişlerin gereksinimlerine bağlı olsa bile sabit kabul edilir (Şekil 3.4)



Şekil 3.4. İşleme merkezi üretiminin aşamaları (Schwindt ve Zimmermann 2015b)

3.1.3. Kaynak dengeleme

Kaynak dengeleme, proje yürütme sırasında kaynak kullanımındaki varyansı en aza indirgemeyi amaçlar. Kıt ve pahalı yenilenebilir kaynakların etkin kullanımını sağlamak proje çizelgelemesinde çok önemlidir ve sipariş üzerine üretim veya sipariş üzerine mühendislik sistemleri gibi üretim ortamlarına başarıyla uygulanmıştır. Gerçek hayat projelerinde, faaliyetler arasındaki karmaşık zaman bağımlılıklarını modellemek için genellikle genel zamansal ilişkilere ihtiyaç duyurmaktadır (Li ve ark. 2018).

Li ve ark. (2018), geliştirdikleri amaç fonksiyonu ile bir zaman çizelgesi üzerinden kaynak kullanımındaki varyansı enküçükleyecek bir matematiksel model oluşturmuşlardır. Bu modele ilişkin amaç fonksiyonu ve ilgili kısıt aşağıdaki gibi verilmiştir:

$$\text{Min} \sum_{k=1}^{K} \sum_{t=1}^{d} c_k \, {u_{kt}}^2 \tag{3.18}$$

$$\sum_{i \in ACT_t} r_{ik} = u_{kt}, \qquad \forall k = 1, \dots, K \text{ ve } t = 1, \dots, d \tag{3.19}$$

Modelde kullanılan parametreler:

K Kaynak tiplerinin sayısı,

ACT_t t dönemi içinde aktif faaliyetler kümesi,

d Projenin tamamlanma zamanı,

r_{ik} i faaliyetini tamamlamak için gereken yenilenebilir k kaynak miktarı,

c_k k kaynak tipi için birim maliyeti gösteren ağırlık.

Modelde kullanılan değişken:

u_{kt} k kaynağından t dönemi boyunca kullanım miktarı.

(3.18)'deki amaç fonksiyonu, varyans tanımına karşılık gelen karesi alınmış kaynak kullanım toplamını, kaynak tipine göre ağırlıklandırarak, en küçükleyen bir kaynak dengeleme performans ölçütüdür. c_k, k kaynak tipi için birim maliyeti gösteren bir ağırlıktır. Kısıt (3.19), t dönemindeki tüm aktif faaliyetlerin k kaynağına olan gereksinim toplamını t dönemindeki kaynak kullanım miktarı olarak, u_{kt} değerini hesaplamak için kullanılır.

3.1.4. Genetik Algoritma

Genetik Algoritma (GA), permütasyon tabanlı bir optimizasyon yapar ve olasılıklar üzerinden yakınsama kriterleri altında arama yapan bir fonksiyondur. Doğada

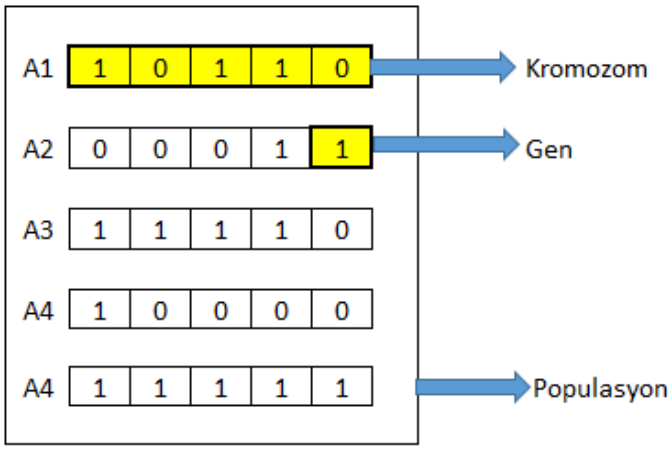
gözlemlenen evrimsel sürece benzer bir şekilde çalışan, arama ve eniyileme yöntemidir. Genetik algoritmaların uygulandığı problemlerin aralığı çok geniştir, genetik algoritmalar genellikle fonksiyon optimize ediciler olarak görülür. Bir genetik algoritmanın uygulanması, bir kromozom popülasyonu (genellikle rastgele) ile başlar. Daha sonra bu yapılar değerlendirilir ve hedef probleme daha iyi bir çözüm sağlayan bu kromozomlara, daha zayıf çözümlere sahip kromozomlardan daha iyi bir üretim şansı vermek için üreme fırsatları belirlenir.

Genetik algoritma, Charles Darwin'in doğal evrim teorisinden esinlenen bir sezgisel arama yöntemidir. Bu algoritma, bir sonraki neslin bireylerini üretmek üzere ebeveyn olarak en uygun bireylerin seçildiği doğal seçilim sürecini yansıtır. Genetik algoritma en genel hali ile beş aşamadan oluşmaktadır. Bu aşamalar ve aşamalara ilişkin açıklamalar aşağıda verilmiştir:

- İlk popülasyon
- Uygunluk fonksiyonu
- Seçim
- Çaprazlama
- Mutasyon

İlk popülasyon

Algoritmanın uygulaması popülasyon adı verilen bir grup birey oluşturma ile başlar. Her birey, çözmek istediğiniz soruna bir çözümdür. Bir birey, gen olarak bilinen belirli dizilişe sahip veri yapıları ile karakterize edilir. Genler, bir kromozom oluşturmak için bir dizi halinde birleştirilir (Şekil 3.5).

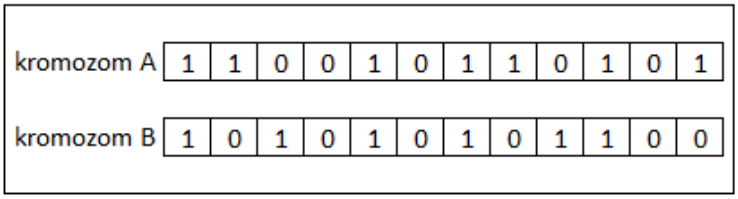


Şekil 3.5. Örnek popülasyon

Kromozom yapılarını oluşturma üzere belirli kodlama yöntemleri kullanılmaktadır. Bu kodlama yöntemleri:

İkili Kodlama:

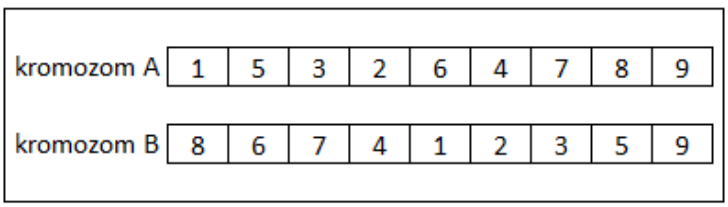
En yaygın kodlama yöntemi olup, kromozomlar, 1'ler ve 0'lar dizisidir ve kromozomdaki her konum, problemin belirli bir değişkenine ait çözümün parçasıdır (Şekil 3.6).



Şekil 3. 6. İkili Kodlama

Permütasyon Kodlaması:

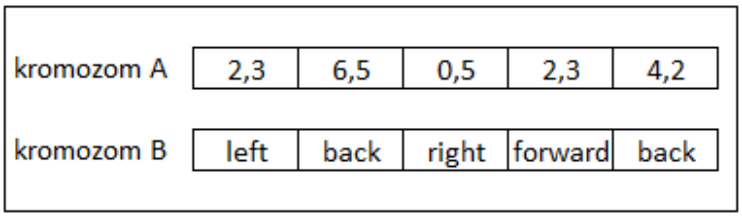
Gezgin Satıcı Problemi (TSP) gibi bir sıralama problemi için kullanışlıdır. TSP'de her kromozom, her biri ziyaret edilecek bir şehri temsil eden bir sayı dizisidir.



Şekil 3.7. Permütasyon Kodlaması

Değer Kodlaması:

Gerçek sayılar gibi karmaşık değerlerin kullanıldığı ve ikili kodlamanın yeterli olmadığı problemlerde kullanılır. Bu kromozomlar, bazı özel çaprazlama ve mutasyon tekniklerinin geliştirilmesi gereken problem tiplerine uygundur.



Şekil 3.8. Değer Kodlaması

Uygunluk fonksiyonu

Uygunluk Fonksiyonu, verilen bir çözümün istenen problemin optimum çözümüne ne kadar yakın olduğunu değerlendirir. Bir çözümün ne kadar uygun olduğunu belirler.

Genetik algoritmalarda, her çözüm genellikle kromozom olarak bilinen ikili bir sayı dizisi olarak temsil edilir. Bu çözümler, belirli bir sorun için en iyi çözüm kümesini bulmak için test edilmelidir. Bu nedenle, istenen çözümün genel özelliklerini karşılamaya ne kadar yakın olduğunu göstermek için her çözüme bir puan verilmesi gerekir. Bu puan, uygunluk fonksiyonunun teste uygulanmasıyla veya test edilen çözümden elde edilen sonuçlarla üretilir.

Her problemin kendi uygunluk fonksiyonu vardır. Kullanılması gereken uygunluk fonksiyonu verilen probleme bağlıdır. Genetik algoritmalar kullanarak bir problem formüle etme söz konusu olduğunda, verilen problem için bir uygunluk fonksiyonu bulmak en zor kısımdır.

Belirli bir problemde belirli bir fonksiyonun kullanılması gerektiğine dair kesin ve hızlı bir kural yoktur. Bununla birlikte, veri bilimcileri tarafından belirli sorun türleriyle ilgili olarak belirli işlevler benimsenmiştir.

Optimizasyon problemleri için, problem alanıyla ilgili bir dizi hesaplanmış parametrenin toplamı gibi temel fonksiyonlar uygunluk fonksiyonu olarak kullanılabilir.

Seçim Yöntemleri

Rulet tekerliği yöntemi

Rulet tekerliği seçiminde, bir sonraki neslin üremesi için bir bireyi seçme olasılığı, uygunluğu ile orantılıdır; uygunluk ne kadar iyi olursa, o bireyin seçilme şansı o kadar yüksek olur. Bu, bir Rulet çarkına benzer şekilde tasvir edilebilir. Genellikle, uygunluk değerlerine dayalı olarak olası seçimlerin her birine çarkın bir oranı atanır. Bu, bir seçimin uygunluğunu tüm seçimlerin toplam uygunluğuna bölerek ve böylece onları 1'e normalleştirerek başarılabilir. Ardından, rulet çarkının nasıl döndürüldüğüne benzer şekilde rastgele bir seçim yapılır. Bireysel i seçme olasılığı, $\boldsymbol{p_i} = \frac{\boldsymbol{f_i}}{\sum_{j=1}^{N} \boldsymbol{f_j}}$'ye eşittir; burada $\boldsymbol{f_i}$, i'nin uygunluğudur ve N, mevcut neslin boyutudur.

Sıra Seçimi

Sıra Seçimi de negatif uygunluk değerleriyle çalışır ve çoğunlukla popülasyondaki bireylerin uygunluk değerleri çok yakın olduğunda kullanılır. Bu, her bireyin pastanın neredeyse eşit bir payına sahip olmasına yol açar ve dolayısıyla her bireyin birbirine göre ne kadar uygun olursa olsun, ebeveyn olarak seçilme olasılığı yaklaşık olarak aynıdır. Bu da uygun bireylere yönelik seçim baskısında bir kayba yol açarak, GA'nın bu gibi durumlarda zayıf ebeveyn seçimleri yapmasına neden olur.

Turnuva seçim yöntemi

Turnuva seçimi, bireylerden oluşan bir popülasyondan bir bireyi seçme yöntemidir. Turnuva seçimi, birkaç birey arasından birkaç "turnuva" (veya "kromozom") popülasyonundan rastgele seçilir. Her turnuvanın galibi geçişler için seçilir. Seçim olasılığı, katılımcı seçim havuzunun boyutuna bağlıdır. Turnuva boyutu değiştirilerek bir kromozomun turnuvaya katılma olasılığı kolayca ayarlanabilir, bunun nedeni, turnuva boyutu daha büyükse daha zayıf bireylerin seçilme şansının daha düşük olmasıdır.

Elitist yöntem

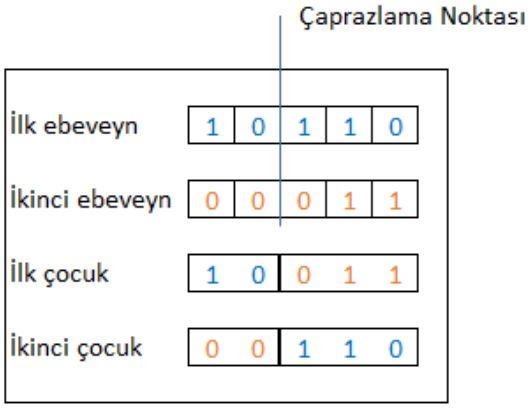
Bu yaklaşım ile kötü nesili daha çabuk yok edebilmek için, kötü sonucu iyi sonuç ile değiştirerek popülasyonun iyileşmesi sağlanır. En iyi sonuç saklanarak yeni nesile aktarılması sağlanır. Genellikle daha iyi parametreler elde etmek için kısmi çoğaltma stratejileri kullanılır. Bunlardan biri, son nesilden en iyi bireylerin küçük bir kısmının (herhangi bir değişiklik olmaksızın) bir sonrakine aktarıldığı elitizmdir.

Çaprazlama

Çaprazlama, bir veya daha fazla kromozomun programlanmasını bir nesilden diğerine değiştirmek için kullanılan bir genetik operatördür. Üstün yavrular üretmek için çiftleşme havuzundan çaprazlama için iki dizi rastgele seçilir. Seçilen yöntem, Kodlama Yöntemine bağlıdır. Farklı çaprazlama yöntemleri aşağıdaki gibi açıklanmıştır.

Tek noktadan çaprazlama

Rastgele alınan bir çaprazlama noktasına göre, iki ebeveyn kromozom bölünür; birinci çocuk için ilk bölen kısım birinci kromozomdan, kalan kısım ise diğerinden alınır. Diğer çocuk için tersi yapılır (Şekil 3.9).

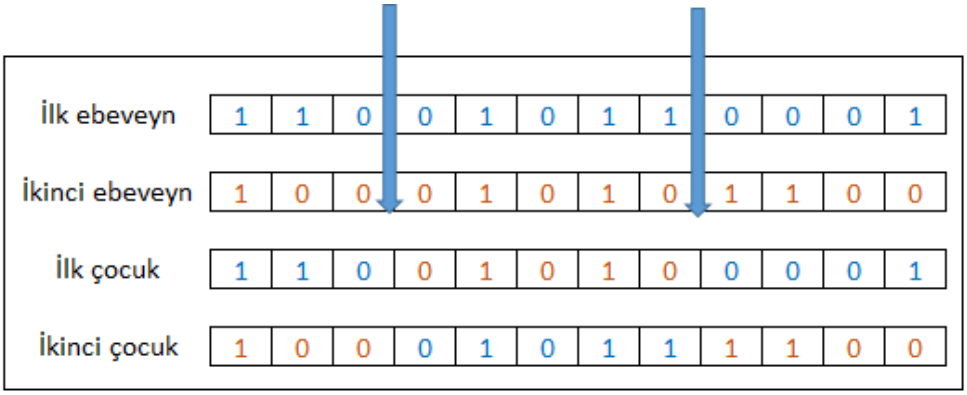


Şekil 3.9. Tek noktadan çaprazlama

Çift noktadan çaprazlama

Bireysel kromozomlar üzerinde iki rastgele nokta seçilir, iki nokta arasındaki bitler ebeveynler arasında değiştirilir.

İki noktalı çaprazlama, farklı çaprazlama noktalarıyla iki tek noktalı çaprazlama gerçekleştirmeye eşdeğerdir. Bu strateji, herhangi bir pozitif tamsayı k için k-noktası çaprazlaması olarak genelleştirilebilir (Şekil 3.10).



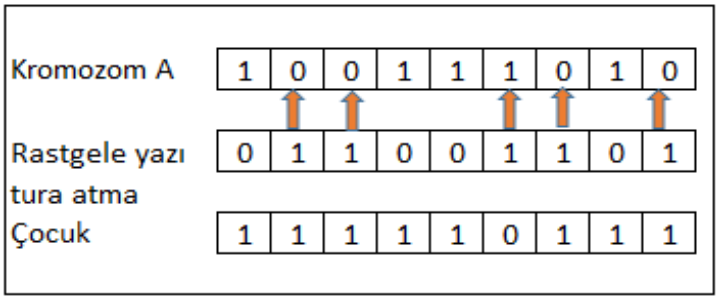
Şekil 3.10. Çift noktadan çaprazlama

Üniform Çaprazlama

Üniform çaprazlamada, tipik olarak, kromozom parçalara ayrılmaz ve her bir gen ayrı ayrı ele alır. Kromozom için bir yazı tura atarak çocuğa dâhil edilip edilmeyeceğine karar verilir (Şekil 3.11).

Çocuklar aşağıdaki algoritma ile oluşur:

Eğer $Atış = 1$ ise, bitler değişir.
Eğer $Atış = 0$ ise, bitler değişmez.



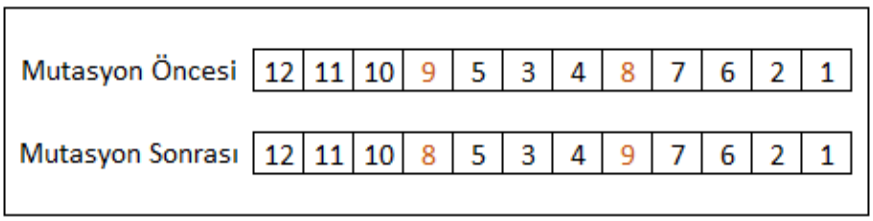
Şekil 3.11. Üniform Çaprazlama

Çaprazlamanın kalitesi, sonucun kalitesini etkiler. Bu nedenle temel ve geleneksel çaprazlama yöntemlerinden farklı olarak birçok farklı çaprazlama tekniği geliştirilmiştir. Çaprazlama yöntemlerin de sınırlamaları olabilir. Örneğin permütasyon kodlaması ile oluşturulan iş sıralama algoritmalarında öncüllük ilişkileri olduğu için kesişim noktasında bu öncül ilişkilere göre düzenlenmesi gerekir. Aksi takdirde algoritma, mümkün olmayan anlamsız sonuçlar üretmeye başlar.

Mutasyon

Mutasyon, genetik algoritmada, kromozom popülasyonunun bir neslinden diğerine genetik çeşitliliğini korumak için kullanılan bir genetik operatördür. Mutasyon operatörünün klasik örneği, genetik dizideki rastgele bir bitin orijinal durumundan çevrilmesi olasılığını içerir. Mutasyon operatörünü uygulamanın yaygın bir yöntemi, bir dizideki her bit için rastgele bir değişken üretmeyi içerir. Bu rastgele değişken, belirli bir bitin çevrilip çevrilmeyeceğini söyler. Diğer türleri ise ters çevirme ve kayan nokta mutasyonudur. Gen kodlaması, permütasyon problemlerinde olduğu gibi kısıtlayıcı olduğunda, mutasyonlar, değiş tokuşlar, ters çevirmeler ve karıştırmalar şeklinde yapılır.

Mutasyon operatörleri, kromozom popülasyonunun birbirine çok benzer hale gelmesini önleyerek yerel minimum veya maksimumdan kaçınma girişiminde kullanılır; böylece yerel optimuma yakınsamayı yavaşlatır veya durdurur. Genellikle çizelgeleme problemi için değiştirme mutasyonu veya yerel arama tabanlı mutasyonlar kullanılır (Şekil 3.12).



Şekil 3.12. Mutasyon

3.2. Yöntem

3.2.1. Kaynak Dengeleme, Kaynak Kısıtlı Proje Çizelgeleme ve Yoğunluk Değişkeni Formülasyonu

Burada kaynak kullanımını mevcut sınırlı kaynaklar bazında dengelemek için üretim planlama veya proje çizelgelemedeki her bir faaliyetin başlangıç ve bitiş tarihlerinin öncüllük koşullarına göre ayarlandığı bir yöntem önerilecektir. Kis (2005) tarafından tanıtılan ve Alfieri ve ark. (2012) tarafından geliştirilen Besleme Öncüllük İlişkisi kavramı, $C_{\max}$'ı en küçüklemek için kullanılmakta olup, kaynakların kullanımını dengelemek için herhangi bir işlevi bulunmamaktadır. Bu bölümde Alfieri ve ark. (2012) 'nin modeli, $C_{\max}$'ı enküçüklerken aynı zamanda kaynak kullanımını dengelemek amacıyla iyileştirilecektir.

