ÇİZELGELEME ALGORİTMALARININ ÇALIŞMA SÜRELERİNİN İLERİ VERİ YAPILARI İLE İYİLEŞTİRİLMESİ

Beray BAYAZIT

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÇİZELGELEME ALGORİTMALARININ ÇALIŞMA SÜRELERİNİN İLERİ VERİ YAPILARI İLE İYİLEŞTİRİLMESİ

Beray BAYAZIT
00000-0001-5911-7818

Prof. Dr. Seda ÖZMUTLU
(Danışman)

YÜKSEK LİSANS TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2022

TEZ ONAYI

Beray BAYAZIT tarafından hazırlanan "ÇİZELGELEME ALGORİTMALARININ ÇALIŞMA SÜRELERİNİN İLERİ VERİ YAPILARI İLE İYİLEŞTİRİLMESİ" 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. Seda ÖZMUTLU

Başkan : Prof. Dr. Seda ÖZMUTLU İmza
0000-0002-2744-2744
Bursa Uludağ Üniversitesi,
Mühendislik Fakültesi,
Endüstri Mühendisliği Anabilim Dalı

Üye : Prof. Dr. Fatih ÇAVDUR İmza
0000-0001-8054-5606
Bursa Uludağ Üniversitesi,
Mühendislik Fakültesi,
Endüstri Mühendisliği Anabilim Dalı

Üye : Dr. Öğr. Üyesi Yunus DEMİR İmza
0000-0003-3868-1860
Bursa Teknik Üniversitesi,
Mühendislik ve Doğa Bilimleri Fakültesi,
Endüstri Mühendisliği Anabilim Dalı

Yukarıdaki sonucu onaylarım

Prof. Dr. Hüseyin Aksel EREN
Enstitü Müdürü
21/02/2022

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.

21/02/2022

Beray BAYAZIT

TEZ YAYINLANMA
FİKRİ MÜLKİYET HAKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezin/raporun tamamını veya herhangi bir kısmını, basılı (kâğıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma izni Bursa Uludağ Üniversitesi'ne aittir. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet hakları ile tezin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları tarafımıza ait olacaktır. Tezde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alınarak kullandığını ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederiz.

Yükseköğretim Kurulu tarafından yayınlanan "**Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge**" kapsamında, yönerge tarafından belirtilen kısıtlamalar olmadığı takdirde tezin YÖK Ulusal Tez Merkezi / B.U.Ü. Kütüphanesi Açık Erişim Sistemi ve üye olunan diğer veri tabanlarının (Proquest veri tabanı gibi) erişimine açılması uygundur.

Prof. Dr. Seda ÖZMUTLU
21/02/2022

İmza
Bu bölüme kişinin kendi el yazısı ile okudum anladım yazmalı ve imzalanmalıdır.

Beray BAYAZIT
21/02/2022

İmza
Bu bölüme kişinin kendi el yazısı ile okudum anladım yazmalı ve imzalanmalıdır.

ÖZET

Yüksek Lisans Tezi

ÇİZELGELEME ALGORİTMALARININ ÇALIŞMA SÜRELERİNİN İLERİ VERİ YAPILARI İLE İYİLEŞTİRİLMESİ

Beray BAYAZIT

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Endüstri Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Seda ÖZMUTLU

Günümüzde gelişen teknoloji ve artan ürün çeşitliliği işletmelerin rekabet ortamında hayatta kalabilmeleri için süreçlerinde iyileşme yapmalarını gerektirmektedir. Bu yüzden, işletmelerin sınırlı kaynaklarını daha etkin kullanabilmeleri için üzerinde çokça durduğu önemli bir konu da çizelgelemedir.

Üretim sürecinde zaman tasarrufu elde etmek ya da bir sürecin daha kısa sürede tamamlanmasını sağlayarak sahip olunan sınırlı kaynakların verimini arttırmak adına iyileştirme sürecine yönelik çizelgeleme algoritmaları günümüzün en popüler çözüm yöntemlerindendir. Ancak çizelgeleme algoritmaları gerçek sistemler üzerinde uygulanmak istediğinde, çözüm süreleri uzamakta ve bugünün bilgisayar hızlarında bile gerçek hayatta uygulanabilir sürelerde çözüm sağlanamamaktadır. Çizelgeleme algoritmalarının gerçek hayatta uygulalanabilir olması adına, bu çalışmanın amacı; teknolojik gelişmelere rağmen çözülemeyen çok büyük boyutlu çizelgeleme algoritmalarının ileri veri yapılarıyla performansını iyileştirmektir.