Bir proje, faaliyetleri yürütmek için kaynaklara ihtiyaç duyar. Bu kaynaklar, işin yapılması için gereken işgücü, ekipman ve malzemeleri içerir. İşçigücü, zanaatkârlar, mühendisler, programcılar, sistem analistleri vb. gibi insanlardır. Ekipman, vinçler, test teçhizatları, süreç simülatörleri vb. şeyleri içerir. Malzemeler, dökülecek beton, kurulacak tel gibi şeyleri içerir. İdeal dünyada kaynaklar sınırsızdır ve istendiğinde temin edilebilir. Ancak gerçekte, kaynaklar genellikle sınırsız değildir ve proje ekibinin kaynakların kullanımını ve tüketimini dengelemesi gerekir. Planlayıcının temel zorluğu, verilen kısıtlı zamana uyan ve mevcut kaynaklarla gerçekleştirilebilecek bir proje yürütme planı geliştirmektir. Çeşitli zamanlarda kaynaklar için yarışan birçok faaliyet olduğunda, faaliyetleri kaynak yetersizliğinden geri kalmayacak şekilde planlamaya çalışmak karmaşık bir sorundur.

Bu çalışmada, değişken yoğunluklu proje çizelgelemede başarılı olan bir matematiksel yöntem seçilmiştir; önceki bölümde bahsedildiği gibi modelin amaç fonksiyonu $C_{\max}$'ı minimize etmektedir. Bu model, $C_{\max}$'ın yanı sıra kaynak dengeleme için genelleştirilmiştir. Bir önceki bölümde bahsedilen modele kaynak dengeleme amacını eklemek için, amaç fonksiyonu değişmeli ve aşağıdaki gibi bir kısıt eklenmelidir.

Amaç Fonksiyonu:

$$\text{Min}\left[\alpha \cdot C_{max} + \sum_{k=1}^{K}\sum_{t=1}^{d}(u_{kt})^2\right]$$

Yeni kısıt:

$$u_{kt} = \sum_{j=1}^{J} x_{jt} \cdot Q_{jk} \qquad \forall t\,, k$$

Amaç fonksiyonundaki ikinci terim $\sum_{k=1}^{K}\sum_{t=1}^{d}(u_{kt})^2$, kaynak dengelemeye imkan sağlar. Burada $C_{\max}$ proje süresini en aza indirmek içindir, ancak $C_{\max}$'ın sayısal değeri $\sum_{k=1}^{K}\sum_{t=1}^{d}(u_{kt})^2$ kısmına kıyasla küçük olduğundan, $C_{\max}$'ın minimizasyon ağırlığı bazen etkisiz kalabilir. $C_{\max}$ 'a α gibi bir çarpan eklenerek, amaç fonksiyonunda $C_{\max}$ 'ın ağırlığı kaynak tüketiminin ağırlığına eşitlenebilir. α'nın büyüklüğü, kaynakların sayısal boyutuna bağlıdır.

Yukarıda belirtilen kısıt ve amaç fonksiyonu eklenerek, son model aşağıdaki gibi oluşur:

Parametreler

J	Faaliyetlerin kümesi
T	Proje dönemlerinin kümesi
K	Kaynakların kümesi
τ	Öncül ardıl ilişkiler kümesi
$\tau_1 \in \tau$	Öncüllük ilişkilerinin alt kümesi (*%Completed-to-Start* (CtS))
$\tau_2 \in \tau$	Öncüllük ilişkilerinin alt kümesi (*Start-to-%Completed* (StC))
$\tau_3 \in \tau$	Öncüllük ilişkilerinin alt kümesi (*%Completed-to-Finish* (CtF))
$\tau_4 \in \tau$	Öncüllük ilişkilerinin alt kümesi (*Finish-to-%Completed* (FtC))
B_j	j faaliyeti tarafından bir dönemde yapılabilecek maksimum iş yüzdesi
b_j	j faaliyeti tarafından bir dönemde yapılabilecek minimum iş yüzdesi
i_p	Öncül faaliyet $p \in \tau$
j_p	Ardıl faaliyet $p \in \tau$

q_p j_p faaliyetinin belirli bir dönemde başlamasına veya bitmesine izin vermek için i_p faaliyeti için gereken tamamlanma yüzdesi, $p \in (\tau_1, \tau_3)$.

g_p j_p faaliyetinin belirli bir dönemde başlamış veya bitmiş olması halinde, i_p faaliyetinde elde edilebilecek iş yüzdesi, $p \in (\tau_2, \tau_4)$.

r_j j faaliyetinin başlayabilecek zamanı

d_j j faaliyetinin son tarihi

Q_{jk} j faaliyetini tamamen işlemek için gereken k kaynağının miktarı

R_{kt} t döneminde mevcut olan k kaynağının toplam miktarı

Değişkenler

$C_{\max}$ Maksimum tamamlanma zamanı, $C_{\max} \in \mathbb{R}^+$

u_{kt} k kaynağından t döneminde kullanım miktarı, $u_{kt} \in \mathbb{R}^+$

x_{jt} t döneminde j faaliyetinin tamamlanma yüzdesini temsil eden sürekli pozitif değişken, $x_{jt} \in \mathbb{R}^+$

f_{jt} $\begin{cases} 1, & j \text{ faaliyeti } t \text{ döneminde biterse} \\ 0, & d.d. \end{cases}$

s_{jt} $\begin{cases} 1, & j \text{ faaliyeti } t \text{ döneminde başlarsa} \\ 0, & d.d. \end{cases}$

η_{jt} $\begin{cases} 1, & j \text{ faaliyeti t döneminde işlenirse} \\ 0, & d.d. \end{cases}$

Amaç fonksiyonu:

$$\text{Min} \left[\alpha \cdot C_{max} + \sum_{k=1}^{K} \sum_{t=1}^{d} (u_{kt})^2 \right] \tag{3.20}$$

Kısıtlar:

$$u_{kt} = \sum_{j=1}^{J} x_{jt} \cdot Q_{jk} \qquad \forall t\,, k \tag{3.21}$$

$$C_{\max} \geq \sum_{t=r_j}^{d_j} t \cdot f_{jt}\,, \qquad \forall\, j \tag{3.22}$$

$$\sum_{t=r_j}^{d_j} s_{jt} = 1, \quad \forall\, j \tag{3.23}$$

$$\sum_{t=r_j}^{d_j} f_{jt} = 1, \quad \forall\, j \tag{3.24}$$

$$\sum_{t=r_j}^{d_j} t \cdot s_{jt} \geq r_j, \quad \forall\, j \tag{3.25}$$

$$\sum_{t=r_j}^{d_j} x_{jt} = 1, \quad \forall\, j \tag{3.26}$$

$$x_{jt} \leq B_j \cdot \eta_{jt}, \quad \forall\, j, t \tag{3.27}$$

$$x_{jt} \geq b_j \cdot \eta_{jt}, \quad \forall\, j, t \tag{3.28}$$

$$s_{jt} \leq \eta_{jt}, \quad \forall\, j, t \tag{3.29}$$

$$f_{jt} \leq \eta_{jt}, \quad \forall\, j, t \tag{3.30}$$

$$f_{jt} \leq \sum_{h=r_j}^{t} x_{jh}, \quad \forall\, j, t \tag{3.31}$$

$$\sum_{h=r_j}^{t} s_{jh} \geq x_{jt}, \quad \forall\, j, t \tag{3.32}$$

$$\sum_{j \in J} Q_{jk} \cdot x_{jt} \leq R_{kt}, \quad \forall\, k, t \tag{3.33}$$

$$s_{jt} \leq \sum_{h=1}^{t-1} x_{ih} - q_p + 1, \quad \forall\, p \in \tau_1, i = i_p, j = j_p\,, \forall t \tag{3.34}$$

$$\left(1 - \sum_{h=1}^{t} f_{jt}\right) \geq q_p - \sum_{h=1}^{t-1} x_{ih}, \quad \forall\, p \in \tau_3, i = i_p, j = j_p\,, \forall t \tag{3.35}$$

$$\sum_{h=1}^{t} x_{jh} \leq g_p + \left(1 - g_p\right) \sum_{h=1}^{t-1} s_{ih}\,, \quad \forall\, p \in \tau_2, i = i_p, j = j_p\,, \forall t \tag{3.36}$$

$$\sum_{h=1}^{t} x_{jh} \leq g_p + (1 - g_p) \sum_{h=1}^{t-1} f_{ih} \ , \quad \forall\, p \in \tau_4, i = i_p, j = j_p \ , \forall t \qquad (3.37)$$

Kısıtlar $(3.22) - (3.37)$ daha önce detaylı olarak (3.1.1) bölümünde açıklanmıştır. Bölüm (3.1.1) ve bu bölümde açıklanan modeller yanında, sadece kaynak dengelemeli amaç fonksiyonu içeren model, kesin çözüm için kodlanmış ve küçük bir rassal örnek veri seti (üç farklı model için aynı veri seti) üzerinde çalıştırılarak, sonuçlar yorumlanmıştır. Çizelge (3.1) ve (3.2)'de sonuçlar verilmektedir. Söz konusu örnekte 7 faaliyet ve 4 (yenilenebilir) kaynak bulunmaktadır.

Çizelge 3.1. Zaman dönemlerinde iş atama planlaması

(a) Amaç Fonksiyonu Min C_{max}																			
İş No 1	0,6				0,4														
İş No 2			0,6		0,4														
İş No 3						0,4	0,6												
İş No 4				0,2				0,8											
İş No 5					0,4			0,6											
İş No 6						0,6		0,4											
İş No 7							0,5	0,5											
(b) Amaç Fonksiyonu Min C_{max} ve kaynak dengeleme																			
İş No 1	0,5	0,5																	
İş No 2			0,6	0,4															
İş No 3						0,6	0,4												
İş No 4				0,3	0,4		0,3												
İş No 5					0,6			0,4											
İş No 6						0,3		0,3	0,4										
İş No 7							0,2	0,3	0,5										
(c) Amaç Fonksiyonu sadece kaynak dengeleme																			
İş No 1	0,5	0,5																	
İş No 2			0,6	0,4															
İş No 3						0,5	0,5												
İş No 4				0,2										0,2	0,3	0,3			
İş No 5					0,5			0,5											
İş No 6									0,3	0,3		0,4							
İş No 7											0,2		0,2				0,2	0,2	0,2
Dönem	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**

Çizelge 3.1(c)'ye göre amaç fonksiyonu sadece kaynak dengeleme ise, çalışmaların tamamlanması planlama ufkuna yayılmış ve 18 döneme uzamıştır. Amaç fonksiyonu, Min C_{max} ise Çizelge 3.1(a)'ye göre işler 7 periyotta tamamlanmış, ancak her iki hedefin eşit ağırlıkta dikkate alınması söz konusu olduğunda, Çizelge 3.1(b)'ye göre işlerin 8 periyotta tamamlanması sağlanmıştır. Bu durum için Çizelge 3.2(b)'daki detaylı kaynak kullanım miktarlarına bakıldığında, Çizelge (3.2)(a)'ya göre dönemler bazında kaynak dengelemesinin gerçekleştiği açıkça görülmektedir.

Çizelge 3.2. Zaman dönemlerinde kaynak kullanımı

(a) Amaç Fonksiyon Min C_{max}																			
Kaynak 1	3	0	2,4	1,2	4,8	7,2	7,1	13											
Kaynak 2	3	0	3	1	6,8	3,6	6,6	12											
Kaynak 3	6	0	3	0	8,8	3	5,6	6,6											
Kaynak 4	6	0	4,2	2	8,4	5,4	8,6	17											
(b) Amaç Fonksiyon Min C_{max} ve kaynak dengeleme																			
Kaynak 1	2,5	2,5	2,4	4	4,2	6	5	5,7	5,9										
Kaynak 2	2,5	2,5	3	4	6,2	4,2	4,6	5,2	3,6										
Kaynak 3	5	5	3	2	4,2	3,9	3,2	4,3	2,3										
Kaynak 4	5	5	4,2	6,8	6,4	5,1	6,4	6,1	6,5										
(c) Amaç Fonksiyon sadece kaynak dengeleme																			
Kaynak 1	2,5	2,5	2,4	2,8	1,5	3	3	1,5	2,4	2,4	1,4	2,4	1,4	1,8	1,8	1,8	1,4	1,4	1,4
Kaynak 2	2,5	2,5	3	3	3,5	3	3	3,5	0,6	0,6	1,2	0,6	1,2	1,5	1,5	1,5	1,2	1,2	1,2
Kaynak 3	5	5	3	2	3,5	3	3	3,5	0,3	0,3	0,8	0,3	0,8	0	0	0	0,8	0,8	0
Kaynak 4	5	5	4,2	4,8	2	3	3	2	1,5	1,5	2	1,5	2	3	3	3	2	2	2
Dönem	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**

3.2.2. Önerilen Genetik Algoritma

Önerilen matematiksel model NP-zor olduğundan, bu modeli büyük veri kümeleri için kullanmak mümkün değildir ve bu durumda daha kısa sürede kabul edilebilir bir çözüm bulmak için başka bir çözüm yönteminin kullanılması gerekir. Çözüm yöntemi olarak, bu tez kapsamında yeni bir Genetik Algoritma geliştirilmiştir. Kurgulanan GA yaklaşımında başlangıç kromozomları oluşturulduktan sonra, hesaplanan uygunluk değeri, tüm işlerin tamamlanma süresinin α katıyla ile her dönemde kullanılan kaynakların karesi alınmış miktarının toplamı olarak hesaplanır. Yeni nesil üretmek için öncelikle rulet tekerliği tekniği ile iki ebeveyn kromozomu seçilir ve çaprazlama yapılarak yeni çocuklar üretilir. Mutasyonlar yapılarak yerel-optimal çözümlerin önüne geçilir ve elitist yaklaşım kullanılarak sonucun optimum noktaya yaklaşması sağlanır. Bu süreç iterasyon sayısı kadar devam eder (Şekil 3.13). Popülasyon sayısı, çaprazlama ve mutasyon oranı parametre optimizasyonu yöntemleri ile bulunur.

Girdi:
Popülasyon sayısı: pop, çaprazlama oranı P_c, mutasyon oranı P_m, işlerin ve mevcut kaynakların bilgisi, iterasyon sayısı, Elit çözüm sayısı
Çıktı:
Çizelgeleme sonucu.

1 Popülasyon sayısı kadar kromozom oluştur. **Şekil (3.16)**
2 Popülasyonu amaç fonksiyonuna göre sırala
3 For ir = 1 to ir = iteration sayısı
4 Do
5 Çaprazlama sayısı = $p_c \times popülasyon\ sayısı$
6 For cross = 0 to cross = $\frac{çaprazlama\ sayısı - elit\ sayısı}{2}$
7 Do
8 Seçim yöntemine göre 2 ebeveyn seç. **Şekil (3.17)**
9 Tek noktalı çaprazlama ile 2 çocuk oluştur. **Şekil (3.18)**
10 Çocukların kaynak ve öncüllük şartları sağlamasını **Şekil (3.20)** **Şekil (3.21)** kontrol et.
11 Eğer kaynak veya öncüllük şartları sağlanmıyorsa düzeltme yap.
12 Çocukları yeni nesil popülasyona Pop_2 ekle
13 End
14 For mutasyon sayısı = 0 to mutasyon sayısı = $p_m \times popülasyon\ sayısı$
15 Do
16 Pop_2 'den rastgele bir çözüm seçin;
17 Mutasyon ile 1 yeni çocuk oluştur. **Şekil (3.19)**
18 Kaynak ve öncüllük şartları sağlanmiyorsa düzeltme yap.
19 Sonucu yeni nesil olarak popülasyon Pop_2'ye ekle ve Mutasyon için seçilen kromozomu Pop_2'den çıkar.
20 End
21 Pop_1 den elit sayısı kadar kromozomları Pop_2 ekle.
22 End

Şekil 3.13. Genetik Algoritmanın sözde kodu

Kromozom Oluşturma

Her faaliyet için iki kromozom ayrılmıştır. İlk kromozom her bir dönemde faaliyetlerin tamamlanma oranını, ikinci kromozom ise faaliyetlerin yürütüldüğü dönemleri gösterir. Her iki kromozom da aynı uzunluktadır ve faaliyet sayısının n katıdır. h, bir aktivitenin maksimum tamamlanma periyot sayısını temsil eder.

h'ı bulmak için projenin tüm faaliyetlerin arasında en küçük b_{min} bulunmalıdır ve ardından $h = \left\lfloor \frac{1}{\min_j b_{min}(j)} \right\rfloor$ olur. b_{min}bir dönemde bir faaliyetin en az tamamlanma oranı, $\boldsymbol{B_{max}}$ ise en fazla tamamlanma oranı olarak tanımlanmaktadır. Örneğin, $n = 4$ faaliyet için en küçük b_{min} değeri 0,3 ise kromozom uzunluğu $= 4 \times \left\lfloor \frac{1}{0,3} \right\rfloor = 12$ olarak bulunur.

İlk kısım (h=3, kromozom uzunluğu=12)

0,3	0,7	0	0,2	0,2	0,6	0,5	0,5	0	0,4	0,6	0
$İş_1$			$İş_2$			$İş_3$			$İş_4$		

İkinci kısım (h=3, kromozom uzunluğu=12)

1	2	0	2	4	5	2	3	0	5	6	0
$İş_1$			$İş_2$			$İş_3$			$İş_4$		

Şekil 3.14. $\boldsymbol{n = 4}$ faaliyet için Örnek Kromozom yapısı

Şekil 3.14'deki Örnek Kromozom yapısı Şekil 3.15'deki çizelgeye karşılık gelmektedir..

$İş_4$					0,4	0,6
$İş_3$		0,5	0,5			
$İş_2$		0,2		0,2	0,6	
$İş_1$	0,3	0,7				
Dönem	1	2	3	4	5	6

Örnek	$\boldsymbol{b_{min}}$	$\boldsymbol{B_{max}}$	**Başlaya bilecek dönem**	**Termin tarihi**	**Kaynak kullanımı**			**Ardıl iş**	**Ön koşul türü**
					Kaynak 1	**Kaynak 2**	**Kaynak 3**		
$İş_1$	0,3	0,6	1	4	2	3	0	2	2
$İş_2$	0,2	0,6	2	6	4	0	6	3	4
$İş_3$	0,3	0,7	2	8	1	3	2	4	1
$İş_4$	0,3	0,8	2	10	2	4	5	…	…

Şekil 3.15. $n = 4$ faaliyet için örnek rassal veri seti ve sonuç çizelgesi

Şekil 3.14'deki kromozom yapısı ve Şekil 3.15 yer alan verilere göre kromozom oluşturma adımları aşağıdaki gibi tanımlanabilir:

Adım 1:

En küçük b_{min} ve kromozom uzunluğunu hesapla. Kromozom uzunluğu eşit olan iki boş kromozom dizisi oluştur.

$$\min_j b_{min}\,(\mathrm{j}) = 0.3$$

$$\text{kromozom uzunluğu } = 4 \times \left\lfloor \frac{1}{0{,}3} \right\rfloor = 12$$

Birinci kromozom

–	–	–	–	–	–	–	–	–	–	–	–

İkinci kromozom

–	–	–	–	–	–	–	–	–	–	–	–

Adım 2:

Öncüllere göre uygun faaliyetler listesini oluştur ve bunları küçükten büyüğe sırala. Uygun faaliyetler listesi: $\{\text{İş}_1\}$.

Adım 3:

Aktiviteler listesinde en önce gelen aktiviteyi seç ve seçilen aktivitenin ardıllarını listeye ekle (Zaten listede ise veya seçili ise tekrar eklenmez). Seçilen işi listeden çıkar. Seçilen iş: İş_1, Uygun faaliyetler listesi: $\{\text{İş}_2\}$.

Adım 4:

Bu koşullara göre seçilen işin tamamlanma oranlarını (x_{it}) rastgele bul:

$$b_{\min(\text{iş}_1)} \leq x_{it} \leq B_{\max(\text{iş}_1)}, \quad \sum_{t=releasetime\,(i)}^{termin\ tarihi\,(i)} x_{it} = 1$$

İlk kromozom (h = 3, kromozom uzunluğu=12)

0,3	0,7	0	–	–	–	–	–	–	–	–	–
İş_1			İş_2			İş_3			İş_4		

Adım 5:

Tamamlanma süreleri, öncül ve ardıl koşulları ve kaynakların kullanılabilirliğini sağlayarak, başlatma zamanı ile işin son tarihi arasındaki dönemleri rastgele bul.

İkinci kromozom (h = 3, kromozom uzunluğu=12)

1	2	0	–	–	–	–	–	–	–	–	–
$İş_1$			$İş_2$			$İş_3$			$İş_4$		

Adım 6:

Tüm işler seçilene kadar Adım 2'ye git. Bu aşamanın sonunda başlangıç neslin ilk kromozomu oluşur.

Bu adımlar, kromozom sayısı popülasyona ulaşana kadar devam eder.

```
Girdi: işlerin bilgisi(bmin, Bmax, öncül ardıl işler ve tipler, kaynaklar,
releasetime ve temrin tarihi)
1   j.= işlerin sayısı
2   En küçük b_min bul.
3   h = 1 / (en küçük b_min)
4   For pop =1 to pop popülasyon
5       Do
6           For j=1 to j=J
7               Do
8               X_1[j] ← 0
9                   For h=1 to h=h
10                      Do
11                          X[(j×h)+h] = RANDOM (b_min , B_max )
12                          X_1[j] = X_1 [j] + X[(j×h)+h]
13                          If X_1[j] < 1
14                                  Öncül ve ardıl türü ve işe göre başlanabilir
15                                  ve tamamlanabilir dönemleri bul.
16                                  Do  P[(j×h)+h]  =  RANDOM  (başlanabilir  dönem,
                                    tamamlanabilir dönem)
17                          End
18                          Else
19                                  P[(j×h)+h] ← 0,  X[(j×h)+h] ← 0
20                          End
21                          If (kaynak şartı sağlanıyor)
22                                  End
23                          Elseif
24                                  (Go to 11)
25                          End
26                      End
27              End
28      End
```

Şekil 3.16. Kromozom oluşturma sözde kodu

```
Girdi:
Pöpulasyon sayısı kadar sonuç
Çıktı
İki kromozom ebeveyni olarak seçilmiş.
1   Populasyon amaç fonksiyonuna göre sırala
2   For pop=1 to pop <= popülasyon sayısı.
3       Do
4           N= mevcut neslin boyutu,
5           f_pop = Kromozomun uygunluk değeri.
6           p_pop = f_pop / Σ_{j=1}^{pop} f_pop her kromozom için  p_pop kümülatif hesapla.
7       End
8       For RnKromozom=1 to 2
9           Do
10          p [RanKromozom] <= RANDOM (0.01,1)
11              For pop=1 to pop <= popülasyon sayısı
12                  Do
13                      if (p_(pop-1) (kümülatif) < p [RanKromozom] < = p_pop (kümülatif) )
14                          Ebeveyn kromozom = kromozom[pop]
15                  End
16          End
17      End
```

Şekil 3.17. Rulet tekerliği seçimin sözde kodu

Çaprazlama işlemi

Çaprazlama için tek noktalı ve iki noktalı Çaprazlama yöntemleri kullanılmıştır. Tek noktalı çaprazlama yöntemine göre kromozomun rastgele bir noktadan sonraki alanı, diğer kromozomla değiştirilir. Ancak, öncül ilişkileri ve kaynak kısıtlamaları nedeniyle tam olarak kopyalanamazlar. Algoritma aşağıdaki adımlardan oluşur:

Adım 1:

Ebeveynler Rulet tekerleği Seçimi yöntemine göre seçilir.

İlk ebeveyn, İlk kromozom

0,3	0,3	0,4	0,5	0,5	0	0,6	0,4	0	0,3	0,4	0,3

İlk ebeveyn, İkinci kromozom

1	2	3	2	5	0	3	5	0	4	5	6

İkinci ebeveyn, İlk kromozom

0,5	0,5	0	0,3	0,4	0,3	0,4	0,6	0	0,5	0,5	0

İkinci ebeveyn, İkinci kromozom

1	3	0	2	4	5	4	6	0	5	8	10
$İş_1$			$İş_2$			$İş_3$			$İş_4$		

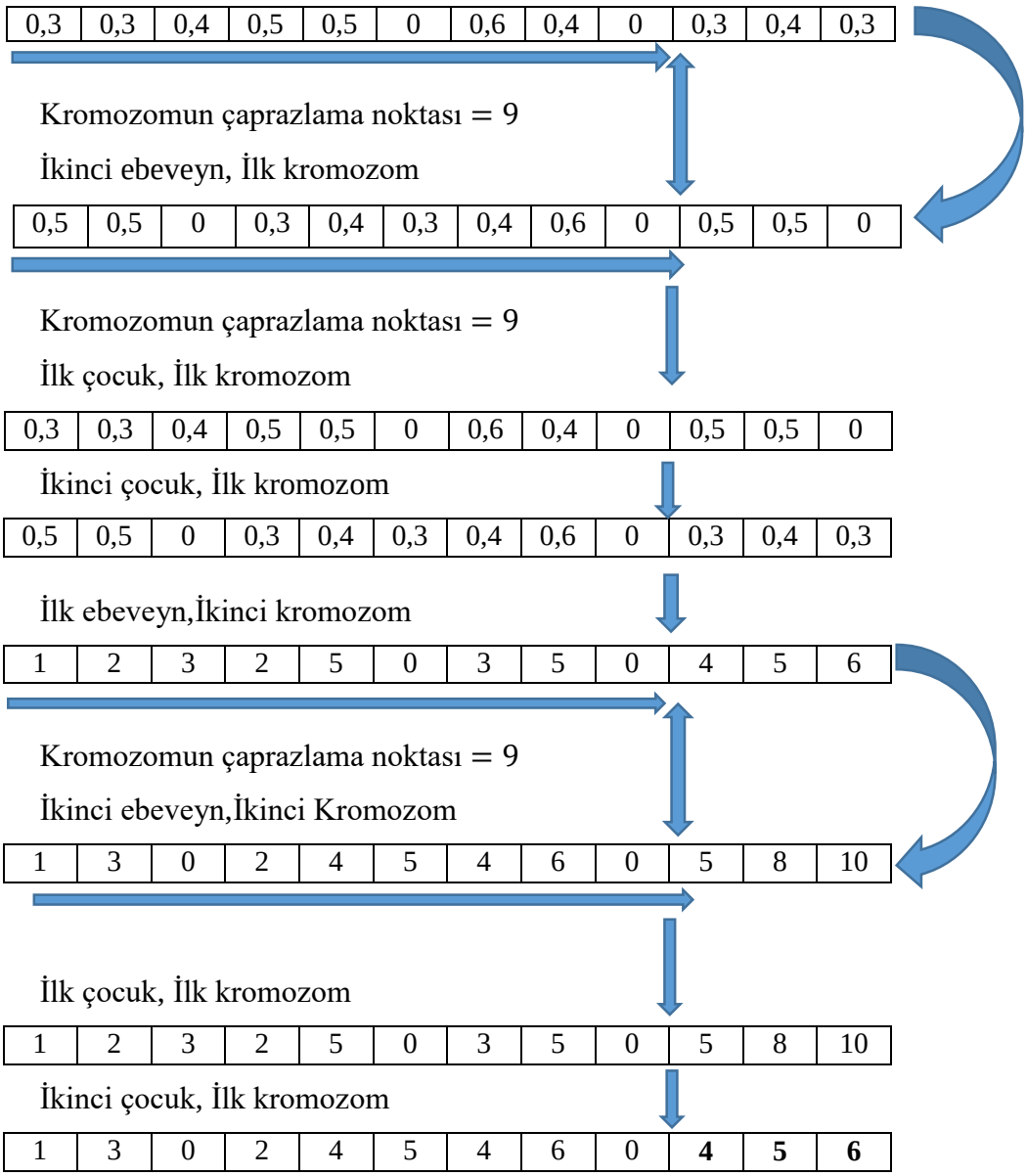
Adım 2:

1'den j'ye kadar olan sayılardan rastgele bir nokta bulunur. Bu noktanın h katından sonra kromozomun yerini diğer kromozom alır.

$j = 4$

Rasgele nokta: rasgele {1…4} = 3

Kromozomun çaprazlama noktası = $3 \times h = 3 \times 3 = 9$



Adım 3:

Öncül koşulları ve kaynakların yeterli olup olmadığını kontrol et. Atama süresi boyunca koşullar karşılanmazsa veya kaynaklar yetersizse çocukların ikinci kromozomlarını yeniden ayarla.

Yukarıdaki örnekte, ikinci çocuğun öncül koşulu karşılanmamıştır. Dördüncü işin öncülü, tip 1 ön koşulunu kullanan üçüncü iştir. Tip 1 ön şart nedeniyle, (%*Completed-to-Sta*rt (CtS)), dördüncü iş, üçüncü işin tamamlanmasının belirli bir yüzdesinden sonra başlayabilir, ancak bu örnekte bu koşul karşılanmamıştır; bu nedenle koşulu sağlamak için çaprazlama noktasından sonraki kromozom parçasının yeniden ayarlanması gerekir.

İkinci çocuk, İlk kromozom

1	3	0	2	4	5	4	6	0	**5**	**6**	**7**

Çaprazlama işlemi, çaprazlama oranına bağlı olarak belirli bir adet kadar yapılmalıdır.

İki noktalı Çaprazlama kullanılıyorsa, 2. Adımda rastgele iki nokta bulunur ve bu iki nokta arasındaki genler ebeveynler arasında değiştirilir. Bu iki nokta arasında öncül ardıl koşulları kontrol edilir.

```
Girdi:
İlk popülasyon ve işlerin bilgisi.
Çıktı:
İki yeni uygun kromozom .
1   Rulet tekerleği tekniği ile 2 ebeveyn seç. (Şekil 3,15)
2   Çaprazlama noktası = RANDOM {1, j}
3   For h=1 to Çaprazlama noktası kadar.
4       DO
5           Krom1[k] ← ilk ebeveyn.
6           Krom2[k] ← ikinci  ebeveyn.
7       End
8   For h= Çaprazlama noktasından to J.
9       Do
10          Krom1[k] ← ikinci  ebeveyn.
11          Krom2[k] ← ilk ebeveyn.
12      End
13  If (kaynak ve ön şartlar sağlanıyor)
14      End
15      Elseif
16          Krom2[k] çaprazlama noktadan sonra düzelt (ön şart ve kaynak sağlasın)
            (Şekil 3,18)
17      End
```

Şekil 3.18. Çaprazlama işlemi sözde (Tek noktalı)

Mutasyon

Adım 1:

1'den j'ye kadar olan sayılardan rastgele bir nokta bulunur. $j = 4$.

Rasgele nokta: rasgele {1…4} = 2, $h = 3$

Mutasyon yapılan Genler: {2×2… (2×2) + (3-1)}: {4, 5, 6}

İlk kromozom

0,5	0,5	0	0,4	0,3	0,3	0,4	0,6	0	0,5	0,5	0

İkinci kromozom

2	3	0	3	4	5	5	6	0	8	10	0

Adım 2:

Ön koşulları sağlayarak tekrar mutasyona uğraması gereken genler, birinci kromozom için b_{min}ve B_{max} arasında ve ikinci kromozom için bırakma zamanı ile temrin tarihi arasında rastgele atama yapılır.

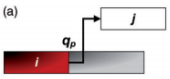
Adım 3:

Kaynakların yeterli olup olmadığını kontrol edilir, yeterli değilse, Adım 2'ye dönülür.

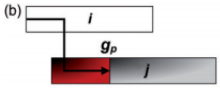
Girdi:
Pop2(Önceki neslin elitleri ve çaprazlamanın sonuçları)
Çıktı :
Mutasyon yapılmış yeni Kromozom
j=işlerin sayısı
h=$\frac{1}{en\ küçük\ b_{min}}$
1 K=RANDOM {1, $çaprazlama\ sayısı + elit\ sayısı$ }
2 Mutasyon noktası=RANDOM {1, J};
3 For b=1 to Mutasyon noktası
4 Do
5 $Krom_{k1}$(Mutasyon) = $Krom_{k1}$ (part 1);
6 $Krom_{k2}$(Mutasyon) = $Krom_{k2}$ (part 2);
7 End
8 X_1[j] =0;
9 For b=Mutasyon noktası ×h to (Mutasyon noktası +1)×h
10 Do
11 $Krom_{k1}$(Mutasyon) = RANDOM ($b_{min}(j)$, $B_{max}(j)$);
12 X_1 [j] = X_1 [j] + $Krom_{k1}$(Mutasyon)
13 IF X_1 [j] <1
14 Öncül ve ardıl türü ve işe göre başlanabilir dönem ve tamamlanabilir dönem dönemleri bul.
15 $Krom_{k2}$(Mutasyon) = RANDOM (başlanabilir dönem, tamamlanabilir dönem)
16 Else
17 $Krom_{k2}$(Mutasyon) = 0, $Krom_{k1}$(Mutasyon) = 0
18 End
19 For j=1 to J
20 Do
21 $Krom_{k1}$(Mutasyon) = $Krom_{k1}$ (part 1);
22 $Krom_{k2}$(Mutasyon) = $Krom_{k2}$ (part 2);
23 End
24 If (kaynak şartı sağlanıyor)
25 End
26 Elseif
27 (Go to 8)
28 End

Şekil 3.19. Mutasyon işlemi sözde kodu

```
Girdi: qp, gp, kromozom
Çaprazlama noktadaki işin ön koşul ve ön koşulun tipini bul(eğer mutasyondan sonraki
kromozomsa mutasyon noktadaki işin ön koşulu ve tipini bul.)
Ön kontrol=0, Kaynak kontrol=0;
```

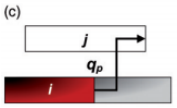



```
1   if ön koşulun tipi== 1
2   Öncülün tamamlanma oranı=0
3   For k=1 to h
4       Do
5       Öncülün tamamlanma oranı= Öncülün tamamlanma oranı+ kromozomunm ilk kısmı
        (öncülün işin numarası × h) + k.
6       Başlama nokta= kromozomun ikinci kısmının (öncül işin numarası* h) +k;
7       İf Öncülün tamamlanma oranı >= qp ( break)
8       End
9       If ardılın başlama dönemi <= Başlama nokta
10              (Önkontrol= 1)
11      End
```

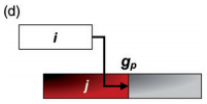



```
12  If ön koşulun tipi == 2
13  ardılın tamamlanma oranı=0
14  for k=1 to h
15      Do
16              Ardılın tamamlanma oranı = Ardılın tamamlanma oranı + kromozomunm ilk
                kısmının' in  (ardıl işin numarası × h) + k;
17              Başlama nokta= kromozomunm ikinci kısmının' in (ardıl işin numarası* h)
                        +k;
18              If ardılın tamamlanma oranı >= gp ( break)
19      End
20  If öncülün başlama noktası >= Başlama nokta
21      (Önkontrol= 1)
22  End
```

Şekil 3.20. Ön koşul ve kontrolünün sözde kodu




```
23  İf ön koşulun tipi == 3
24  Önculun tamamlanma oranı=0
25        For k=1 den h kadar
26            Do
27                Önculun tamamlanma oranı = Öncülün tamamlanma oranı + kromozomunm ilk
                      kısmının' in (öncülün işin numarası * h) + k;
28                Bitiş nokta= kromozomunm ikinci kısmının' in (öncülün işin numarası*
                      h) +k;
29                İf Önculun tamamlanma oranı    >=  q_p ( break)
30            End
31        If Çaprazlama noktadaki işin bitiş noktası <= Bitiş nokta
32            Önkontrol= 1
33        End
```




```
34  If ön koşulun tipi == 4
35      Ardılın tamamlanma oranı=0
36      For k=1 to h
37      Do
38              Ardılın tamamlanma oranı = Ardılın tamamlanma oranı  + kromozomunm ilk
                kısmının' in  (Ardıl işin numarası * h) + k;
39              Bitiş nokta= kromozomunm ikinci kısmının' in (Ardıl işin işin numarası*
                      h) +k;
40              If Ardılın tamamlanma oranı >= g_p ( break)
41      End
42      If Bitiş nokta <= öncülün bitiş noktası
43              Önkontrol= 1
44      End
45  If Önkontrol > 0  (ön koşullar sağlanmıyor.)
46  End
```

Şekil 3.21. Ön köşül ve kontrolünün sözde kodu (devam)

3.2.3. Parametre Optimizasyonu

GA için çözüm kalitesini ve hızını etkileyen üç anahtar parametre vardır: popülasyon büyüklüğü POP, çaprazlama oranı P_c ve mutasyon oranı P_m .

Algoritmanın parametrelerini optimize etmek için tüm veri kümesini temsil eden 27 örnekten oluşan küçük bir alt küme oluşturulmuş olup, her örnek 5 kez çalıştırılmış ve ardından tüm 5 sonucun ortalaması alınmıştır. Örneklem kümesi popülasyon büyüklüğü (25,50,100), çaprazlama oranı (0.7,0.8,0.9) ve mutasyon oranı (0.01, 0.05,0.1) kombinasyonlarıdır.

4. BULGULAR

4.1. Yazılım ve Donanım

Matematiksel modelin çözümü amacıyla uygulamada kullanılan yazılım, Visual Studio 2019 ortamında C# dili ve Gurobi çözücüsü ile geliştirilmiştir. Ayrıca önerilen Genetik Algoritma Visual Studio 2019 ortamında C# dilinde kodlanmıştır. Kullanılan bilgisayar donanım özellikleri: 64 bit işletim sistemi, x64 tabanlı işlemci, Intel(R) Core (TM) i5-8265U CPU @ 1.80 GHz, Hafıza takılı RAM 8,00 GB, İşletim Sistemi Windows 10'dur.

4.2. Uygulama

4.2.1. Veri Seti

Küçük örnek veri kümeleri oluşturmak için rastgele üretim yöntemi kullanılmış olup, oluşturulan veri kümeleri ve kaynak dağılımı (sonuç olarak) Çizelge 4.4–4.8'de gösterilmektedir. Küçük veri kümeleri için dört tip kaynak için $g_p = 0{,}2$ ve $q_p = 0{,}2$ değerleri dikkate alınmıştır.

Büyük veri setleri için PSBLIB kütüphanesi (Kolisch ve Hartmann 2005) kullanılmış olup, bu kütüphane çeşitli kaynak kısıtlı proje çizelgeleme problemlerinin yanı sıra optimal ve sezgisel çözümler için farklı problem kümeleri içerir. Veri setleri, tek ve çok modlu kaynak kısıtlı proje çizelgeleme problemlerinin çözüm prosedürlerinin değerlendirilmesi için kullanılabilir. Örnekler, standart proje oluşturucu ProGen tarafından oluşturulmuştur.

Mevcut kütüphanede rastgele örnekler, proje çizelgeleme problemleri için bir faaliyet ağı örnek oluşturucusu olan RanGen2 kullanılarak üretildi. Ancak RanGen2, klasik kaynak kısıtlamaları, sabit çalışma süreleri ve bitiş-başlangıç önceliği ilişkileri ile proje çizelgeleme problemi için örnekler üretir. Aktiviteleri ve aktiviteler arasında farklı öncüllük ilişkileri gerçekleştirmek ve değişken yoğunluk formülü kullanmak için oluşturulan örnekler aşağıdaki gibi değiştirilmiştir:

- Bitiş-başlangıç öncüllük ilişkilerinin belirli bir kısmı rastgele seçilir ve %20 örtüşme ile besleme öncüllüklerine dönüştürülür (yani $g_p = 0{,}2$ ve $q_p = 0{,}2$).
- Literatürde mevcut veri setinde bir faaliyetin tamamlanma süresi faaliyetin tamamlanması için gereken an küçük süre ($Süre_i$) olarak kabul edilir ve bu faaliyetin en fazla tamamlanma oranı $B_{max}(i) = \frac{1}{Süre_i}$ şeklinde elde edilir.
- Bir faaliyetin bir dönemde en az tamamlanma oranı $b_{min}(i)$'nin her zaman periyodunda, 0.05'ye eşit olduğu varsayılmıştır ($b_{min}(i) = 0.05$).
- j30.7, j60.7, j90,7 kütüphane bilgilerinden seçilmiş ve veri modelinde açıklanan şekilde gerekli değişiklikler yapılmıştır. Veri setleri, j30 Çizelge 4.1'de, j60 EK1'de verilmiştir.