Bu nedenle çalışmada liste ve bağlı liste yapılarına sahip iki adet algoritma önerilmiştir. Veri yapısının algoritma çözüm hızına çok önemli etkisi mevcuttur ve özellikle çok büyük boyutlu problemler üzerinde bu etki daha iyi görüleceği için sentetik veriler oluşturulmuştur. Esnek atölye tipi çizelgeleme problemi üzerinden farklı parametreleri de dikkate almaya çalışarak üç senaryo tasarlanmıştır. Bu senaryolara göre liste ve bağlı liste veri yapılarına sahip algoritmaların çalışma süreleri açışından performansları değerlendirilmiştir. İleri veri yapısı içeren algoritmanın, temel veri yapısına sahip algoritmaya göre bütün senaryolar için istatistiksel olarak anlamlı bir farkla daha hızlı çalıştığı gözlenmiştir. Diğer çizelgeleme algoritmalarına, özellikle popülasyon tabanlı algoritmalara, uygulandığında CPU işlem süresinin düşeceği ön görülmektedir.

Anahtar Kelimeler: Esnek atölye tipi çizelgeleme, sezgiseller, veri yapısı, listeler, çift bağlı doğrusal listeler

2022, vii + 47 sayfa.

ABSTRACT

MSc Thesis

IMPROVEMENT OF EXECUTION TIMES OF SCHEDULING ALGORITHMS WITH ADVANCED DATA STRUCTURES

Beray BAYAZIT

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

Supervisor: Prof. Dr. Seda ÖZMUTLU

Nowadays, developing technology and increasing product diversity require businesses to improve their processes in order to survive in competitive environments. Therefore, an businesses focus on scheduling to effectively use their limited resources.

Scheduling algorithms are one of the most popular solution methods today in order to save time in the production processes or to increase the efficiency of limited resources by ensuring that a process is completed in a shorter time. However, when scheduling algorithms are applied, the solution durations are extensively large and applicable results cannot be attained in real time. In order to make scheduling algorithms applicable in real-time, the aim of study is to improve the performance of very large scheduling algorithms that cannot be solved despite technological developments with advanced data structures.

Two algorithms, one with listdaha strauctur, and the second with linked list data structure, were proposed. The data structures have a very important effect on the algorithm solution speed, and therefore synthetic data were created to show their effect especially on very large sized problems. The performances of the algorithms with two different data structures were evaluated in terms of running times. Three scenarios were designed with different parameters, and algorithms were written according to these scenarios with list and linked list data structures. It has been observed that the algorithm with the advanced data structure Works statistically significantly faster for all scenarios than the algorithm with basic data structure. It is predicted that CPU processing time will decrease when applied to other scheduling algorithms, especially population-based algorithms.

Key words: Flexible job-shop scheduling, heuristics, data structure, lists, two-linked linear lists

2022, vii + 47 pages.

TEŞEKKÜR

Tez çalışmamın oluşturulması ve yürütülmesi aşamalarında her konuda yol gösterici yaklaşımıyla her zaman bana destek olan, tecrübelerini ve bilgilerini paylaşarak ufkumu genişleten çok değerli hocam rahmetli Prof. Dr. H. Cenk ÖZMUTLU'ya minnettarlığımı belirtmek isterim. Çalışmalarım boyunca sabrı ve hoşgörüsüyle bana her türlü olanağı sağlayan, bilgileriyle tez çalışmalarıma katkıda bulunan çok değerli hocam ve tez danışmanım Prof. Dr. Seda ÖZMUTLU'ya saygı ve teşekkürü bir borç bilirim. Akademik ve kültürel gelişimimde emeği olan tüm öğretmenlerime teşekkür ederim.