Çizelge 4.1. Uygun j30 veri seti

İş No	b_min	B_max	örtüşme	Başlayabilecek dönem	Termin tarihi	Ufuk : 162			Her dönem mevcut kaynakların miktarı : R1=22, R2=26, R3=24, R4=21				Ufuk : 162					
						Ardıllar			Gerekli kaynaklar				Öncüller					
						1	**2**	**3**	**1**	**2**	**3**	**4**	No	Tür	No	Tür	No	Tür
0	0	0,00	0,2	0	0	1	2	3	0	0	0	0						
1	0,1	0,25	0,2	1	10	7	11		4	5	0	10	0	1				
2	0,1	0,13	0,2	2	15	5	13	22	5	3	6	4	0	1				
3	0,1	0,13	0,2	3	20	4	6	11	2	5	0	10	0	1				
4	0,1	0,25	0,2	5	25	5	14	15	3	7	7	0	3	1				
5	0,1	1,00	0,2	7	30	8	9	21	9	0	1	5	2	1	4	2		
6	0,1	0,14	0,2	9	35	8	9	10	4	9	7	0	3	1				
7	0,1	0,11	0,2	11	40	10	13	18	2	0	0	10	1	1				
8	0,1	0,33	0,2	13	45	17	23	24	4	0	0	4	5	2	6	4		
9	0,1	0,13	0,2	15	50	17	23		8	7	0	6	5	2	6	4		
10	0,1	0,20	0,2	17	55	19	25		3	5	0	0	6	1	7	3		
11	0,1	0,50	0,2	19	60	12	20	21	0	6	10	8	1	4	3	1		
12	0,1	0,10	0,2	21	65	15	18	19	9	1	5	0	11	2				
13	0,1	0,33	0,2	23	70	15	16		7	9	0	9	2	4	7	1		
14	0,1	0,33	0,2	25	75	19	20	26	0	0	1	4	4	4				
15	0,1	0,13	0,2	27	80	24	25	27	7	0	0	3	4	4	12	1	13	1
16	0,1	0,25	0,2	29	85	17	21	24	7	5	8	1	13	1				
17	0,1	0,14	0,2	31	90	20	27		4	4	5	3	8	1	9	1	16	1
18	0,1	0,10	0,2	33	95	22	26	29	8	10	6	7	7	4	12	2		
19	0,1	0,25	0,2	35	100	22			9	9	6	5	10	1	12	2	14	4
20	0,1	0,14	0,2	37	105	28			3	0	3	9	11	1	14	1	17	2
21	0,1	0,25	0,2	39	110	23	25	27	5	5	2	1	5	4	11	1	16	2
22	0,1	1,00	0,2	41	115	30			8	0	9	0	18	4	19	4		
23	0,1	0,10	0,2	43	120	29			0	1	7	4	8	4	9	3	21	1
24	0,1	0,13	0,2	45	125	30			4	10	10	9	8	4	15	1	16	1
25	0,1	0,25	0,2	47	130	26			0	10	0	0	10	1	15	1	21	4
26	0,1	0,13	0,2	49	135	28			0	2	7	6	14	1	18	4	25	4
27	0,1	0,50	0,2	51	140	28	29	30	1	7	3	4	15	1	17	3	21	3
28	0,1	0,50	0,2	53	145	31			4	2	4	6	20	1	26	3	27	3
29	0,1	0,17	0,2	55	150	31			6	3	1	0	18	1	23	3	27	4
30	0,1	0,50	0,2	57	162	31			0	4	7	4	22	2	24	3	27	4
31	0	0,00	0,2	59	0				0	0	0	0	28	1	29	1	30	2

4.2.2. Parametre Optimizasyonu

Parametre optimizasyonu için popülasyon büyüklüğü, çaprazlama oranı ve mutasyon oranı parametreleri için aşağıdaki değerlerin yer aldığı tüm kombinasyonlar Çizelge 4.2 de verilmiş olup, her bir kombinasyon için yapılan tekrarların ortalama değerleri arasından en iyi parametre kombinasyonu elde edilmiştir.

POP	P_c	P_m
25	**0.7**	**0.01**
50	**0.8**	**0.05**
100	**0.9**	**0.10**

Çizelge 4.2'de gösterildiği gibi, her bir veri seti için kodun beş kez tekrar edilmesi ile elde edilen uygunluk fonksiyonu sonuçları OF1-OF5 olarak verilmiş, OF1-OF5 değerlerinin ortalamaları arasında sonuç olarak en küçük değer dikkate alınmıştır. Sonuca göre 13. satırdaki parametre seti optimal olarak seçilmiştir. (POP= 50, P_c = 0.8, P_m= 0.01).

Çizelge 4.2. Parametre optimizasyonu

				Tekrarlı Sonuçlar					
	POP	P_c	P_m	**OF1**	**OF2**	**OF3**	**OF4**	**OF5**	**Ortalama**
1	25	0,7	0,01	2825	2709	2789	2664	2687	2735
2	25	0,7	0,05	3366	3356	2833	2658	2586	2960
3	25	0,7	0,1	3158	3135	2829	2708	2703	2907
4	25	0,8	0,01	2640	2664	2661	2602	2626	2639
5	25	0,8	0,05	2633	2633	2663	2640	2587	2631
6	25	0,8	0,1	2750	2689	2718	2764	2658	2716
7	25	0,9	0,01	2700	2669	2608	2613	2664	2651
8	25	0,9	0,05	2722	2768	2733	2663	2723	2722
9	25	0,9	0,1	2770	2808	2905	2780	3022	2857
10	50	0,7	0,01	2801	2728	2598	2615	2910	2730
11	50	0,7	0,05	2802	2674	2607	2625	2586	2659
12	50	0,7	0,1	2624	2628	2794	2732	2727	2701
13	**50**	**0,8**	**0,01**	**2639**	**2586**	**2604**	**2617**	**2638**	**2617**
14	50	0,8	0,05	2635	2617	2672	2618	2643	2637
15	50	0,8	0,1	2688	2658	2632	2688	2636	2660
16	50	0,9	0,01	2647	2624	2603	2632	2589	2619
17	50	0,9	0,05	2657	2723	2841	2844	2804	2774
18	50	0,9	0,1	2596	3083	3268	3268	2800	3003
19	100	0,7	0,01	2725	2590	2676	2592	2571	2631
20	100	0,7	0,05	2709	2719	2701	2620	2761	2702
21	100	0,7	0,1	2650	2620	2569	2634	2655	2626
22	100	0,8	0,01	2593	2635	2585	2679	2596	2618
23	100	0,8	0,05	2718	2722	2680	2653	2642	2683
24	100	0,8	0,1	2580	2599	2631	2834	2593	2647
25	100	0,9	0,01	2654	2645	2737	2798	2983	2763
26	100	0,9	0,05	2738	2689	2567	2987	2875	2771
27	100	0,9	0,1	2671	2811	2976	2891	1879	2646

4.2.3. Sonuçlar

Çizelge 4.3'de önceki bölümde açıklanan veri kümelerinin karşılaştırmalı test sonuçları verilmiştir. Küçük veri kümelerinde, Genetik Algoritmanın yaklaşık 50 iterasyonundan sonra matematiksel model ile aynı sonucu verdiği görülmektedir. Kullanılan veri setine bağlı olarak amaç fonksiyonundaki $\boldsymbol{\alpha}$ değeri 100 olarak alınmıştır.

Ayrıca, veri kümesinde b_{min} küçükse, yani faaliyetlerin dönemlere yayılmasına izin veriliyorsa, programlamanın alternatif sonuçları olması muhtemeldir ve matematiksel modelin sonucuna yaklaşmak için daha fazla iterasyon gerekir.

Çizelge 4.3. Genetik Algoritma ve matematiksel modelin sonucu

iş sayısı	**Matematiksel Model**			**Sezgisel Model (tek noktalı çaprazlama) 50 iterasyon**		
	Amaç fonksiyonu Min $\left[100 \cdot C_{max} + \sum_{k=1}^{K}\sum_{t=1}^{d}(u_{kt})^2\right]$	Kodun çalışma süresi (san)	$\boldsymbol{C_{max}}$ (dönem)	**Amaç fonksiyonu** Min $\left[100 \cdot C_{max} + \sum_{k=1}^{K}\sum_{t=1}^{d}(u_{kt})^2\right]$	Kodun çalışma süresi (san)	$\boldsymbol{C_{max}}$ (dönem)
6	1700	0,16	9	1700	59	9
7	1757	0,19	9	1757	60	9
8	1983	0,24	10	1983	75	10
9	2366	0,15	12	2366	181	12
10	2374	0,19	13	2374	267	13
11	2754	0,24	14	2754	321	14

Tüm proje çalışması boyunca yukarıdaki veri kümelerinin kaynak kullanımı (Çizelge 4.4 - Çizelge 4.8)'te gösterilmektedir.

Elde edilen sonuca göre maksimum 11 işten oluşan bir projede geliştirilmiş matematiksel model, yoğunluk değişken formülasyonu ve dört çeşit öncüllük koşulu kullanarak en uygun çözümü hızlı bir şekilde bulur. Genetik algoritma ise daha uzun süre çalışarak aynı sonucu bulabilmektedir. Proje faaliyetlerinin sayısı 11 den fazla olduğunda matematiksel model verilen sürede herhangi bir çözüm üretememekte iken, genetik algoritma geçerli bir çözüm bulabilmektedir.

Bu çalışmada, genetik algoritmanın durdurma kriteri iterasyon sayısı olarak belirlenir. Durdurma kriterine göre elde edilen sonuç optimal olmayabilir; ancak optimal sonuca yakındır. Ayrıca, genetik algoritma her iterasyonda geçerli bir çözüm üretebilmekte, ancak, bu en iyi çözüm olmasa da, büyük ölçekli projelerde uygun bir çözüm olabilmektedir.

j30, j60, j90 veri setleri sadece genetik algoritma ile test edilmiş olup, 2 tip çaprazlama yaklaşımı kullanılmıştır. İlk aşamada tek nokta çaprazlama kullanılırken, ikinci aşamada çaprazlama yönteminin etkisini anlamak için çift noktalı çaprazlama yöntemi kullanılmıştır.

Elde edilen sonuçlara göre çift noktalı çaprazlamanın hesaplama süresini kısalttığı görülmektedir (Çizelge 4. 9).

Çizelge 4.4. Proje çalışması boyunca kaynak kullanımı (projenin iş sayısı :7)

İş sayısı	İş 1: bmin = 0,2, Bmax = 0,5, kaynak kullanımı (k1:5, k2: 5,k3:10,k4:10) , İş 2: bmin = 0,4, Bmax = 0,5, (k1:4, k2: 5,k3:5,k4:10) , ön koşulu: iş 1 tip 1, İş 3: bmin = 0,2, Bmax = 0,4, (k1:5, k2:3,k3:6,k4:4) , ön koşulu: iş 2 tip 4, İş 4: bmin = 0,4, Bmax = 0,5, (k1:2, k2: 5,k3:0,k4:10) , ön koşulu: İş 3 tip 2, İş 5: bmin = 0,4, Bmax = 0,5, (k1:3, k2: 7,k3:7,k4:0) , ön koşulu: iş no 4 tip 1, İş 6: bmin = 0,4, Bmax = 0,5, (k1:9, k2: 0,k3:1,k4:5) , ön koşulu: 5 tip 2, İş 7: bmin = 0,4, Bmax = 0,5, (k1:7, k2: 6,k3:4,k4:10) , ön koşulu: 5 tip 2														
7	Kaynak 4	5	10	5	1,2	6,2	6,2	3,7	8,7	5					
	Kaynak 3	5	7,5	2,5	1,2	1,2	4,7	5,2	3,7	2					
	Kaynak 2	2,5	5	2,5	1,2	3,7	7,2	4,7	4,2	3					
	Kaynak 1	2,5	4,5	2	1,2	2,2	3,7	7,2	9,2	3,5					
	Dönem	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**

Çizelge 4.5. Proje çalışması boyunca kaynak kullanımı (projenin iş sayısı :8)

İş sayısı	İş 1: bmin = 0,2, Bmax = 0,5, kaynak kullanımı (k1:5, k2: 5,k3:10,k4:10) , İş 2: bmin = 0,4, Bmax = 0,5, (k1: 4, k 2: 5,k3:5,k4:10) , ön koşulu: iş 1 tip 1, İş 3: bmin = 0,2, Bmax = 0,4, (k1:5, k2:3,k3:6,k4:4) , ön koşulu: iş 2 tip 4, İş 4: bmin = 0,4, Bmax = 0,5, (k1:2, k2: 5,k3:0,k4:10) , ön koşulu: İş 3 tip 2, İş 5: bmin = 0,4, Bmax = 0,5, (k1:3, k2: 7,k3:7,k4:0) , ön koşulu: iş no 4 tip 1, İş 6: bmin = 0,4, Bmax = 0,5, (k1:9, k2: 0,k3:1,k4:5) ön kösülü 5 tip 2, İş 7: bmin = 0,4, Bmax = 0,5, (k1:7, k2: 6,k3:4,k4:10) , ön koşulu: 6 tip 2, İş 8: bmin = 0,4, Bmax = 0,5, (k1:8, k2: 10,k3:10,k4:10) , ön koşulu: 7 tip 4														
8	Kaynak 4	5	10	7	7	5	2,5	7,5	5	5	5				
	Kaynak 3	5	7,5	5,5	3	3,5	4	2,5	2	5	5				
	Kaynak 2	2,5	5	4	4	6	3,5	3	3	5	5				
	Kaynak 1	2,5	4,5	4,5	3,5	2,5	6	8	3,5	4	4				
	Dönem	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**

Çizelge 4.6. Proje çalışması boyunca kaynak kullanımı (projenin iş sayısı :9)

İş sayısı	İş 1: bmin = 0,2, Bmax = 0,5, kaynak kullanımı (k1:5, k2: 5,k3:10,k4:10) , İş 2: bmin = 0,4, Bmax = 0,5, (k1:4, k2: 5,k3:5,k4:10) , ön koşulu: iş 1 tip 1, İş 3: bmin = 0,2, Bmax = 0,4, (k1:5, k2:3,k3:6,k4:4) , ön koşulu: 2 tip 4, İş 4: bmin = 0,4, Bmax = 0,5, (k1:2, k2: 5,k3:0,k4:10) , ön koşulu: İş 3 tip 2, İş 5: bmin = 0,4, Bmax = 0,5, (k1:3, k2: 7,k3:7,k4:0) , ön koşulu: iş 4 tip 1, İş 6: bmin = 0,4, Bmax = 0,5, (k1:9, k2: 0,k3:1,k4:5) , ön koşulu: 5 tip 2, İş 7: bmin = 0,4, Bmax = 0,5, (k1:7, k2: 6,k3:4,k4:10) , ön koşulu: 6 tip 2, İş 8: bmin = 0,4, Bmax = 0,5, (k1:7, k2: 6,k3:4,k4:10) , ön koşulu: 7 tip 2, İş 9: bmin = 0,4, Bmax = 0,5, (k1:8, k2:10,k3:10,k4:10) , ön koşulu: 8 tip3														
9	Kaynak 4	5	10	7	7	5	2,5	7,5	5	5	5	5	5		
	Kaynak 3	5	7,5	5,5	3	3,5	4	2,5	2	5	5	5	5		
	Kaynak 2	2,5	5	4	4	6	3,5	3	3	5	5	5	5		
	Kaynak 1	2,5	4,5	4,5	3,5	2,5	6	8	3,5	4	4	4	4		
	Dönem	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**

Çizelge 4.7. Proje çalışması boyunca kaynak kullanımı (projenin iş sayısı :10)

İş sayısı	İş 1: bmin = 0,2, Bmax = 0,5, kaynak kullanımı (k1:5, k2: 5,k3:6,k4:8) , İş 2: bmin = 0,4, Bmax = 0,5, (k1:5, k2: 5,k3:5,k4:5) ön koşulu iş 1 tip 1, İş 3: bmin = 0,2, Bmax = 0,4, (k1:6, k2:6,k3:6,k4:6) ön koşulu iş 2 tip 4, İş 4: bmin = 0,4, Bmax = 0,5, (k1:2, k2: 5,k3:3,k4:5) ön koşulu İş 3 tip 2, İş 5: bmin = 0,4, Bmax = 0,5, (k1:3, k2: 7,k3:7,k4:4) ön koşulu iş no 4 tip 2, İş 6: bmin = 0,4, Bmax = 0,5, (k1:5, k2: 4,k3:3,k4:5) ön koşulu 5 tip 2, İş 7: bmin = 0,4, Bmax = 0,5, (k1:7, k2: 6,k3:4,k4:6) ön koşulu 6tip 2, İş 8: bmin = 0,4, Bmax = 0,5, (k1:7, k2: 8,k3:8,k4:8) ön koşulu 7 tip 4, İş 9: bmin = 0,4, Bmax = 0,5, (k1:7, k2:6,k3:4,k4:6) ön koşulu 8 tip3, İş 10: bmin = 0,4, Bmax = 0,5, (k1:8, k2:8,k3:7,k4:7) ön koşulu 9 tip4														
10	Kaynak 4	4	6,5	2,5	5,6	4,4	5	5	6	4	3,6	6,4	3,5	3,5	
	Kaynak 3	3	5,5	2,5	4,8	5,9	5,3	3,2	3,8	4	2,4	5,6	3,5	3,5	
	Kaynak 2	2,5	5	2,5	5,6	5,9	5,9	4,6	6	4	3,6	6,4	4	4	
	Kaynak 1	2,5	5	2,5	4,4	3,9	4,5	5,5	4,7	3,5	4,2	6,3	4	4	
	Dönem	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**

Çizelge 4.8. Proje çalışması boyunca kaynak kullanımı (projenin iş sayısı :11)

İş sayısı	İş 1: bmin = 0,2, Bmax = 0,5, kaynak kullanımı (k1:5, k2: 5,k3:6,k4:8) , İş 2: bmin = 0,4, Bmax = 0,5, (k1:5, k2: 5,k3:5,k4:5) ön koşulu iş 1 tip 1, İş 3: bmin = 0,2, Bmax = 0,4, (k1:6, k2:6,k3:6,k4:6) ön koşulu iş 2 tip 4, İş 4: bmin = 0,4, Bmax = 0,5, (k1:2, k2: 5,k3:3,k4:5) ön koşulu İş 3 tip 2, İş 5: bmin = 0,4, Bmax = 0,5, (k1:3, k2: 7,k3:7,k4:4) ön koşulu iş no 4 tip 2, İş 6: bmin = 0,4, Bmax = 0,5, (k1:5, k2: 4,k3:3,k4:5) ön koşulu 5 tip 2, İş 7: bmin = 0,4, Bmax = 0,5, (k1:7, k2: 6,k3:4,k4:6) ön koşulu 6tip 2, İş 8: bmin = 0,4, Bmax = 0,5, (k1:7, k2: 8,k3:8,k4:8) ön koşulu 7 tip 4, İş 9: bmin = 0,4, Bmax = 0,5, (k1:7, k2:6,k3:4,k4:6) ön koşulu 8 tip1, İş 10: bmin = 0,4, Bmax = 0,5, (k1:8, k2:8,k3:7,k4:7) ön koşulu 9 tip4, İş 11: bmin = 0,2, Bmax = 0,3, (k1:10, k2:10,k3:10,k4:10) ön koşulu 8 tip3														
11	Kaynak 4	4	6,5	2,5	5,6	4,4	5	6	5	4	5	5	5,5	5,5	6
	Kaynak 3	3	5,5	2,5	4,8	5,9	5,3	3,8	3,2	4	4	4	5,5	5,5	6
	Kaynak 2	2,5	5	2	5,6	5,9	5,9	6	4,6	4	5	5	6	6	6
	Kaynak 1	2,5	5	2,5	4,4	3,9	4,5	4,7	5,5	3,5	5,5	5,5	6	6	5,5
	Dönem	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**

Çizelge 4.9. Tek nokta ve çift nokta çaprazlamalı Genetik Algoritmanın sonuçları

iş sayısı	Sezgisel Model (tek noktalı çaprazlama) 1 iterasyon			Sezgisel Model (çift noktalı çaprazlama) 1 iterasyon		
	Amaç fonksiyonu Min A	İşlem süresi (san)	$C_{\max}$ (dönem)	Amaç fonksiyonu Min A	İşlem süresi (san)	$C_{\max}$ (dönem)
j30	12304	212	111	10966	193	96
j60	25432	625	274	25971	413	276
j90	31248	1223	351	29458	874	325
iş sayısı	Sezgisel Model (tek noktalı çaprazlama) 50 iterasyon			Sezgisel Model (çift noktalı çaprazlama) 50 iterasyon		
	Amaç fonksiyonu Min A	İşlem süresi (san)	$C_{\max}$ (dönem)	Amaç fonksiyonu Min A	İşlem süresi (san)	$C_{\max}$ (dönem)
j30	10454	2918	83	10641	2144	85
j60	20567	7200	212	21354	4329	241
j90	27326	11160	279	26974	6278	263
iş sayısı	Sezgisel Model (tek noktalı çaprazlama) 100 iterasyon			Sezgisel Model (çift noktalı çaprazlama) 100 iterasyon		
	Amaç fonksiyonu Min A	İşlem süresi (san)	$C_{\max}$ (dönem)	Amaç fonksiyonu Min A	İşlem süresi (san)	$C_{\max}$ (dönem)
j30	10147	51248	67	10037	40482	66
j60	14853	14325	132	14190	11582	124
j90	20453	25032	193	20148	12354	185

A = $100 \cdot C_{max} + \sum_{k=1}^{K} \sum_{t=1}^{d} (u_{kt})^2$

5. TARTIŞMA VE SONUÇ

Bu çalışmada, Siparişe Göre Üretim (MTO) ve Siparişe Göre Mühendislik (ETO) ürünlerinin üretim planlamasında proje çizelgeleme yöntemi, değişken yoğunluklu formülasyon ve dört farklı türde öncül ve ardıl ilişkisi ile birlikte kullanılmıştır.

Amaç, proje süresini en aza indirirken kaynak tüketimini dengelemektir. Bu amaçla daha önce yapılan çalışmalardan yararlanılarak matematiksel bir model geliştirilmiştir. Geliştirilen matematiksel model ile küçük ölçekli projelerde kesin çözüme gidilebilirken, orta ve büyük projeler için modelin NP zor olması nedeniyle sonuç elde edilememekte olduğundan, bu durumlar için bir genetik algoritma önerilmiştir.

Modelin kendine özgü yapısı ve değişken yoğunluk kavramının kullanılması nedeniyle yeni bir kromozom yapısı önerilmiştir.

Elde edilen sonuçlara göre, önerilen algoritma küçük veri setinde matematiksel model ile aynı sonucu elde etmekte ancak genetik algoritmanın küçük veri setinde sonuçları elde etmesi daha fazla zaman almaktadır. Projenin faaliyet sayısı az ama öncüllük ilişkileri karmaşık ise (bir aktivitenin kaç öncülü varsa veya b_{min} çok küçükse) genetik algoritmanın hesaplama süresi daha uzun sürebilmektedir. Uzun süren çalışma zamanını kısaltmak adına çalışmanın son aşamasında, genetik algoritma uygulamasında çift noktalı çaprazlama yaklaşımı kullanılmış olup, sonuçlara göre, çift noktalı çaprazlamanın, tek noktalı çaprazlamaya göre daha hızlı çalıştığı gösterilmiştir.

KAYNAKLAR

Alfieri, A., Tolio, T., Urgo, M. 2011. A project scheduling approach to production planning with feeding precedence relations. International Journal of Production Research, 49(4):, 995–1020. https://doi.org/10.1080/00207541003604844

Alfieri, A., Tolio, T., Urgo, M. 2012. A project scheduling approach to production and material requirement planning in Manufacturing-to-Order environments. Journal of Intelligent Manufacturing, 23(3):, 575–585. https://doi.org/10.1007/s10845-010-0396-1

Andrade, C. E., Silva, T., Pessoa, L. S. 2019. Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm. Expert Systems with Applications, 128:, 67–80. https://doi.org/10.1016/j.eswa.2019.03.007

Benjamin, V., Tabyang, T., Sookil, Nart. 2015. Precedence relationship options for the resource levelling problem using a genetic algorithm. Construction Management and Economics, 33(9):, 711–723. https://doi.org/10.1080/01446193.2015.1100317

Bianco, L., Caramia, M. 2011. Minimizing the completion time of a project under resource constraints and feeding precedence relations: A Lagrangian relaxation based lower bound. 4or, 9(4):, 371–389. https://doi.org/10.1007/s10288-011-0168-6

Brucker, P., Knust, S., Schoo, A., Thiele, O. 1998. A branch and bound algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research, 107(2):, 272–288. https://doi.org/10.1016/S0377-2217(97)00335-4

Chen, R., Liang, C., Gu, D., Leung, J. Y. T. 2017. A multi-objective model for multi-project scheduling and multi-skilled staff assignment for IT product development considering competency evolution. International Journal of Production Research, 55(21):, 6207–6234. https://doi.org/10.1080/00207543.2017.1326641

Damci, A., Arditi, D., Polat, G. 2013. Resource Leveling in Line-of-Balance Scheduling. Computer-Aided Civil and Infrastructure Engineering, 55(21):, 6207–6234. https://doi.org/ 10.1111/mice.12038

De Reyck, B., Herroelen, W. 1998. A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations. European Journal of Operational Research, 679–692. https://doi.org/10.1016/S0377-2217(97)00305-6

Ghiyasinasab, M., Lehoux, N., Ménard, S., Cloutier, C. 2021. Production planning and

project scheduling for engineer-to-order systems- case study for engineered wood production. International Journal of Production Research, 59(4):, 1068–1087. https://doi.org/10.1080/00207543.2020.1717009

Hung, Y. F., Leachman, R. C. 1996. A production planning methodology for semiconductor manufacturing based on iterative simulation and linear programming calculations. IEEE Transactions on Semiconductor Manufacturing, 9(2):, 257–269. https://doi.org/10.1109/66.492820

Jia, Q., Seo, Y. 2013. An improved particle swarm optimization for the resource-constrained project scheduling problem. International Journal of Advanced Manufacturing Technology, 67(9–12):, 2627–2638. https://doi.org/10.1007/s00170-012-4679-x

Kazemi, S., Davari-Ardakani, H. 2020. Integrated resource leveling and material procurement with variable execution intensities. Computers and Industrial Engineering, 148(July):, 106673. https://doi.org/10.1016/j.cie.2020.106673

Kis, T. 2005. A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. Mathematical Programming, 103(3):, 515–539. https://doi.org/10.1007/s10107-004-0551-6

Kleiny, R. 2000. Project scheduling with time-varying resource constraints. International Journal of Production Research ISSN 0020±7543 print/ISSN 1366±588X (Vol. 38, 3937-3952). https://doi:/10.1080/00207540050176094

Kolisch, R., Hartmann, S. 2005. Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update. . Retrieved from www.som.wi.tum.dewww.bwl.uni-kiel.de/bwlinstitute/Prod/alumni/hartmann/hartmann.html

Li, H., Xiong, L., Liu, Y., Li, H. 2018. An effective genetic algorithm for the resource levelling problem with generalised precedence relations. International Journal of Production Research, 56(5):, 2054–2075. https://doi.org/10.1080/00207543.2017.1355120

Liess, O., Michelon, P. 2008. A constraint programming approach for the resource-constrained project scheduling problem. Annals of Operations Research, 157(1):, 25–36. https://doi.org/10.1007/S10479-007-0188-Y

Márkus, A., Váncza, J., Kis, T., Kovács, A. 2003. Project scheduling approach to production planning. CIRP Annals - Manufacturing Technology, 52(1):, 359–362. https://doi.org/10.1016/S0007-8506(07)60601-5

Neumann, K., Schwindt, C., Zimmermann, J. 2006. Resource-Constrained Project Scheduling with Time Windows. Perspectives in Modern Project Scheduling, 375–407. https://doi.org/10.1007/978-0-387-33768-5_15

Panwalkar, S. S., Koulamas, C. 2014. The two-machine no-wait general and proportionate open shop makespan problem. European Journal of Operational Research, 238(2):, 471–475. https://doi.org/10.1016/J.EJOR.2014.04.030

Pellerin, R., Perrier, N., Berthaut, F. 2020. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. European Journal of Operational Research, 280(2):, 395–416. https://doi.org/10.1016/j.ejor.2019.01.063

Ponz-Tienda, J. L., Salcedo-Bernal, A., Pellicer, E. 2017. A Parallel Branch and Bound Algorithm for the Resource Leveling Problem with Minimal Lags. Computer-Aided Civil and Infrastructure Engineering, 32(6):, 474–498. https://doi.org/10.1111/mice.12233

Quintanilla, S., Pérez, Á., Lino, P., Valls, V. 2012. Time and work generalised precedence relationships in project scheduling with pre-emption: An application to the management of Service Centres. European Journal of Operational Research, 219(1):, 59–72. https://doi.org/10.1016/j.ejor.2011.12.018

Riane, F., Artiba, A., Elmaghraby, S. E. 1998. A hybrid three-stage flowshop problem: Efficient heuristics to minimize makespan. European Journal of Operational Research, 109(2):, 321–329. https://doi.org/10.1016/S0377-2217(98)00060-5

Schwindt, C., Zimmermann, J. 2015a. Handbook on project management and scheduling vol. 2 (Vol. 2). https://doi.org/10.1007/978-3-319-05915-0

Schwindt, C., Zimmermann, J. 2015b. Handbook on project management and scheduling vol. 2 (Vol. 1). https://doi.org/10.1007/978-3-319-05915-0

Shtub, A., LeBlanc, L. J., Cai, Z. 1996. Scheduling programs with repetitive projects: A comparison of a simulated annealing, a genetic and a pair-wise swap algorithm. European Journal of Operational Research, 88(1):, 124–138. https://doi.org/10.1016/0377-2217(94)00158-8

Snauwaert, J., Vanhoucke, M. 2021. A new algorithm for resource-constrained project scheduling with breadth and depth of skills. European Journal of Operational Research, 292(1):, 43–59. https://doi.org/10.1016/j.ejor.2020.10.032

Thammano, A., Phu-ang, A. 2012. A hybrid evolutionary algorithm for the resource-constrained project scheduling problem. Artificial Life and Robotics, 17(2):, 312–316. https://doi.org/10.1007/s10015-012-0065-x

Tolio, T., Urgo, M. 2007. A Rolling Horizon Approach to Plan Outsourcing in Manufacturing-to-Order Environments Affected by Uncertainty. CIRP Annals - Manufacturing Technology, 56(1):, 487–490. https://doi.org/10.1016/J.CIRP.2007.05.116

Tritschler, M., Naber, A., Kolisch, R. 2017. A hybrid metaheuristic for resource-

constrained project scheduling with flexible resource profiles. European Journal of Operational Research, 262(1):, 262–273. https://doi.org/10.1016/j.ejor.2017.03.006

Ulusoy, G. 2015. Proje planlamada kaynak kısıtlı çizelgeleme , Yöneylem Araştırması - Halim Doğrusöz'e Armağan, M. Köksalan, N. Erkip (Editörler), 6. Bölüm, 89-128, Ankara, 2000.

Vanczal, J., Kis, T. 2003. Project Scheduling Approach to Production Planning. Manufacturing Technology. 52(1):359-362. https://doi.org/10.1016/S0007-8506(07)60601-5.

Volling, T., Matzke, A., Grunewald, M., Spengler, T. S. 2013. Planning of capacities and orders in build-to-order automobile production: A review. European Journal of Operational Research, 224(2):, 240–260. https://doi.org/10.1016/J.EJOR.2012.07.034

Yang, T., Li, J., Chen, H. 2018. Multi-objective flexible job-shop scheduling problems with limited resource constraints using nondominated sorting genetic algorithm II. 2018 IEEE International Conference on Information and Automation, ICIA 2018, (August):, 1127–1131. https://doi.org/10.1109/ICInfA.2018.8812599

Zamani, R. 2011. A hybrid decomposition procedure for scheduling projects under multiple resource constraints. Operational Research, 11(1):, 93–111. https://doi.org/10.1007/S12351-009-0073-3

Zennaro, I., Finco, S., Battini, D., Persona, A. 2019. Big size highly customised product manufacturing systems: a literature review and future research agenda. International Journal of Production Research, 57(15–16):, 5362–5385. https://doi.org/10.1080/00207543.2019.1582819

Ziarati, K., Akbari, R., Zeighami, V. 2011. On the performance of bee algorithms for resource-constrained project scheduling problem. Applied Soft Computing Journal, 11(4):, 3720–3733. https://doi.org/10.1016/J.ASOC.2011.02.002

EKLER

EK. 1. Uyumlu j60 veri seti

İş No	b_{min}	B_{max}	Örtüşme	Başlayabilecek dönem	Termin tarihi	Ufuk : 162			Her dönem mecüt kaynakların miktarı: R1=14, R2=15, R3=14, R4=12				Ufuk: 162					
						Ardıllar			Gerekli kaynaklar				Öncüller					
						1	2	3	1	2	3	4	No	Type	No	Type	No	Type
0	0,1	1,00	0,2	0	5	1	2	3	0	0	0	0						
1	0,1	0,25	0,2	2	9	31			0	3	0	10	0	1				
2	0,1	0,20	0,2	3	15	4	5	8	2	0	0	0	0	1				
3	0,1	0,13	0,2	4	18	6	7	8	0	9	2	0	0	1				
4	0,1	0,11	0,2	5	24	17	18	23	0	0	10	0	2	1				
5	0,1	0,10	0,2	6	30	9	15	16	10	0	6	0	2	2				
6	0,1	1,00	0,2	7	32	10	14	50	0	0	0	3	3	1				
7	0,1	0,14	0,2	8	33	25	27		0	9	0	0	3	4				
8	0,1	0,17	0,2	9	37	14	16	31	0	0	7	0	2	1	3	2		
9	0,1	0,11	0,2	10	41	19	24	25	3	0	2	5	5	2				
10	0,1	0,33	0,2	11	45	12	20	36	3	10	3	0	6	1				
11	0,1	0,25	0,2	12	49	13	16	21	5	8	5	0	10	1				
12	0,1	0,25	0,2	13	53	46	56	58	0	3	0	0	10	1				
13	0,1	0,50	0,2	14	57	18	28		3	7	5	9	11	1				
14	0,1	0,13	0,2	15	61	18	26		0	4	0	9	6	1	8	1		
15	0,1	0,11	0,2	16	65	22	23	26	2	7	0	4	5	2		0		
16	0,1	0,17	0,2	17	69	33	37	54	10	0	9	0	5	1	8	2	11	2
17	0,1	0,17	0,2	18	73	21	25	39	5	2	0	0	4					
18	0,1	1,00	0,2	19	77	39	42	45	7	8	7	1	4	1	13	1	14	1
19	0,1	0,13	0,2	20	81	22	28	48	0	5	0	0	9	4				
20	0,1	0,14	0,2	21	85	33	38		6	8	0	0	19	1				
21	0,1	0,50	0,2	22	89	26	29	35	3	0	0	0	11	1	17	2		
22	0,1	0,50	0,2	23	95	49			0	2	0	0	15	2	19	2		
23	0,1	0,20	0,2	24	110	37	51		9	4	0	10	4	1	15	2		
24	0,1	0,17	0,2	25	115	29	35	36	7	0	0	6	9	1				
25	0,1	0,13	0,2	26	120	30	32	38	4	0	6	0	7	3	9	1	17	1
26	0,1	0,20	0,2	27	125	34	44	59	0	10	5	0	14	1	15	2	21	2
27	0,1	0,11	0,2	28	130	35	40	53	4	2	0	0	7	1				
28	0,1	0,33	0,2	29	135	34	36	44	0	0	6	1	13	1	19	1		
29	0,1	0,20	0,2	30	140	32	40	47	0	0	0	1	21	2	24	4		
30	0,1	0,17	0,2	31	145	31	43	52	5	4	1	3	25	4				

EK. 1. Uyumlu j60 veri seti (devam)

İş No	b_{min}	B_{max}	Örtüşme	Başlayabilecek dönem	Termin tarihi	Ufuk : 162			Her dönem mecüt kaynakların miktarı: R1=14, R2=15, R3=14, R4=12				Ufuk: 162							
						Ardıllar			Gerekli kaynaklar				Öncüller							
						1	2	3	1	2	3	4	No	Type	1	2	3	Type	No	Type
31	0,1	0,20	0,2	32	150	41	45	48	0	0	0	1	1	1	8	2	30	1		
32	0,1	1,00	0,2	33	155	49			0	5	3	9	25	1	29	1				
33	0,1	0,50	0,2	34	160	34	42	43	2	0	0	4	16	1	20	1				
34	0,1	0,20	0,2	35	165	52	60		0	7	0	0	26	2	28	3				
35	0,1	0,13	0,2	36	170	37	50		0	1	5	0	21	1	24	1	27	1		
36	0,1	0,13	0,2	37	175	43	45		0	4	0	0	10	1	24	2	28	1	33	4
37	0,1	0,11	0,2	38	180	52			10	7	0	0	16	1	23	2	35	3		
38	0,1	0,11	0,2	39	185	41	42	44	2	0	3	1	20	1	25	3				
39	0,1	0,25	0,2	40	190	40	41	60	1	0	7	0	17	1	18	1				
40	0,1	0,20	0,2	41	195	55	56		9	0	0	0	27	1	29	2	39	3		
41	0,1	0,25	0,2	42	200	49			0	0	0	10	31	1	38	1	39	4		
42	0,1	1,00	0,2	43	205	48			0	0	9	3	18	1	33	2	38	1		
43	0,1	1,00	0,2	44	210	59			4	3	0	0	30	1	33	1	36	1		
44	0,1	0,25	0,2	45	215	46	47		0	0	5	0	26	2	28	2	38	1		
45	0,1	0,50	0,2	46	220	47	51	53	0	7	0	0	18	2	31	2	33	1		
46	0,1	0,33	0,2	47	225	53	54		8	8	0	5	12	2	44	1				
47	0,1	0,14	0,2	48	230	54			4	9	1	0	29	1	44	1	45	2		
48	0,1	0,10	0,2	49	235	55			0	6	1	0	19	1	22	1	31	1		
49	0,1	1,00	0,2	50	240	58			0	0	4	5	32	2	41	2			42	1
50	0,1	1,00	0,2	51	245	51	57		9	1	4	0	6	1	35	2				
51	0,1	0,50	0,2	52	250	56			0	0	0	2	23	2	45	2	50	2		
52	0,1	0,33	0,2	53	255	55			0	5	0	5	30	1	34	2	37	2		
53	0,1	0,50	0,2	54	260	57			2	1	0	6	27	2	45	2	46	2		
54	0,1	0,13	0,2	55	265	57			9	9	0	0	16	2	46	2	47	2		
55	0,1	0,25	0,2	56	270	59			0	10	0	8	40	1	48	2	52	2		
56	0,1	0,33	0,2	57	275	59			5	6	0	5	12	2	40	2	51	2		
57	0,1	0,20	0,2	58	280	60			4	0	0	10	50	1	53	2				
58	0,1	0,14	0,2	59	285	61			7	0	0	0	12	2	49	2				
59	0,1	0,33	0,2	60	290	61			10	9	9	3	26	1	43	3	55	1		
60	0,1	0,20	0,2	61	295	61			0	7	9	0	34	2	39	3	56	4		