Destekleri ile hep yanımda olan, beni yetiştiren ve bugüne kadar emeklerini esirgemeyen ve tez çalışmamın her döneminde ihtiyacım olduğu anda yardımıma koşan annem Fatma BAYAZIT'a, babam Osman Seza BAYAZIT'a, ağabeyim Süleyman Ceyhun BAYAZIT'a ve manevi desteğiyle bu dönemi güzel şekilde geçirmeme katkıda bulunan Bengisu TUNA'ya ve burada adını sayamadığım ancak üzerimde emeği bulunan herkese gönülden teşekkür ederim.

Beray BAYAZIT
21/02/2022

İÇİNDEKİLER

Sayfa

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler	**Açıklama**
α	Makine Ortamı
β	İşlem Kısıtları
γ	Amaç Fonksiyonu

Kısaltmalar	**Açıklama**
GA	Genetik Algoritma
TA	Tabu Arama
PSO	Parçacık Sürü Optimizasyonu
MA	Memetik Algoritma
VNS	Değişken Komşuluk Arama
ILP	Yinelenen Yerel Arama
NSGA-II	Bastırılmamış Sınıflandırmalı Genetik Algoritma-II
GGA	Gruplayıcı Genetik Algoritma
CRO	Kimyasal Reaksiyon Optimizasyonu
HEA	Hibrit Evrimsel Algoritma
NP	Polinomsal Olmayan
P	Polinomsal

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

1. GİRİŞ

1.1. Çalışmanın Amacı

Son yıllarda imalat sanayi benzeri görülmemiş derecede bir değişim yaşamıştır; küresel rekabet, kısalan üretim yaşam döngüsü, yönetimdeki değişiklikler, artan kalite gereksinimleri, artan müşteri beklentileri, karmaşık teknolojideki daha hızlı ilerlemeler ve malzeme ve süreçlerde hızla genişleyen seçenekler imalat ortamını zorlaştıran koşullardır. Günümüz işletmeleri için yalnızca bu değişen iş ortamına nasıl uyum sağlayacakları değil, aynı zamanda bunu yapmayı seçtikleri yollardan nasıl rekabet avantajı elde edecekleri de büyük bir zorluktur. Bu tür avantajları elde etmek için işletmeler üretim sistemlerinin işleyişini optimize etmeye başlamışlardır. Üretim planlaması, çizelgeleme ve kontrol mekanizmalarının kullanılması da bu ihtiyaçlara yanıt vermek için zaruri hale gelmiştir.

Esnek atölye çizelgeleme problemi, endüstriyel alanlarda geniş uygulamaları olan NP-Zor sınıfında yer alan kombinatoryal optimizasyon problemidir ve atölye çizelgeleme probleminin genişletilmiş halidir (Türkyılmaz ve diğerleri, 2020). Klasik atölye tipi çizelgeleme problemi, bir makine seti üzerinde işlerin sıralanması ile ilgilidir. Esnek atölye tipi çizelgeleme probleminde ise bir işin bir operasyonu birden fazla makinede işlenebilir, yani her operasyon için alternatif makineler söz konusudur. Bu nedenle esnek atölye tipi çizelgeleme problemi, operasyonların alternatif makinelerden herhangi birisine atanması ve bu makine grubunda işlerin sıralanması alt problemlerinden oluşmaktadır (Kaya ve Fığlalı, 2018).

Araştırmacılar atama ve sıralama alt problemleri için çeşitli çözüm yöntemleri geliştirmiştir. Matematiksel yöntemlerin istenilen sürelerde çözümde yetersiz kalması nedeniyle literatürde %90 sezgisel ve meta sezgisel yöntemlerle çalışılmıştır (Sirkeci, 2015). Literatüre bakıldığında çoğunlukla iki veya daha fazla yaklaşık yöntemlerin birleştirildiği hibrit çalışmalar çoğunluktayken, matematiksel yöntemlerin yakınsama ve yaklaşık yöntemlerin hız avantajını birleştiren yöntemler nadirdir. Ayrıca esnek atölye tipi çizelgeleme probleminin literatürüne baktığımızda, çalışmalarda genelde sentetik verilerle geliştirilen yöntemlerin test edildiği, büyük boyutlu gerçek hayat problemlerine

çok yer verilmediği görülmektedir (Çınar ve diğerleri, 2015; Gao ve diğerleri, 2019). Literatürdeki bu boşluğun, 15 makine ve 15 işin bile büyük boyut sayıldığı esnek atölye tipi çizelgeleme problemi için gerçek hayat problemlerini çözmede gerek hesaplama süresi gerek hafıza sorunlarıyla karşılaşılması ve gerçek hayat problemlerinin dinamik yapısının etkili olduğu söylenebilir (Baykasoğlu ve diğerleri, 2020).