EK. 2. C# Genetik algoritma kodu

EK. 2.1 C# Kromozom oluşturma

class DNA (Kromozom oluşturma)

```
{
    public DNA() { MyData.Init(); }
    public void ReadData() { MyMethod.DataOku(); }// Read from JSON
    private static readonly Random random = new Random();
    private static double RandomNumberBetween(double minValue, double maxValue)
    { var next = random.NextDouble(); return minValue + (next * (maxValue - minValue)); }
    public void Initialpopulation()
    {
        int NumOfActivity = MyData.All.Activities.Count;
        int Horizen = MyData.All.HorizenPeriodNo;
        MyData.All.MinBmin = MyData.All.Activities[0].Bmin;
        for (int j = 1; j < NumOfActivity; j++)
        {
            if (MyData.All.Activities[j].Bmin <= MyData.All.MinBmin)
            { MyData.All.MinBmin = (MyData.All.Activities[j].Bmin); }
        }
        double h = (1 / MyData.All.MinBmin); h = Math.Round(h, 0);
        MyData.All.NoOfGenForActivities = (int)h; int hh = MyData.All.NoOfGenForActivities;
        for (int NoOfGenerateChromosome = 0; NoOfGenerateChromosome < 1; NoOfGenerateChromosome++)
        {
            MyData.All.GenerateChromos2.Clear(); MyData.All.GenerateChromos1.Clear();
            MyData.All.OkActCanSelect.Clear(); MyData.All.SelectedAct.Clear();
            int U2 = 0;
            for (int Res = 0; Res < MyData.All.Resources.Count; Res++)
            {
                MyData.All.Resources[Res].UsingResourcesAtPeriodT.Clear();
                MyData.All.Resources[Res].ResourcesAtPeriodTControl.Clear();
                for (int Period = 0; Period < Horizen; Period++)
                {
                    MyData.All.Resources[Res].UsingResourcesAtPeriodT.Add(0);
                    MyData.All.Resources[Res].ResourcesAtPeriodTControl.Add(0);
                }
```

```
} // Reset resource usage at startup
for (int j = 0; j < NumOfActivity; j++)
{
    MyData.All.Activities[j].GenDataChrom1.Clear();
    if (j == 0)
    {
        MyData.All.Activities[j].GenDataChrom2.Clear(); MyData.All.Activities[j].NoOfProcessingPeriod = 0;
        double Y = 1; double X = 0;
        for (var k = 0; k < h; k++)
        {
            double value = RandomNumberBetween(MyData.All.Activities[j].Bmin, MyData.All.Activities[j].Bmax);
            double RandPercentage = Math.Round(value, 4);
            if ((Y - RandPercentage) >= MyData.All.Activities[j].Bmin && (X + RandPercentage) <= 1)
            {
                MyData.All.Activities[j].GenDataChrom1.Add(Math.Round(RandPercentage, 1));
                X = X + RandPercentage; Y = Y - RandPercentage;
                MyData.All.Activities[j].NoOfProcessingPeriod++;
            }
            else if ((Y - RandPercentage) <= MyData.All.Activities[j].Bmin && X < 1)
            {
                if (X > 0)
                {
                    double XX = 1 - X;
                    MyData.All.Activities[j].GenDataChrom1.Add(Math.Round(XX, 1));
                    MyData.All.Activities[j].NoOfProcessingPeriod++;
                    X = X + XX; Y = Y - XX;
                }
                else if (X == 0 && Y == 1)
                {
                    MyData.All.Activities[j].GenDataChrom1.Add(MyData.All.Activities[j].Bmin);
                    MyData.All.Activities[j].NoOfProcessingPeriod++;
                    X = X + MyData.All.Activities[j].Bmin; Y = Y - MyData.All.Activities[j].Bmin;
                }
            }
            else if ((Y - RandPercentage) < 0 && X < 1)
            {
                MyData.All.Activities[j].GenDataChrom1.Add(MyData.All.Activities[j].Bmin);
                MyData.All.Activities[j].NoOfProcessingPeriod++;
                X = X + MyData.All.Activities[j].Bmin; Y = Y - MyData.All.Activities[j].Bmin;
            }
```

```
                    else if (X == 1) { MyData.All.Activities[j].GenDataChrom1.Add((0)); }
                    MyData.All.GenerateChromos1.Add(MyData.All.Activities[j].GenDataChrom1[k]);
                }
                int U = MyData.All.Activities[j].NoOfProcessingPeriod;
                int starttime = MyData.All.Activities[j].ReleaseT; int finishtime = MyData.All.Activities[j].DuedateT;
                for (var k = 0; k < h; k++)
                {
                    Random rnd = new Random();
                    int prossesingperiod = rnd.Next(starttime, (finishtime - (U - 1)));
                    if (U > 0)
                    {
                        MyData.All.Activities[j].GenDataChrom2.Add(prossesingperiod);
                        starttime = prossesingperiod + 1; U--;
                    }
                    else { MyData.All.Activities[j].GenDataChrom2.Add(0); }
                    MyData.All.GenerateChromos2.Add(MyData.All.Activities[j].GenDataChrom2[k]);
                }
                for (int Ss = 0; Ss < MyData.All.Activities[j].successors.Count; Ss++)
                {
                    MyData.All.OkActCanSelect.Add(MyData.All.Activities[j].successors[Ss]);
                }
                for (int ResourceType = 0; ResourceType < MyData.All.Resources.Count; ResourceType++)
                {
                    for (var kk = 0; kk < h; kk++)
                    {
                        double Percentresourcetype = 0;
                        Percentresourcetype = (MyData.All.Activities[j].GenDataChrom1[kk]) *
(MyData.All.Activities[j].Q[ResourceType]);
                        for (int Period = 0; Period < Horizen; Period++)
                        {
                            if (MyData.All.Activities[j].GenDataChrom2[kk] == Period)
                            {
                                MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] =
Math.Round(((MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period]) + Percentresourcetype), 2);
                                MyData.All.Resources[ResourceType].ResourcesAtPeriodTControl[Period] =
MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period];
                            }
                        }
                    }
                }
```

```
                    }
                    else
                    {
                        MyData.All.OkActCanSelect.Sort();
                        double SelecetedActivity = MyData.All.OkActCanSelect.Min();
                        MyData.All.SelectedAct.Add(SelecetedActivity); MyData.All.OkActCanSelect.Remove(SelecetedActivity);
                        for (int Ss = 0; Ss < MyData.All.Activities[j].successors.Count; Ss++)
                        {
                            if ((!MyData.All.SelectedAct.Contains(MyData.All.Activities[j].successors[Ss])) &&
(!MyData.All.OkActCanSelect.Contains(MyData.All.Activities[j].successors[Ss])))
                            {
                                MyData.All.OkActCanSelect.Add(MyData.All.Activities[j].successors[Ss]);
                            }
                        }
                        MyData.All.Activities[j].NoOfProcessingPeriod = 0;
                        double X = 0; double Y = 1;
                        for (var k = 0; k < h; k++)
                        {
                            double value = RandomNumberBetween(MyData.All.Activities[j].Bmin, MyData.All.Activities[j].Bmax);
                            double RandPercentage = Math.Round(value, 2);
                            if ((Y - RandPercentage) >= MyData.All.Activities[j].Bmin && (X + RandPercentage) <= 1)
                            {
                                MyData.All.Activities[j].GenDataChrom1.Add(Math.Round(RandPercentage, 1));
                                X = X + RandPercentage; Y = Y - RandPercentage;
                                MyData.All.Activities[j].NoOfProcessingPeriod++;
                            }
                            else if ((Y - RandPercentage) <= MyData.All.Activities[j].Bmin && X < 1)
                            {
                                if (X > 0)
                                {
                                    double XX = 1 - X;
                                    MyData.All.Activities[j].GenDataChrom1.Add(Math.Round(XX, 1));
                                    MyData.All.Activities[j].NoOfProcessingPeriod++; X = X + XX; Y = Y - XX;
                                }
                                else if (X == 0 && Y == 1)
                                {
                                    MyData.All.Activities[j].GenDataChrom1.Add(MyData.All.Activities[j].Bmin);
                                    MyData.All.Activities[j].NoOfProcessingPeriod++;
                                    X = X + MyData.All.Activities[j].Bmin; Y = Y - MyData.All.Activities[j].Bmin;
                                }
```

```
    }
    else if ((Y - RandPercentage) < 0 && X < 1)
    {
        MyData.All.Activities[j].GenDataChrom1.Add(MyData.All.Activities[j].Bmin);
        MyData.All.Activities[j].NoOfProcessingPeriod++;
        X = X + MyData.All.Activities[j].Bmin; Y = Y - MyData.All.Activities[j].Bmin;
    }
    else if (X == 1)
    { MyData.All.Activities[j].GenDataChrom1.Add((0)); }
    MyData.All.GenerateChromos1.Add(MyData.All.Activities[j].GenDataChrom1[k]);
}
int U = MyData.All.Activities[j].NoOfProcessingPeriod;
int starttime = MyData.All.Activities[j].ReleaseT; int finishtime = MyData.All.Activities[j].DuedateT;
int finishtime3 = 0;
foreach (var Preact in MyData.All.Activities[j].FeedingRelations.FindAll(l => l.TypeOfprecedence == "1"))
{
    double Percent = 0;
    int indexofPreAct = MyData.All.Activities[j].FeedingRelations.IndexOf(Preact);
    int PredesenseOFAct = MyData.All.Activities[j].FeedingRelations[indexofPreAct].PreActivities;
    if (MyData.All.Activities[PredesenseOFAct].GenDataChrom2.Count != 0)
    {
        double StartPriodOfPreAct = 0;
        for (var kk = 0; kk < h; kk++)
        {
            Percent = Percent + MyData.All.Activities[PredesenseOFAct].GenDataChrom1[kk];
            StartPriodOfPreAct = MyData.All.Activities[PredesenseOFAct].GenDataChrom2[kk];
            if (Percent >= MyData.All.q1) { break; }
        }
        if (starttime <= StartPriodOfPreAct) { starttime = (int)StartPriodOfPreAct + 1; }
    }
}
foreach (var Preact in MyData.All.Activities[j].FeedingRelations.FindAll(l => l.TypeOfprecedence == "2"))
{
    int indexofPreAct = MyData.All.Activities[j].FeedingRelations.IndexOf(Preact);
    int PredesenseOFAct = MyData.All.Activities[j].FeedingRelations[indexofPreAct].PreActivities;
    double startofpreAct2 = 0;
    if (MyData.All.Activities[PredesenseOFAct].GenDataChrom2.Count != 0)
    {
        startofpreAct2 = MyData.All.Activities[PredesenseOFAct].GenDataChrom2[0];
        int indisFirstGen2Actj = ((j * hh));
```

```
        if (starttime <= startofpreAct2)
        {
            starttime = (int)startofpreAct2 + 1;
        }
    }
}
foreach (var Preact in MyData.All.Activities[j].FeedingRelations.FindAll(l => l.TypeOfprecedence == "3"))
{
    double Percent = 0;
    int indexofPreAct = MyData.All.Activities[j].FeedingRelations.IndexOf(Preact);
    int PredesenseOFAct = MyData.All.Activities[j].FeedingRelations[indexofPreAct].PreActivities;
    if (MyData.All.Activities[PredesenseOFAct].GenDataChrom2.Count != 0)
    {
        double percentprocessofPreAct = 0;
        for (var kk = 0; kk < h; kk++)
        {
            Percent = Percent + MyData.All.Activities[PredesenseOFAct].GenDataChrom1[kk];
            percentprocessofPreAct = MyData.All.Activities[PredesenseOFAct].GenDataChrom2[kk];
            if (Percent >= MyData.All.q3) { break; }
        }
        finishtime3 = (int)percentprocessofPreAct + 1;
    }
}
foreach (var Preact in MyData.All.Activities[j].FeedingRelations.FindAll(l => l.TypeOfprecedence == "4"))
{
    double Percent = 0;
    int indexofPreAct = MyData.All.Activities[j].FeedingRelations.IndexOf(Preact);
    int PredesenseOFAct = MyData.All.Activities[j].FeedingRelations[indexofPreAct].PreActivities;
    if (MyData.All.Activities[PredesenseOFAct].GenDataChrom2.Count != 0)
    {
        double percentprocessofPreAct = 0;
        for (var kk = 0; kk < h; kk++)
        {
            Percent = Percent + MyData.All.Activities[PredesenseOFAct].GenDataChrom1[kk];
            percentprocessofPreAct = MyData.All.Activities[PredesenseOFAct].GenDataChrom2[kk];
            if (Percent >= 1) { break; }
        }
        if (starttime <= percentprocessofPreAct)
        {
            starttime = (int)percentprocessofPreAct + 1;
```

```
                    }
                }
            }
            int ResourceControl = 0; int Isnotok = 0;
            for (int Gen2isOK = 0; Gen2isOK < 900000000; Gen2isOK++)
            {
                if (Isnotok == 0 && ResourceControl > 0)
                {
                    for (int k = 0; k < h; k++) {
MyData.All.GenerateChromos2.Add(MyData.All.Activities[j].GenDataChrom2[k]); }
                    break;
                }
                if (Isnotok == 1)
                {
                    for (int ResourceType = 0; ResourceType < MyData.All.Resources.Count; ResourceType++)
                    {
                        for (int y = 0; y < MyData.All.HorizenPeriodNo; y++)
                        {
                            MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[y] =
MyData.All.Resources[ResourceType].ResourcesAtPeriodTControl[y];
                        }
                    }
                    U = U2; Isnotok = 0;
                }
                if (Isnotok == 0 && ResourceControl == 0)
                {
                    for (int ResourceType = 0; ResourceType < MyData.All.Resources.Count; ResourceType++)
                    {
                        for (int y = 0; y < MyData.All.HorizenPeriodNo; y++)
                        {
                            MyData.All.Resources[ResourceType].ResourcesAtPeriodTControl[y] =
MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[y];
                        }
                    }
                    U2 = U; int newstarttime = 0;
                    MyData.All.Activities[j].GenDataChrom2.Clear();
                    for (var k = 0; k < h; k++)
                    {
                        if (Isnotok != 0) { break; }
                        Random rnd = new Random();
```

```
                                if (U == 1 && k == 0)
                                {
                                    int prossesingperiod = rnd.Next(starttime, (MyData.All.Activities[j].DuedateT));
                                    MyData.All.Activities[j].GenDataChrom2.Add(prossesingperiod);
                                    starttime = prossesingperiod + 1; U--; newstarttime = prossesingperiod;
                                }
                                if (U > 1)
                                {
                                    int prossesingperiod = rnd.Next(starttime, (MyData.All.Activities[j].DuedateT - (U - 1)));
                                    MyData.All.Activities[j].GenDataChrom2.Add(prossesingperiod);
                                    starttime = prossesingperiod + 1; U--; newstarttime = prossesingperiod;
                                }
                                else if (U == 1 && k != 0)
                                {
                                    newstarttime = (int)MyData.All.Activities[j].GenDataChrom2[k - 1];
                                    if (newstarttime < finishtime3)
                                    { newstarttime = finishtime3 + 1; }
                                    else { newstarttime = newstarttime + 1; }
                                    int prossesingperiod = rnd.Next(newstarttime, (MyData.All.Activities[j].DuedateT));
                                    MyData.All.Activities[j].GenDataChrom2.Add(prossesingperiod); U--;
                                }
                                else { MyData.All.Activities[j].GenDataChrom2.Add(0); }
                            }
                            for (int ResourceType = 0; ResourceType < MyData.All.Resources.Count; ResourceType++)
                            {
                                if (Isnotok != 0) { break; }
                                for (var kk = 0; kk < h; kk++)
                                {
                                    if (Isnotok != 0) { break; }
                                    double Percentresourcetype = 0;
                                    Percentresourcetype = (MyData.All.Activities[j].GenDataChrom1[kk]) *
(MyData.All.Activities[j].Q[ResourceType]);
                                    for (int Period = 0; Period < MyData.All.Activities[j].DuedateT; Period++)
                                    {
                                        if (MyData.All.Activities[j].GenDataChrom2[kk] == Period)
                                        {
                                            MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] =
Math.Round(((MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period]) + Percentresourcetype), 2);
                                        }
```

```
                                            if (MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] <
(MyData.All.Resources[ResourceType].AvailableResourcesAtPeriodT[Period]))
                                            { ResourceControl = 1; }
                                            else if (MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] >
(MyData.All.Resources[ResourceType].AvailableResourcesAtPeriodT[Period]))
                                            {
                                                ResourceControl = 0; Isnotok = 1; break;
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

EK. 2.2 C# SelectionRulet

```
class SelectionRulet…….
{
    public SelectionRulet()
    {
        MyData.Init();
    }
    public void ReadData()              // JSON dosyasını okumak için
    {
        MyMethod.DataOku();
    }
    public void RulletSelection()
    {
        MyData.All.FirstparentD = 1;
        MyData.All.SecondparentD = 0;
        MyData.All.SumOfCmaxAllFeasible = 0;
        MyData.All.MaxCmax = 0;
        MyData.All.MaxCmax = MyData.All.FeasibleChromosome[0].MaxCmax;
            for (int d = 1;d < MyData.All.Population; d++)
            {
            if(MyData.All.FeasibleChromosome[d].MaxCmax> MyData.All.MaxCmax)
            {
                MyData.All.MaxCmax = MyData.All.FeasibleChromosome[d].MaxCmax;
            }
            }
        for (int j = 0; j < MyData.All.FeasibleChromosome.Count(); j++)
        {
            MyData.All.SumOfCmaxAllFeasible = MyData.All.SumOfCmaxAllFeasible + (MyData.All.MaxCmax -
MyData.All.FeasibleChromosome[j].MaxCmax);
        }
        double cumulative = 0;
        for (int j = 0; j < MyData.All.FeasibleChromosome.Count(); j++)
        {
            MyData.All.FeasibleChromosome[j].Probability = Math.Round(((MyData.All.MaxCmax-
(MyData.All.FeasibleChromosome[j].MaxCmax)) / (MyData.All.SumOfCmaxAllFeasible)), 5);
            MyData.All.FeasibleChromosome[j].CumulativeProbability = Math.Round((cumulative +
MyData.All.FeasibleChromosome[j].Probability), 5);
```

```
                cumulative = MyData.All.FeasibleChromosome[j].Probability + cumulative;
            }
            Random rnd = new Random();
            double PRO1 = RandomNumberBetween(0.02, 0.7);
            if (PRO1 < MyData.All.FeasibleChromosome[0].CumulativeProbability)
            {
                MyData.All.FirstparentD = 0;
            }
            for (int j = 1; j < MyData.All.FeasibleChromosome.Count(); j++)
            {
                if (PRO1 < MyData.All.FeasibleChromosome[j].CumulativeProbability && PRO1 >= MyData.All.FeasibleChromosome[j -
1].CumulativeProbability)
                {
                    MyData.All.FirstparentD = j;
                }
            }
            double PRO2 = RandomNumberBetween(0.001, 1);
            if (PRO2 < MyData.All.FeasibleChromosome[0].CumulativeProbability)
            {
                MyData.All.SecondparentD = 0;
            }
            for (int j = 1; j < MyData.All.FeasibleChromosome.Count(); j++)
            {
                if (PRO2 < MyData.All.FeasibleChromosome[j].CumulativeProbability && PRO2 >= MyData.All.FeasibleChromosome[j -
1].CumulativeProbability)
                {
                    MyData.All.SecondparentD = j;
                }
            }
        }
    }
}
```

EK. 2.3 C# CrossOver

```
    class CrossOver…….
    {
        public CrossOver()
        {
            MyData.Init();
        }

        public void CrossOverAndControl()
        {
            int PreTypeOk = 0;
            int Firstparent = MyData.All.FirstparentD;
            int Secondparent = MyData.All.SecondparentD;
            for (int parent = 0; parent <= 1; parent++)
            {
                MyData.All.ForCrossoverChrom[parent].ChromosomePart1.Clear();
                MyData.All.ForCrossoverChrom[parent].ChromosomePart2.Clear();
                MyData.All.ChildChromosome[parent].ChildChromosomePart1.Clear();
                MyData.All.ChildChromosome[parent].ChildChromosomePart2.Clear();
            }
            for (int Res = 0; Res < MyData.All.Resources.Count; Res++)
            {
                MyData.All.Resources[Res].UsingResourcesAtPeriodT.Clear();
                MyData.All.Resources[Res].ResourcesAtPeriodTControl.Clear();
                for (int Period = 0; Period < Horizen; Period++) /// Tam mevcut olan periyodlar 1 den horizene kadar
                {
                    MyData.All.Resources[Res].UsingResourcesAtPeriodT.Add(0);
                    MyData.All.Resources[Res].ResourcesAtPeriodTControl.Add(0);
                }
            }
            for (int k = 0; k < (h * PercentCrossoveFinal); k++)// ilk çocuk( 70 % ilk parent  %30 ikinci parent ) ikinci tam tarsi
            {
MyData.All.ForCrossoverChrom[0].ChromosomePart1.Add(MyData.All.FeasibleChromosome[Firstparent].ChromosomePart1[k]);
                MyData.All.ForCrossoverChrom[0].ChromosomePart2.Add(MyData.All.FeasibleChromosome[Firstparent].ChromosomePart2[k]);
                MyData.All.ForCrossoverChrom[1].ChromosomePart1.Add(MyData.All.FeasibleChromosome[Secondparent].ChromosomePart1[k]);
                MyData.All.ForCrossoverChrom[1].ChromosomePart2.Add(MyData.All.FeasibleChromosome[Secondparent].ChromosomePart2[k]);
            }
```

```
            for (int k = (h * PercentCrossoveFinal); k < (h * j); k++)
            {
                MyData.All.ForCrossoverChrom[0].ChromosomePart1.Add(MyData.All.FeasibleChromosome[Secondparent].ChromosomePart1[k]);
                MyData.All.ForCrossoverChrom[0].ChromosomePart2.Add(MyData.All.FeasibleChromosome[Secondparent].ChromosomePart2[k]);
                MyData.All.ForCrossoverChrom[1].ChromosomePart1.Add(MyData.All.FeasibleChromosome[Firstparent].ChromosomePart1[k]);
                MyData.All.ForCrossoverChrom[1].ChromosomePart2.Add(MyData.All.FeasibleChromosome[Firstparent].ChromosomePart2[k]);
            }
            for (int i = PercentCrossoveFinal; i < j; i++)
            {
                for (int parent = 0; parent <= 1; parent++)
                {
                    int ResourceControl = 0; int Isnotok = 0;
                    foreach (var Preact in MyData.All.Activities[i].FeedingRelations.FindAll(l => l.TypeOfprecedence == "1"))
                    {
                        double Percent = 0;
                        int indexofPreAct = MyData.All.Activities[i].FeedingRelations.IndexOf(Preact);
                        int PredesenseOFAct = MyData.All.Activities[i].FeedingRelations[indexofPreAct].PreActivities;
                        double StartPeriodOfPreAct = 0;
                        double StartPeriodOfActJ = 0;
                        int indisFirstGen2Actj = ((i * h));
                        StartPeriodOfActJ = MyData.All.ForCrossoverChrom[parent].ChromosomePart2[indisFirstGen2Actj];
                        for (var kk = 0; kk < h; kk++)
                        {
                            int v = (PredesenseOFAct * h) + kk;
                            Percent = Percent + MyData.All.ForCrossoverChrom[parent].ChromosomePart1[v];
                            StartPeriodOfPreAct = MyData.All.ForCrossoverChrom[parent].ChromosomePart2[v];
//StartPeriodOfActJ = MyData.All.AllChomosomes[NoOfChromosome].ChromosomePart2[indisFirstGen2Actj];
                            if (Percent >= MyData.All.q1) { break; }
                        }
                        for (int ResorceNotOk = 0; ResorceNotOk < 500000; ResorceNotOk++)
                        {
                            if (Isnotok == 0 && ResourceControl == 1) { break; }
                            if (ResourceControl == 0 && Isnotok == 1) // Resource was not Ok
                            {
                                Isnotok = 0;
                                for (int Res = 0; Res < MyData.All.Resources.Count; Res++)
                                {
                                    MyData.All.Resources[Res].UsingResourcesAtPeriodT.Clear();
                                    for (int Period = 0; Period < Horizen; Period++) /// Tam mevcut olan periyodlar 1 den horizene
kadar sıfırla
```

```
                {
                    MyData.All.Resources[Res].UsingResourcesAtPeriodT.Add(0);
                }
            }
            int start = (int)StartPeriodOfPreAct + 1;
            for (var n = 0; n < h; n++)
            {
                if (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[(i * h) + n] == 0)
                {
                    MyData.All.ForCrossoverChrom[parent].ChromosomePart2[(i * h) + n] = 0;
                }
                else
                {
                    MyData.All.ForCrossoverChrom[parent].ChromosomePart2[(i * h) + n] = StartPeriodOfPreAct + 2 +
n;