Orta ve büyük boyutlu problemlere geçildiğinde kesin yöntemlerin başarısız olması nedeniyle yaklaşık yöntemlere geçilmesiyle de hedef, amaç fonksiyonunun en iyilenmesinden ziyade istenilen sürelerde iyi bir çözüme ulaşabilmeye odaklanmaktadır. Ancak yaklaşık yöntemlerde hesaplama süresi düşük olsa da iş sayısı ve makine sayısı yüzlerce olduğunda bu hesaplama süresi çok artacaktır. Her ne kadar teknolojik gelişmelerle algoritmaların verimliliği artmış olsa da, birçok büyük boyutlu problem için optimal bir çözüme ulaşılamaması, hafıza ve süre problemleriyle hala karşılaşılmaktadır. Bu nedenle veri mimarisinin algoritma çözüm hızına ve hafızaya çok kritik etkisinin olduğu sonucuna varılmıştır.

Veri yapıları, bilgisayar ortamında verilerin etkin bir şekilde saklanmasını ve verimli bir şekilde veriyi yönetmeyi sağlar. Veri, yapı ve algoritma birbirinden ayrılmaz bileşenlerdir. Yapının sağlam bir mantıkla kurulması, verimli, doğru ve donanımı daha az yoran, hızlı çalışan algoritma geliştirmeyi kolaylaşır (Cedimoğlu, 2008). Uygun olmayan bir veri yapısı, yavaş çalışma sürelerine neden olabilir. Daha önce kullanılan algoritmaların uzun çözüm sürelerini gözönüne alarak ve veri yapılarının komplike algoritmalar üzerinde etkisini de değerlendirerek; bu tezde esnek atölye tipi çizelgeleme problemi için liste ve bağlı liste olmak üzere iki farklı veri yapısıyla algoritmalar kurulması düşünülmüştür. Sıralama alt probleminin süreye etkisini almamak için oluşturulan test örneklerini sıraladıktan sonra algoritmaların atama alt problemini çözme süreleri ele alınmıştır. Veri yapılarının algoritma hızına etkisini göstermek için esnek atölye tipi çizelgeleme atama alt probleminde problem büyüklüğünün değiştirildiği, farklı senaryoların dikkate alındığı bir çalışma gerçekleştirilmiştir.

Son olarak, karar verilen liste ve bağlı liste veri yapısıyla geliştirilen algoritmaların esnek atölye tipi problemi atama alt problemi çözümü gerçekleştirilmiş ve çalışma süresi performansları karşılaştırılmıştır. Literatürde bildiğimiz kadarıyla daha önce incelenmemiş bu çalışma sayesinde, büyük boyutlu problemlerin çözüm algoritmasında ileri veri yapılarının çalışma süresi performansında önemli kazançlar elde etmemizi sağladığı gösterilmiştir.

1.2. Kullanılan Metodoloji

Bu çalışmada veri yapılarının çizelgeleme algoritmaları üzerine etkisini incelemek için esnek atölye tipi çizelgeleme problemi ele alınmıştır. Bu çizelgeleme problemi uygulamada tercih edilmiştir çünkü endüstriyel alanlarda geniş uygulamaları olan bir problemdir ve çözümü NP-Zor sınıfında yer alan kombinatoryal optimizasyon gerektirmektedir(Türkyılmaz ve diğerleri, 2020). Atölye tipi çizelgeleme problemlerinde, n iş sayısını ve m makine sayısını temsil etmek üzere $(n!)^m$ adet mümkün çizelge vardır (Liu ve diğerleri, 2009). Bu nedenle en iyi çözümü elde etmek zordur ve literatürde çözümü için sezgisel ve meta-sezgisel algoritmalar kullanılmaktadır. Bu algoritmalarda ele alacağımız liste ve bağlı liste veri yapılarının kullanılması mümkündür.

Bu çalışmada, gerçek hayat problemleri boyutunda üretilen sentetik veriler üzerinde, ele aldığımız veri yapılarının esnek atölye tipi çizelgeleme probleminin atama alt probleminin çözüm sürelerine etkisini incelemek için deneysel tasarım yapılmıştır. Bağlı liste veri yapısının, liste veri yapısından çalışma süresi açısından daha iyi sonuç verdiği gözlemlenmiştir.

1.3. Literatüre Yapılan Katkılar

Bilindiği kadarıyla literatürde, çizelgeleme problemlerinde veri yapılarının algoritma performansına etkileri henüz incelenmemiştir. Çalışmanın konusu ve geliştirilen algoritmalar özgündür ve literatüre katkıdır. Esnek atölye tipi çizelgeleme esnek atölye tipi çizelgeleme problemi, Np-Zor olduğu için gerçek boyutlu problemlerin çözümü de güç olmaktadır. Ayrıca optimal çözümüne de istenen sürelerde ulaşılamamaktadır. Bu sebeple literatüre bakıldığında matematiksel modellerden çok sezgisel algoritmaların

geliştirildiği gözlenmektedir. Bu tez çalışması kapsamında kullanılan bağlı liste veri yapısının entegre edilmesiyle geliştirilen algoritmaların daha iyi performans vermesi sağlanabilir. Ayrıca, bu veri yapısı literatürdeki benzer yapıdaki Np-Zor problemlere de uygulanabilir.

1.4. Tezin Orgazinasyonu

Bu tez çalışması beş bölümden oluşmaktadır. İlk bölümde, çizelgeleme problemlerine genel bir bakış yapılmış, günümüzdeki önemi vurgulanmış daha sonra ele alınan çizelgeleme problemi özelinde yapılan çalışma özetlenmiş ve gerçek hayattaki zorlukları üzerinden çalışmanın önemine değinilmiştir. İkinci bölümde, çalışmada ele alınan esnek atölye tipi çizelgeleme problemi ve veri yapıları ile ilgili genel düzeyde literatür çalışmaları ortaya konmuştur. Dördüncü bölümde ele alınan problemin varsayımları, çözümünde kullanılacak veri yapıları hakkında genel bilgi verilmiştir. Daha sonra, iki farklı veri yapısıyla geliştirdiğimiz algoritmalar detaylı olarak anlatılmış ve büyük boyutlu problemler üzerinde karşılaştırmaları yapılmıştır. Son olarak beşinci bölümde, elde edilen sonuçlar özetlenmiş ve gelecek çalışmalar için önerilerde bulunulmuştur.

2. KAYNAK ARAŞTIRMASI ve KURAMSAL TEMELLER

Bu bölümde, çizelgeleme ve esnek atölye tipi çizelgeleme problemi ile ilgili kaynak araştırması yapılmıştır. Çizelgeleme problemlerinin karmaşıklığı ve veri yapıları ile ilgili genel bilgiler verilmiş ve bazı çalışmalara değinilmiştir.

2.1. Çizelgeleme Kavramı ve Esnek Atölye Tipi Çizelgeleme Problemi

Çizelgeleme, işletmelerde işlerin hangi sırayla, hangi kaynakta, hangi zaman aralığında yapılacağına verme sürecidir. Bu kararları verirken çeşitli kısıtlar bulunmaktadır ve kaynaklar sınırlıdır; amaç fonksiyonunu en iyilemek söz konusudur (Pinedo, 2012).

Çizelgeleme problemleriyle ilgili çalışmalar 1960'lı yılların başlarına dayanmaktadır (Baker ve Trietsch, 2009). Çizelgeleme problemlerinde, aktiviteler bir veya daha fazla performans ölçütünü optimize etmek amacıyla sınırlı kaynaklara bölüştürülür (Leung, 2004). Makinelerin kapasitesindeki ve işlerin işlem sıralarındaki kısıtlamalar bu problemin çözümünü oluşturduğundan çizelgeleme problemi atama ve sıralama işlemlerinden oluşmaktadır.