                }
            }
        }
        if (ResourceControl == 0 && Isnotok == 0)  // First time Resource control
        {
            if (StartPeriodOfPreAct < StartPeriodOfActJ)
            {
                PreTypeOk = PreTypeOk + 0;
            }
            if (StartPeriodOfActJ <= StartPeriodOfPreAct)
            {
                PreTypeOk = PreTypeOk + 1;

                for (var n = 0; n < h; n++)
                {
                    if (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[(i * h) + n] == 0)
                    {
                        MyData.All.ForCrossoverChrom[parent].ChromosomePart2[(i * h) + n] = 0;
                    }
                    else
                    {
                        MyData.All.ForCrossoverChrom[parent].ChromosomePart2[(i * h) + n] = StartPeriodOfPreAct + 1
+ n;

                    }
                }
```

```
                                PreTypeOk = 0;
                            }
                        }
                        for (int ResourceType = 0; ResourceType < MyData.All.Resources.Count; ResourceType++)
                        {
                            if (Isnotok != 0) { break; }
                            for (var kk = 0; kk < ((i * h) + h); kk++)
                            {
                                if (Isnotok != 0) { break; }
                                double percentresourcetype = 0;
                                percentresourcetype = (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[kk]) *
(MyData.All.Activities[i].Q[ResourceType]);
                                for (int Period = 0; Period < Horizen; Period++)
                                {
                                    if (MyData.All.ForCrossoverChrom[parent].ChromosomePart2[kk] == Period)
                                    {
                                        MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] =
Math.Round(((MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period]) + percentresourcetype), 2);
                                    }
                                    if (MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] <=
(MyData.All.Resources[ResourceType].AvailableResourcesAtPeriodT[Period]))
                                    {
                                        ResourceControl = 1;
                                    }
                                    else if (MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] >
(MyData.All.Resources[ResourceType].AvailableResourcesAtPeriodT[Period]))
                                    {
                                        ResourceControl = 0; Isnotok = 1; break;
                                    }
                                }
                            }
                        }
                    }
                    foreach (var Preact in MyData.All.Activities[i].FeedingRelations.FindAll(l => l.TypeOfprecedence == "2"))
                    {
                        double Percent = 0;
                        int indexofPreAct = MyData.All.Activities[i].FeedingRelations.IndexOf(Preact);
                        int PredesenseOFAct = MyData.All.Activities[i].FeedingRelations[indexofPreAct].PreActivities;//
                        double StartPriodOfActj = 0;
```

```
double StartPeriodOfPredeActJ = 0;
int indisFirstGen2ActPrej = ((PredesenseOFAct * h));
int percentcompletedpoitJ = 0;
for (var kk = 0; kk < h; kk++)
{
    int v = (i * h) + kk;
    Percent = Percent + MyData.All.ForCrossoverChrom[parent].ChromosomePart1[v];
    StartPriodOfActj = MyData.All.ForCrossoverChrom[parent].ChromosomePart2[v];
    StartPeriodOfPredeActJ = MyData.All.ForCrossoverChrom[parent].ChromosomePart2[indisFirstGen2ActPrej];
    percentcompletedpoitJ++;
    if (Percent >= MyData.All.q2) { break; }
}
for (int ResorceNotOk = 0; ResorceNotOk < 50000; ResorceNotOk++)
{
    if (Isnotok == 0 && ResourceControl == 1) { break; }
    if (ResourceControl == 0 && Isnotok == 1)
    {
        Isnotok = 0;
        for (int Res = 0; Res < MyData.All.Resources.Count; Res++)
        {
            MyData.All.Resources[Res].UsingResourcesAtPeriodT.Clear();
            for (int Period = 0; Period < Horizen; Period++)
            {
                MyData.All.Resources[Res].UsingResourcesAtPeriodT.Add(0);
            }
        }
        int start = (int)StartPeriodOfPredeActJ + 1;
        for (var n = 0; n < h; n++)
        {
            if (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[(i * h) + n] == 0)
            {
                MyData.All.ForCrossoverChrom[parent].ChromosomePart2[(i * h) + n] = 0;
            }
            else
            {
                if ((StartPeriodOfPredeActJ + 2 + n) < MyData.All.Activities[i].DuedateT)
                {
                    MyData.All.ForCrossoverChrom[parent].ChromosomePart2[(i * h) + n] = StartPeriodOfPredeActJ
+ 2 + n;
                }
```

```
                                        else
                                        {
                                            MyData.All.ForCrossoverChrom[parent].ChromosomePart2[(i * h) + n] = rnd.Next(start,
MyData.All.Activities[i].DuedateT);
                                            start = start + 1;
                                        }
                                    }
                                }
                            }
                            if (ResourceControl == 0 && Isnotok == 0)
                            {
                                if (StartPeriodOfPredeActJ < StartPriodOfActj) { PreTypeOk = PreTypeOk + 0; }
                                else
                                {
                                    PreTypeOk = PreTypeOk + 1;
                                    for (var nn = 0; nn < percentcompletedpoitJ; nn++)
                                    {
                                        MyData.All.ForCrossoverChrom[parent].ChromosomePart2[(i * h) + nn] =
MyData.All.FeasibleChromosome[Secondparent].ChromosomePart2[(i * h) + nn];
                                    }
                                    for (var n = percentcompletedpoitJ; n < h; n++)
                                    {
                                        if (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[(i * h) + n] == 0)
                                        {
                                            MyData.All.ForCrossoverChrom[parent].ChromosomePart2[(i * h) + n] = 0;
                                        }
                                        else
                                        {
                                            MyData.All.ForCrossoverChrom[parent].ChromosomePart2[(i * h) + n] = StartPeriodOfPredeActJ
+ 1 + n;
                                        }
                                    }
                                }
                            }
                            for (int ResourceType = 0; ResourceType < MyData.All.Resources.Count; ResourceType++)
                            {
                                {
                                    if (Isnotok != 0)
                                    {
                                        break;
```

```
                                    }
                                    for (var kk = 0; kk < ((i * h) + h); kk++)
                                    {
                                        if (Isnotok != 0) { break; }
                                        double percentresourcetype = 0;
                                        percentresourcetype = (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[kk]) *
(MyData.All.Activities[i].Q[ResourceType]);
                                        for (int Period = 0; Period < Horizen; Period++)
                                        {
                                            if (MyData.All.ForCrossoverChrom[parent].ChromosomePart2[kk] == Period)
                                            {
                                                MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] =
Math.Round(((MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period]) + percentresourcetype), 2);
                                            }
                                            if (MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] <=
(MyData.All.Resources[ResourceType].AvailableResourcesAtPeriodT[Period]))
                                            {
                                                ResourceControl = 1;
                                            }
                                            else if (MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] >
(MyData.All.Resources[ResourceType].AvailableResourcesAtPeriodT[Period]))
                                            {
                                                ResourceControl = 0; Isnotok = 1; break;
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                    foreach (var Preact in MyData.All.Activities[i].FeedingRelations.FindAll(l => l.TypeOfprecedence == "3"))
                    {
                        double Percent = 0;
                        int indexofPreAct = MyData.All.Activities[i].FeedingRelations.IndexOf(Preact);
                        int PredesenseOFAct = MyData.All.Activities[i].FeedingRelations[indexofPreAct].PreActivities;//
                        double percentprocessofPreAct = 0;
                        double LastGen2Actj = 0;
                        for (int indixofGen2ActjLastGen = (i * h); indixofGen2ActjLastGen < (i * h) + h; indixofGen2ActjLastGen++)
                        {
                            if (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[indixofGen2ActjLastGen] != 0)
```

```
                        {
                            LastGen2Actj = MyData.All.ForCrossoverChrom[parent].ChromosomePart2[indixofGen2ActjLastGen];
                        }
                    }
                    for (var kk = 0; kk < h; kk++)
                    {
                        int v = (PredesenseOFAct * h) + kk;
                        Percent = Percent + MyData.All.ForCrossoverChrom[parent].ChromosomePart1[v];
                        percentprocessofPreAct = MyData.All.ForCrossoverChrom[parent].ChromosomePart2[v];
                        if (Percent >= MyData.All.q3) { break; }
                    }
                    for (int ResorceNotOk = 0; ResorceNotOk < 50000; ResorceNotOk++)
                    {
                        int d = 0;
                        for (int indixofGen2ActjLastGen = (i * h); indixofGen2ActjLastGen < (i * h) + h; indixofGen2ActjLastGen++)
                        {
                            if (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[indixofGen2ActjLastGen] != 0)
                            {
                                d = d + 1;
                            }
                        }
                        if (ResourceControl == 1) { break; }
                        if (ResourceControl == 0 && Isnotok == 0)
                        {
                            if (percentprocessofPreAct < LastGen2Actj) { PreTypeOk = PreTypeOk + 0; }
                            else
                            {
                                PreTypeOk = PreTypeOk + 1;
                                LastGen2Actj = percentprocessofPreAct + 1;
                                for (int indixofGen2ActjLastGen = (i * h); indixofGen2ActjLastGen < (i * h) + h;
indixofGen2ActjLastGen++)
                                {
                                    if (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[indixofGen2ActjLastGen] == 0)
                                    {
                                        MyData.All.ForCrossoverChrom[parent].ChromosomePart2[indixofGen2ActjLastGen] = 0;
                                    }
                                    else
                                    {
                                        MyData.All.ForCrossoverChrom[parent].ChromosomePart2[indixofGen2ActjLastGen] = LastGen2Actj
+ 1 - d + 1;
```

```
                    d = d - 1;
                }
            }
        }
    }
    if (ResourceControl == 0 && Isnotok == 1)
    {
        Isnotok = 0;
        for (int Res = 0; Res < MyData.All.Resources.Count; Res++)
        {
            MyData.All.Resources[Res].UsingResourcesAtPeriodT.Clear();
            for (int Period = 0; Period < Horizen; Period++)
            {
                MyData.All.Resources[Res].UsingResourcesAtPeriodT.Add(0);
            }
        }
        for (var n = 0; n < h; n++)
        {
            for (int indixofGen2ActjLastGen = (i * h); indixofGen2ActjLastGen < (i * h) + h;
indixofGen2ActjLastGen++)
            {
                if (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[indixofGen2ActjLastGen] == 0)
                {
                    MyData.All.ForCrossoverChrom[parent].ChromosomePart2[indixofGen2ActjLastGen] = 0;
                }
                else
                {
                    MyData.All.ForCrossoverChrom[parent].ChromosomePart2[indixofGen2ActjLastGen] = LastGen2Actj
+ 1 - d + 1;

                    d = d - 1;
                }
            }
        }
    }
    for (int ResourceType = 0; ResourceType < MyData.All.Resources.Count; ResourceType++)
    {
        {
            if (Isnotok != 0) { break; }
            for (var kk = 0; kk < ((i * h) + h); kk++)
            {
```

```
                                    if (Isnotok != 0) { break; }
                                    double percentresourcetype = 0;
                                    percentresourcetype = (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[kk]) *
(MyData.All.Activities[i].Q[ResourceType]);
                                    for (int Period = 0; Period < Horizen; Period++)
                                    {
                                        if (MyData.All.ForCrossoverChrom[parent].ChromosomePart2[kk] == Period)
                                        {
                                            MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] =
Math.Round(((MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period]) + percentresourcetype), 2);
                                        }
                                        if (MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] <=
(MyData.All.Resources[ResourceType].AvailableResourcesAtPeriodT[Period]))
                                        {
                                            ResourceControl = 1;
                                        }
                                        else if (MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] >
(MyData.All.Resources[ResourceType].AvailableResourcesAtPeriodT[Period]))
                                        {
                                            ResourceControl = 0; Isnotok = 1; break;
                                        }
                                    }
                                }
                            }
                        }
                    }
                    foreach (var Preact in MyData.All.Activities[i].FeedingRelations.FindAll(l => l.TypeOfprecedence == "4"))
                    {
                        double Percent = 0;
                        int indexofPreAct = MyData.All.Activities[i].FeedingRelations.IndexOf(Preact);
                        int PredesenseOFAct = MyData.All.Activities[i].FeedingRelations[indexofPreAct].PreActivities;//
                        double percentprocessofPreAct = 0;
                        double LastGen2Actj = 0;
                        double percentj = 0;
                        for (int indixofGen2ActjLastGen = (i * h); indixofGen2ActjLastGen < (i * h) + h; indixofGen2ActjLastGen++)
                        {
                            if (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[indixofGen2ActjLastGen] != 0)
                            {
                                LastGen2Actj = MyData.All.ForCrossoverChrom[parent].ChromosomePart2[indixofGen2ActjLastGen];
```

```
                    percentj = percentj + MyData.All.ForCrossoverChrom[parent].ChromosomePart1[indixofGen2ActjLastGen];
                    if (percentj >= MyData.All.q4) { break; }
                }
            }
            for (var kk = 0; kk < h; kk++)
            {
                int v = (PredesenseOFAct * h) + kk;
                Percent = Percent + MyData.All.ForCrossoverChrom[parent].ChromosomePart1[v];
                percentprocessofPreAct = MyData.All.ForCrossoverChrom[parent].ChromosomePart2[v];
                if (Percent >= 1) { break; }
            }
            for (int ResorceNotOk = 0; ResorceNotOk < 50000; ResorceNotOk++)
            {
                int d = 0;
                for (int indixofGen2ActjLastGen = (i * h); indixofGen2ActjLastGen < (i * h) + h; indixofGen2ActjLastGen++)
                {
                    if (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[indixofGen2ActjLastGen] != 0)
                    {
                        d = d + 1;
                    }
                }
                if (ResourceControl == 1) { break; }
                if (ResourceControl == 0 && Isnotok == 0)
                {
                    if (percentprocessofPreAct < LastGen2Actj) { PreTypeOk = PreTypeOk + 0; }
                    else
                    {
                        PreTypeOk = PreTypeOk + 1;
                        LastGen2Actj = percentprocessofPreAct + 1;
                        for (int indixofGen2ActjLastGen = (i * h); indixofGen2ActjLastGen < (i * h) + h;
indixofGen2ActjLastGen++)
                        {
                            if (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[indixofGen2ActjLastGen] == 0)
                            {
                                MyData.All.ForCrossoverChrom[parent].ChromosomePart2[indixofGen2ActjLastGen] = 0;
                            }
                            else
                            {
                                MyData.All.ForCrossoverChrom[parent].ChromosomePart2[indixofGen2ActjLastGen] = LastGen2Actj
+ 1;
```

```
                    LastGen2Actj = LastGen2Actj + 1;
                    d = d - 1;
                }
            }
        }
    }
    if (ResourceControl == 0 && Isnotok == 1)
    {
        Isnotok = 0;
        for (int Res = 0; Res < MyData.All.Resources.Count; Res++)
        {
            MyData.All.Resources[Res].UsingResourcesAtPeriodT.Clear();
            for (int Period = 0; Period < Horizen; Period++)
            {
                MyData.All.Resources[Res].UsingResourcesAtPeriodT.Add(0);
            }
        }
        for (var n = 0; n < h; n++)
        {
            for (int indixofGen2ActjLastGen = (i * h); indixofGen2ActjLastGen < (i * h) + h;
indixofGen2ActjLastGen++)
            {
                if (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[indixofGen2ActjLastGen] == 0)
                {
                    MyData.All.ForCrossoverChrom[parent].ChromosomePart2[indixofGen2ActjLastGen] = 0;
                }
                else
                {
                    MyData.All.ForCrossoverChrom[parent].ChromosomePart2[indixofGen2ActjLastGen] = LastGen2Actj
+ 1 - d + 1;
                    d = d - 1;
                }
            }
        }
    }
    for (int ResourceType = 0; ResourceType < MyData.All.Resources.Count; ResourceType++)
    {
        {
            if (Isnotok != 0) { break; }
            for (var kk = 0; kk < ((i * h) + h); kk++)
```

```
                                    {
                                        if (Isnotok != 0) { break; }
                                        double percentresourcetype = 0;
                                        percentresourcetype = (MyData.All.ForCrossoverChrom[parent].ChromosomePart1[kk]) *
(MyData.All.Activities[i].Q[ResourceType]);
                                        for (int Period = 0; Period < Horizen; Period++)
                                        {
                                            if (MyData.All.ForCrossoverChrom[parent].ChromosomePart2[kk] == Period)
                                            {
                                                MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] =
Math.Round(((MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period]) + percentresourcetype), 2);
                                            }
                                            if (MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] <=
(MyData.All.Resources[ResourceType].AvailableResourcesAtPeriodT[Period]))
                                            {
                                                ResourceControl = 1;
                                            }
                                            else if (MyData.All.Resources[ResourceType].UsingResourcesAtPeriodT[Period] >
(MyData.All.Resources[ResourceType].AvailableResourcesAtPeriodT[Period]))
                                            {
                                                ResourceControl = 0;
                                                Isnotok = 1;
                                                break;
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }  // Feasibility Control
                for (int l = 0; l < (h * j); l++) // normal crossoverden sonra şartlar OK
                {
                    MyData.All.ChildChromosome[0].ChildChromosomePart1.Add(MyData.All.ForCrossoverChrom[0].ChromosomePart1[l]);
                    MyData.All.ChildChromosome[1].ChildChromosomePart1.Add(MyData.All.ForCrossoverChrom[1].ChromosomePart1[l]);
                    MyData.All.ChildChromosome[0].ChildChromosomePart2.Add(MyData.All.ForCrossoverChrom[0].ChromosomePart2[l]);
                    MyData.All.ChildChromosome[1].ChildChromosomePart2.Add(MyData.All.ForCrossoverChrom[1].ChromosomePart2[l]);
                    MyData.All.ChildChromosome[0].MaxCmax = MyData.All.ChildChromosome[0].ChildChromosomePart2.Max();
                    MyData.All.ChildChromosome[1].MaxCmax = MyData.All.ChildChromosome[1].ChildChromosomePart2.Max();
                }
```

```
            }
        }
    }
}
```

EK. 2.4 C# Mutation

```
    class Mutation……
    {
        public Mutation()
        {
            MyData.Init();
        }
         public void MutationAndControl()
        {
            int h = MyData.All.NoOfGenForActivities;int hh = h;
            int j = MyData.All.Activities.Count();
            int Horizen = MyData.All.HorizenPeriodNo;
            for (int NoMutation = 0; NoMutation < 1; NoMutation++)
            {
                MyData.All.ForMutation[NoMutation].ChromosomePart1.Clear();
                MyData.All.ForMutation[NoMutation].ChromosomePart2.Clear();
                Random rnd = new Random();
                int Jmutationj = rnd.Next(0, (j-1)); // nerden mutation yapsın in which j
                MyData.All.Activities[Jmutationj].GenDataChrom1.Clear();
                MyData.All.ForMutation[NoMutation].Mutationpointj = Jmutationj;
                int FeasibleChromForMutation = rnd.Next(1, (int)MyData.All.Population);
                MyData.All.ForMutation[NoMutation].FeasiblechromNoForMutation = FeasibleChromForMutation; // in which chromosome

                for (int m = 0; m < (h * (Jmutationj)); m++)
                {

MyData.All.ForMutation[NoMutation].ChromosomePart1.Add(MyData.All.FeasibleChromosome[FeasibleChromForMutation].ChromosomePart1[m]);
```

```
MyData.All.ForMutation[NoMutation].ChromosomePart2.Add(MyData.All.FeasibleChromosome[FeasibleChromForMutation].ChromosomePart2[m]);
                }   // before mutation point
                double X = 0; double Y = 1; int PP = 0;
                for (int n = (h * (Jmutationj)); n < (h * (Jmutationj + 1)); n++)
                {
                    double value = RandomNumberBetween(MyData.All.Activities[Jmutationj].Bmin, MyData.All.Activities[Jmutationj].Bmax);
                    double RandPercentage = Math.Round(value, 2);
                    if ((Y - RandPercentage) >= MyData.All.Activities[Jmutationj].Bmin && (X + RandPercentage) <= 1)
                    {
                        MyData.All.Activities[Jmutationj].GenDataChrom1.Add(Math.Round(RandPercentage, 2));
                        MyData.All.ForMutation[NoMutation].ChromosomePart1.Add(Math.Round(RandPercentage, 2));
                        X = X + RandPercentage; Y = Y - RandPercentage;
                        PP++;
                    }
                    else if ((Y - RandPercentage) <= MyData.All.Activities[Jmutationj].Bmin && X < 1) /*&& (X + RandPercentage) <= 1)*/
                    {
                        if (X > 0)
                        {
                            double XX = 1 - X;
                            MyData.All.Activities[Jmutationj].GenDataChrom1.Add(Math.Round(XX, 2));
                            MyData.All.ForMutation[NoMutation].ChromosomePart1.Add(Math.Round(XX, 2));
                            PP++;
                            X = X + XX;
                            Y = Y - XX;
                        }
                        else if (X == 0 && Y == 1)
                        {
                            MyData.All.Activities[Jmutationj].GenDataChrom1.Add(MyData.All.Activities[Jmutationj].Bmin);
                            MyData.All.ForMutation[NoMutation].ChromosomePart1.Add(MyData.All.Activities[Jmutationj].Bmin);
                            PP++;
                            X = X + MyData.All.Activities[Jmutationj].Bmin;
                            Y = Y - MyData.All.Activities[Jmutationj].Bmin;
                        }
                    }
                    else if ((Y - RandPercentage) < 0 && X < 1)
                    {
                        MyData.All.Activities[Jmutationj].GenDataChrom1.Add(MyData.All.Activities[Jmutationj].Bmin);
                        MyData.All.ForMutation[NoMutation].ChromosomePart1.Add(MyData.All.Activities[Jmutationj].Bmin);
                        PP++;
```