Giriş kısmında da bahsedildiği üzere; tez çalışması esnek atölye tipi çizelgeleme üzerine yoğunlaşmaktadır. Literatürde, esnek atölye tipi çizelgeleme problemini ilk olarak Brucker ve Schlie (1990) ele almıştır: Atölye tipi çizelgeleme probleminin bir genellemesi ve uzantısıdır (Yuan ve Xu, 013). Atölye tipi çizelgeleme problemi, bir makine seti üzerinde işlerin sıralanması ile ilgilidir. Esnek atölye tipi çizelgeleme probleminde ise işin bir operasyonu birden fazla makinede işlenebilir: her operasyon için alternatif makineler söz konusudur (Kaya ve Fığlalı, 2016). Makine esnekliği ve işlerin rotası esnek atölye tipi çizelgeleme problemini karmaşık hale getirir. Buna yönelik olarak Kacem vd. (2002) esnek atölye tipi çizelgeleme problemini her makinede işlenebilecek operasyon sayısına göre, tam veya kısmi esnek olarak sınıflandırmıştır. Tam esneklik olan esnek atölye tipi çizelgeleme probleminde tüm makineler tüm operasyonları gerçekleştirebilirken, kısmi esnek olan esnek atölye tipi çizelgeleme probleminde makineler bazı operasyonları gerçekleştirebilir (Kaya ve Fığlalı, 2016).

Esnek atölye tipi çizelgeleme problemlerinde farklı amaçlar ve farklı karakteristikler ele alınarak kesin, sezgisel, meta-sezgisel ve hibrit yöntemler geliştirilmiştir. Bu çalışmalar Gao vd. (2019); Xie vd. (2019); Chaudhry ve Khan (2016); Amjad vd. (2018) gibi farklı araştırmacılar tarafından derlenmiş ve yöntemlerin karşılaştırmaları yapılmıştır.

Bu çalışmada, iki farklı veri yapısı kullanarak geliştirilen algoritmaları karşılaştırmada esnek atölye tipi çizelgeleme problemi kullanılmıştır. Bu nedenle, literatürdeki çözüm yöntemlerinde kullanılan algoritmalar özetlenmiş ve gerçek hayat problemleri incelenmiştir.

Esnek atölye tipi çizelgeleme problemi çözümünde genetik algoritma (GA) sık başvurulan bir yöntem olmuştur. Zhang vd. (2020), tamamlanma zamanı ile toplam hazırlık ve taşıma süresini en aza indirmek için bir GA önermiştir. Defersha ve Rooyani (2020) ve Defersha ve Chen (2010) tarafından sunulan modelde ise iki aşamalı bir GA ele alınmıştır: ilk aşamasında sıralama kararları için bir kodlama gösterimi vardır ve ikinci aşamasında GA için bir prosedür geliştirmişlerdir. Chang vd. (2015) çözümünün etkinliğini artırmak için çaprazlamadan sonra Taguchi yöntemini kullanan bir GA önermiştir. Pezzella ve Ciaschetti (2008), başlangıç popülasyonunu oluşturmak, bireyleri seçmek ve yeni çözümler üretmek için farklı stratejileri birleştiren bir genetik algoritma sunmuştur.

Huang vd. (2016a), tamamlanma zamanını minimize etmeyi amaçlayan esnek atölye tipi çizelgeleme problemi için iki çaprazlama ve iki mutasyon yönteminin tasarlandığı bir GA gelişmiştir. Daha sonra, çiftleşme sürecinde (2016b) yeni bir adaptif çaprazlama ve mutasyon olasılığına sahip başka bir GA geliştirmiş ve büyük ölçüde daha iyi yakınsama sağlamışlardır.

Zhang vd. (2019), hazırlık ve taşıma sürelerini dikkate alarak tamamlanma zamanını ve toplam enerji tüketimini en aza indirmeyi amaçlayan esnek atölye tipi çizelgeleme problemi için bastırılmamış sınıflandırmalı genetik algoritma (NSGA-II) geliştirmiştir.