```
                        X = X + MyData.All.Activities[Jmutationj].Bmin;
                        Y = Y - MyData.All.Activities[Jmutationj].Bmin;
                    }
                    else if (X == 1)
                    {
                        MyData.All.Activities[Jmutationj].GenDataChrom1.Add((0));
                        MyData.All.ForMutation[NoMutation].ChromosomePart1.Add(0);
                    }
                }  // part 1 mutation point
                for (int O = h * (Jmutationj); O < (h * j); O++)
                {

MyData.All.ForMutation[NoMutation].ChromosomePart1.Add(MyData.All.FeasibleChromosome[FeasibleChromForMutation].ChromosomePart1[O]);
                } // part one after mutation part
                int finishtime3 = 0;

                for (int nn = (h * (Jmutationj )); nn < (h * (Jmutationj + 1)); nn++)
                {
                    int starttime = MyData.All.Activities[Jmutationj].ReleaseT; int finishtime =
MyData.All.Activities[Jmutationj].DuedateT;
                    foreach (var Preact in MyData.All.Activities[Jmutationj].FeedingRelations.FindAll(l => l.TypeOfprecedence == "1"))
                    {
                        double Percent = 0;
                        int indexofPreAct = MyData.All.Activities[Jmutationj].FeedingRelations.IndexOf(Preact);
                        int PredesenseOFAct = MyData.All.Activities[Jmutationj].FeedingRelations[indexofPreAct].PreActivities;

                            double StartPriodOfPreAct = 0;
                            for (var kk = 0; kk < h; kk++)
                            {
                            Percent = Percent + MyData.All.ForMutation[NoMutation].ChromosomePart1[PredesenseOFAct * h + kk];

                            StartPriodOfPreAct = MyData.All.ForMutation[NoMutation].ChromosomePart2[PredesenseOFAct * h + kk];
                            if (Percent >= MyData.All.q1) { break; }
                            }
                            if (starttime <= StartPriodOfPreAct) { starttime = (int)StartPriodOfPreAct + 1; }

                    }
                    foreach (var Preact in MyData.All.Activities[Jmutationj].FeedingRelations.FindAll(l => l.TypeOfprecedence == "2"))
                    {
                        double Percent = 0;
```

```
    int indexofPreAct = MyData.All.Activities[Jmutationj].FeedingRelations.IndexOf(Preact);
    int PredesenseOFAct = MyData.All.Activities[Jmutationj].FeedingRelations[indexofPreAct].PreActivities;

    double StartPriodOfPreAct = 0;
    for (var kk = 0; kk < h; kk++)
    {
        Percent = Percent + MyData.All.ForMutation[NoMutation].ChromosomePart1[PredesenseOFAct * h + kk];

        StartPriodOfPreAct = MyData.All.ForMutation[NoMutation].ChromosomePart2[PredesenseOFAct * h + kk];
        if (Percent >= MyData.All.q2) { break; }
    }
    if (starttime <= StartPriodOfPreAct) { starttime = (int)StartPriodOfPreAct + 1; }

}
foreach (var Preact in MyData.All.Activities[Jmutationj].FeedingRelations.FindAll(l => l.TypeOfprecedence == "3"))
{
    double Percent = 0;
    int indexofPreAct = MyData.All.Activities[Jmutationj].FeedingRelations.IndexOf(Preact);
    int PredesenseOFAct = MyData.All.Activities[Jmutationj].FeedingRelations[indexofPreAct].PreActivities;
    if (MyData.All.Activities[PredesenseOFAct].GenDataChrom2.Count != 0)
    {
        double percentprocessofPreAct = 0;
        for (var kk = 0; kk < h; kk++)
        {
            Percent = Percent + MyData.All.ForMutation[NoMutation].ChromosomePart1[PredesenseOFAct * h + kk];
            percentprocessofPreAct = MyData.All.ForMutation[NoMutation].ChromosomePart2[PredesenseOFAct * h + kk];
            if (Percent >= MyData.All.q3) { break; }
        }
        finishtime3 = (int)percentprocessofPreAct + 1;
    }
}
foreach (var Preact in MyData.All.Activities[Jmutationj].FeedingRelations.FindAll(l => l.TypeOfprecedence == "4"))
{
    double Percent = 0;
    int indexofPreAct = MyData.All.Activities[Jmutationj].FeedingRelations.IndexOf(Preact);
    int PredesenseOFAct = MyData.All.Activities[Jmutationj].FeedingRelations[indexofPreAct].PreActivities;
    if (MyData.All.Activities[PredesenseOFAct].GenDataChrom2.Count != 0)
    {
        double percentprocessofPreAct = 0;
        for (var kk = 0; kk < h; kk++)
```

```
        {
            Percent = Percent + MyData.All.ForMutation[NoMutation].ChromosomePart1[PredesenseOFAct * h + kk];
            percentprocessofPreAct =MyData.All.ForMutation[NoMutation].ChromosomePart2[PredesenseOFAct * h + kk];
            if (Percent >= 1) { break; }
        }
        if (starttime <= percentprocessofPreAct)
        {
            starttime = (int)percentprocessofPreAct + 1;
        }
    }
}

int finish2 =(int) MyData.All.FeasibleChromosome[FeasibleChromForMutation].ChromosomePart2[(Jmutationj * h) + 1];
if(finish2 < finishtime3) { finishtime3 = finish2;  }
if (finishtime3 <finishtime ) { finishtime3 = finishtime; }
int U = 0; int newstarttime = 0;
for (var k = 0; k < h; k++)
{
    if (MyData.All.ForMutation[NoMutation].ChromosomePart1[Jmutationj * h + k] != 0) { U++; }
}


for (var k = 0; k < h; k++)
{
    if (U == 1 && k == 0)
    {
        int prossesingperiod = rnd.Next(starttime, finishtime3);
        MyData.All.ForMutation[NoMutation].ChromosomePart2.Add(prossesingperiod);
        starttime = prossesingperiod + 1; U--; newstarttime = prossesingperiod;
    }
    if (U > 1|| (U == 1 && k != 0))
    {
        int prossesingperiod = rnd.Next(starttime, (finishtime3 - (U - 1)));
        MyData.All.ForMutation[NoMutation].ChromosomePart2.Add(prossesingperiod);
        starttime = prossesingperiod + 1; U--; newstarttime = prossesingperiod;
    }

    else { MyData.All.ForMutation[NoMutation].ChromosomePart2.Add(0); }
}
```

```
MyData.All.ForMutation[NoMutation].ChromosomePart2.Add(MyData.All.FeasibleChromosome[FeasibleChromForMutation].ChromosomePart2[nn]);
                }
                for (int OO = h * (Jmutationj + 1); OO < (h * j); OO++)
                {

MyData.All.ForMutation[NoMutation].ChromosomePart2.Add(MyData.All.FeasibleChromosome[FeasibleChromForMutation].ChromosomePart2[OO]);

                }
                MyData.All.ForMutation[NoMutation].MaxCmax = MyData.All.ForMutation[NoMutation].ChromosomePart2.Max() + 1;
            }   // Mutation
        }
    }
}
```

EK. 3 C# Gurobi Matematiksel Model (amaç fonksiyon ve kısıtların kodu)

```
                var expr = new GRBQuadExpr(); // amaç fonksiyon
                expr.AddTerm(100, Cmax);

                expr.Add(SumusingResourceAllPeriod * SumusingResourceAllPeriod);
                for (int r = 0; r < MyData.All.Resources.Count(); r++)
                {
                    for (int t = 0; t < Time; t++)
                    {
                        expr.Add(W[r, t]* W[r, t]);
                    }
                }

                model.SetObjective(expr, GRB.MINIMIZE);
// kısıtlar



                GRBQuadExpr ResourceSum = 0;
                for (int K = 0; K < MyData.All.Resources.Count; K++)           // NO. 1
                {
                    GRBQuadExpr Eq18 = 0;
                    for (int t = 0; t < Time; t++)
                    {
                        GRBLinExpr Eq17 = 0;
                        for (int i = 0; i < NumOfActivity; i++)
                        {
                            if (MyData.All.Activities[i].IDofQ.Contains(K))//
                            {
                                int indexK = MyData.All.Activities[i].IDofQ.IndexOf(K);
                                Eq17.AddTerm(MyData.All.Activities[i].Q[indexK], X[i, t]);
                            }
                        }
                        model.AddConstr(Eq17 - W[K, t] == 0, "EquationNo17");
                    }
                }
                // Constrait ------------------------------------------------------------------------------------------------
                for (int i = 0; i < NumOfActivity; i++)                                        // NO. 5
```

```
{
    GRBLinExpr EqNo5 = 0; //xtot double
    for (int t = MyData.All.Activities[i].ReleaseT; t <= MyData.All.Activities[i].DuedateT; t++)
    {
        EqNo5.AddTerm(1, X[i, t]);
    }
    model.AddConstr(EqNo5 == 1, (i + 1).ToString() + "EquationNo5");
}
// ------------------------------------------------------------------------------------------------
for (int i = 0; i < NumOfActivity; i++)                                      // NO . 7
{
    for (int t = 0; t < Time; t++)
    {
        GRBLinExpr EqNo6 = 0;
        EqNo6.AddTerm(1, X[i, t]);
        EqNo6.AddTerm(-1 * (MyData.All.Activities[i].Bmax), Eta[i, t]);
        model.AddConstr(EqNo6 <= 0, (i + 1).ToString() + (t + 1).ToString() + "EquationNo6");
    }
}
//  ------------------------------------------------------------------------------------------------
for (int i = 0; i < NumOfActivity; i++)                                     // NO . 8
{
    for (int t = 0; t < Time; t++)
    {
        GRBLinExpr EqNo7 = 0;
        EqNo7.AddTerm(1, X[i, t]);
        EqNo7.AddTerm(-1 * (MyData.All.Activities[i].Bmin), Eta[i, t]);
        model.AddConstr(EqNo7 >= 0, (i + 1).ToString() + (t + 1).ToString() + "EquationNo7");
    }
}
//  ------------------------------------------------------------------------------------------------
for (int K = 0; K < MyData.All.Resources.Count; K++)                          // NO. 13
{
    for (int t = 0; t < Time; t++)
    {
        GRBLinExpr Eq12 = 0;

        for (int i = 0; i < NumOfActivity; i++)
        {
            if (MyData.All.Activities[i].IDofQ.Contains(K))// sadece gerekli olanlar yaratılacak
```

```
                        {
                            int indexK = MyData.All.Activities[i].IDofQ.IndexOf(K);
                            Eq12.AddTerm(MyData.All.Activities[i].Q[indexK], X[i, t]);
                        }
                    }
                    model.AddConstr(Eq12 <= MyData.All.Resources[K].ResourcesR[t], (K + 1).ToString() + (t + 1).ToString() +
"EquationNo12");
                }
            }
            //-------------------------------------------------------------------------------------------
            for (int i = 0; i < NumOfActivity; i++)                                    // NO 2
            {
                GRBLinExpr Eq1 = 0;
                for (int t = MyData.All.Activities[i].ReleaseT; t <= MyData.All.Activities[i].DuedateT; t++)
                {
                    Eq1.AddTerm(t, FinishCO[i, t]);
                }
                model.AddConstr(Eq1 <= Cmax, ToString() + "EquationNo1");
            }
            //-------------------------------------------------------------------------------------------
            for (int i = 0; i < NumOfActivity; i++)                              // NO. 3
            {
                GRBLinExpr Eq2 = 0; //xtot double
                for (int t = MyData.All.Activities[i].ReleaseT; t <= MyData.All.Activities[i].DuedateT; t++) //
                {
                    Eq2.AddTerm(1, StartCO[i, t]);
                }
                model.AddConstr(Eq2 == 1, (i + 1).ToString() + "EquationNo2");
            }
            //-------------------------------------------------------------------------------------------
            for (int i = 0; i < NumOfActivity; i++)                                // NO. 4
            {
                GRBLinExpr Eq3 = 0; //xtot double
                for (int t = MyData.All.Activities[i].ReleaseT; t <= MyData.All.Activities[i].DuedateT; t++) //
                {
                    Eq3.AddTerm(1, FinishCO[i, t]);
                }
                model.AddConstr(Eq3 == 1, (i + 1).ToString() + "EquationNo3");
            }
            //-------------------------------------------------------------------------------------------
```

```
for (int i = 0; i < NumOfActivity; i++)                                  // NO. 5
{
    GRBLinExpr Eq4 = 0;
    for (int t = MyData.All.Activities[i].ReleaseT; t <= MyData.All.Activities[i].DuedateT; t++)
    {
        Eq4.AddTerm(t, StartCO[i, t]);
    }
    model.AddConstr(Eq4 >= MyData.All.Activities[i].ReleaseT, (i + 1).ToString() + "EquationNo4");
}
for (int i = 0; i < NumOfActivity; i++)
{
    GRBLinExpr Eq88 = 0;
    for (int t = 0; t < MyData.All.Activities[i].ReleaseT; t++)
    {
        Eq88.AddTerm(1, X[i, t]);
    }
    for (int t = MyData.All.Activities[i].DuedateT; t < Time; t++)
    {
        Eq88.AddTerm(1, X[i, t]);
    }
    model.AddConstr(Eq88 == 0, (i + 1).ToString() + "EquationNo88");
}
//  ------------------------------------------------------------------------------------------
for (int i = 0; i < NumOfActivity; i++)                                  // NO.9
       {
    for (int t = 0; t < Time; t++)
    {
        GRBLinExpr Eq8 = 0;
        Eq8.AddTerm(1, StartCO[i, t]);
        Eq8.AddTerm(-1, Eta[i, t]);
        model.AddConstr(Eq8 <= 0, (i + 1).ToString() + (t + 1).ToString() + "EquationNo8");
    }
}
//------------------------------------------------------------------------------------------
for (int i = 0; i < NumOfActivity; i++)                                  // NO.10
{
    for (int t = 0; t < Time; t++)
    {
        model.AddConstr(FinishCO[i, t] <= Eta[i, t], (i + 1).ToString() + (t + 1).ToString() + "EquationNo9");
    }
```

```
}
//-------------------------------------------------------------------------------------------
for (int i = 0; i < NumOfActivity; i++)                                    // NO. 11
{
    for (int t = 0; t < Time; t++)
    {
        GRBLinExpr Eq10 = 0;
        Eq10.AddTerm(1, FinishCO[i, t]);
        for (int h = MyData.All.Activities[i].ReleaseT; h <= t; h++)
        {
            Eq10.AddTerm(-1, X[i, h]);
        }
        model.AddConstr(Eq10 <= 0, (i + 1).ToString() + (t + 1).ToString() + "EquationNo10");
    }
}
//-------------------------------------------------------------------------------------------
for (int i = 0; i < NumOfActivity; i++)                                    // NO. 12
{
    for (int t = 0; t < Time; t++)
    {
        GRBLinExpr Eq11 = 0;
        Eq11.AddTerm(1, X[i, t]);
        for (int h = MyData.All.Activities[i].ReleaseT; h <= t; h++)
        {
            Eq11.AddTerm(-1, StartCO[i, h]);
        }
        model.AddConstr(Eq11 <= 0, (i + 1).ToString() + (t + 1).ToString() + "EquationNo11");
    }
}
//  -------------------------------------------------------------------------------------------
foreach (var Act in MyData.All.Activities)
{
    foreach (var Actwithpre1 in Act.FeedingRelations.FindAll(l => l.TypeOfprecedence == "1"))
    {
        for (int t = 0; t < Time; t++)
        {
            GRBLinExpr Eq13 = 0;
            int indexofAct1 = MyData.All.Activities.IndexOf(Act);
            Eq13.AddTerm(1, StartCO[indexofAct1, t]);
```

```
                        for (int h = 0; h <= t-1 ; h++)
                        {
                            Eq13.AddTerm(-1, X[Actwithpre1.PreActivities, h]);
                        }
                        model.AddConstr(Eq13 <= -Actwithpre1.Yüzde + 1, (indexofAct1 + 1).ToString() + (t + 1).ToString() +
"EquationNo13");
                    }
                }
                // ------------------------------------------------------------------------------------------------------------
----------------
                foreach (var Actwithpre3 in Act.FeedingRelations.FindAll(l => l.TypeOfprecedence == "3"))
                {
                    for (int t = 0; t < Time; t++)
                    {
                        GRBLinExpr Eq14 = 0;
                        int indexofAct3 = MyData.All.Activities.IndexOf(Act);
                        for (int h = 0; h <= t ; h++)
                        {
                            Eq14.AddTerm(-1, FinishCO[indexofAct3, h]);
                        }
                        for (int h = 0; h <= t - 1; h++)
                        {
                            Eq14.AddTerm(1, X[Actwithpre3.PreActivities, h]);
                        }
                        model.AddConstr(Eq14 >= Actwithpre3.Yüzde - 1, (indexofAct3 + 1).ToString() + (t + 1).ToString() +
"EquationNo14");
                    }
                }
                // ------------------------------------------------------------------------------------------------------------
----------------
                foreach (var Actwithpre2 in Act.FeedingRelations.FindAll(l => l.TypeOfprecedence == "2"))
                {
                    for (int t = 0; t < Time; t++)
                    {
                        GRBLinExpr Eq15 = 0;
                        int indexofAct2 = MyData.All.Activities.IndexOf(Act);
                        for (int h = 0; h <= t - 1; h++)
                        {
                            Eq15.AddTerm((Actwithpre2.Yüzde - 1), StartCO[Actwithpre2.PreActivities, h]);
                        }
```

```
                            for (int h = 0; h <= t ; h++)
                            {
                                Eq15.AddTerm(1, X[indexofAct2, h]);
                            }
                            model.AddConstr(Eq15 <= Actwithpre2.Yüzde, (indexofAct2 + 1).ToString() + (t + 1).ToString() +
"EquationNo15");
                        }
                    }
                    //    //-------------------------------------------------------------------------------------------------------------------
------------------------
                    foreach (var Actwithpre4 in Act.FeedingRelations.FindAll(l => l.TypeOfprecedence == "4"))
                    {
                        for (int t = 0; t < Time; t++)
                        {
                            GRBLinExpr Eq16 = 0;
                            int indexofAct4 = MyData.All.Activities.IndexOf(Act);
                            for (int h = 0; h <= t - 1; h++)
                            {
                                Eq16.AddTerm((Actwithpre4.Yüzde - 1), FinishCO[Actwithpre4.PreActivities, h]);
                            }
                            for (int h = 0; h <= t ; h++)
                            {
                                Eq16.AddTerm(1, X[indexofAct4, h]);
                            }
                            model.AddConstr(Eq16 <= Actwithpre4.Yüzde, (indexofAct4 + 1).ToString() + (t + 1).ToString() +
"EquationNo16");
                        }
                    }
                }
```

ÖZGEÇMİŞ

Adı Soyadı : Mastaneh JOUSHANI

Doğum Yeri ve Tarihi : 12.09.1978 / TEBRİZ, İRAN

Yabancı Dil : İngilizce, Fransızca, Türkçe

Eğitim Durumu

Lise : (1994-1997) ERAM Tebriz

Lisans : (1997-2001) Tebriz Sahand teknoloji üniversitesi, Elektrik Mühendisliği Fakültesi, Kontrol Mühendisliği

Yüksek Lisans : (2009-2011) Tahran Azad-E İslami Üniversitesi, Elektrik Mühendisliği Fakültesi, Kontrol Mühendisliği

(2018-2022) Uludağ Üniversitesi Endüstri Mühendisliği Dalı

Çalıştığı Kurum/Kurumlar : (2002-2003) İran Niru (Yüksek gerilim elektrik panosu imalat fabrikası)

(2003-2008) SAİPA (Otomotiv şirketi) bakım ve servis Departmanı

(2008-2010) SAİPA. Otomasyon departmanı

(2010-2014) SAİPA. Enerji yönetimi departmanı

(2014- 2018) Farayand sazan. Elektrik ve Otomasyon şantiye

(2021-Halen) Konverta Geri Dönüşüm A.Ş.

İletişim (e-posta) : mjooshani@gmail.com