Brandimarte (1993), tamamlanma zamanını ve toplam ağırlıklı gecikmeyi dikkate alarak esnek atölye tipi çizelgeleme problemi için hiyerarşik bir tabu arama (TA) algoritması geliştirmiştir. GA ve TA stratejilerini birleştiren hibrit algoritmalar geliştirilmiştir (Li ve Gao, 2016; Romero ve diğerleri, 2018). Parçacık sürü optimizasyonu (PSO) ve TA algoritmalarının birlikte kullanıldığı çalışmalar da mevcuttur; parti akışı kısıtlaması ile sezgisel bir yöntem önerilmiştir (Zhang ve diğerleri, 2009). PSO ve yapay arı kolonisi algoritmasının melezleştirilmesini Muthiah vd. (2016) önermiştir. Geliştirilmiş benzetimli tavlamalı GA (Gu ve diğerleri, 2017), yapay bağışıklık mekanizmasını ve GA'yı birleştiren bir bağışıklık genetik algoritması önerilmiştir (Ren ve diğerleri, 2016). Wang vd. (2017), tamamlanma zamanını minimize etmeyi amaçlayan esnek atölye tipi çizelgeleme problemini çözmek için gelişmiş bir karınca kolonisi algoritması geliştirmişlerdir.

Zhang vd. (2018). GA'ya dayalı bir değişken komşuluk araması (VNS) geliştirilmiştir. Azzouz vd. (2017), sıraya bağlı hazırlık sürelerinin dikkate alındığı esnek atölye tipi çizelgeleme problemini öğrenme etkileri ile çözmek için GA ile değişken komşuluk arama (VNS) ve yinelenen yerel aramayı birleştiren bir evrimsel algoritma önermiştir. Yuan ve Xu (2015), tamamlanma zamanını, toplam iş yükünü ve kritik iş yükünü en aza indirmeyi amaçlayan çok amaçlı esnek atölye tipi çizelgeleme problemi için NSGA-II'ye dayalı bir memetik algoritma (MA) geliştirmiştir. Yi vd. (2016) tarafından, esnek atölye tipi çizelgeleme problemi için TA ve GA'nın bir kombinasyonu olan etkili bir MA önerilmiştir. Çınar vd. (2016), esnek atölye tipi çizelgeleme problemini çözmek için önceliğe dayalı gösterimi olan bir GA önermiştir. Gelişmiş çözümler elde etmek için algoritmalarına yinelenen yerel arama yerleştirmiştir.

Baykasoğlu (2002), tamamlanma zamanı, ortalama akış süresi, geciken iş sayısı, maksimum gecikme ve toplam makine boşta kalma süresini dikkate alarak esnek atölye tipi çizelgeleme problemini çözmek için benzetilmiş tavlama algoritmasını kullanmıştır. Kaplanoğlu (2016), çok amaçlı esnek atölye tipi çizelgeleme problemini çözmek için nesne yönelimli (OO) bir yaklaşımla bir benzetilmiş tavlama geliştirmiştir.

Purnomo (2016), esnek atölye tipi çizelgeleme problemini çözmek için bilgi tabanlı bir sistemle geliştirilmiş bir GA sunmuştur. Mati vd. (2001) esnek atölye tipi çizelgeleme problemi çözümünde entegre bir açgözlü sezgisel yöntem kullanmıştır. Wang ve Yu (2010), bakım faaliyetlerini dikkate aldıkları esnek atölye tipi çizelgeleme problemi çözümüne yönelik filtrelenmiş ışın arama tabanlı bir sezgisel algoritma geliştirmişlerdir. Tamssaouet vd. (2021), sıra bağımlı hazırlık sürelerini dikkate aldıkları esnek atölye tipi çizelgeleme problemi için tanımlanan birden çok kriteri optimize etmek için açgözlü, rassal, uyarlanabilir bir arama prosedürü ve benzetilmiş tavlama algoritması geliştirmişlerdir.

Marzouki ve Driss (2015), tamamlanma zamanını en aza indirgemeyi amaçlayan esnek atölye tipi çizelgeleme problemini çözmek için meta sezgisel kimyasal reaksiyon optimizasyonuna (CRO) dayanan yeni bir model önermiştir. Daha sonra, aynı problem çözümü için CRO'yu, TA ile melezleştirmişlerdir (2018).

Karmaşık gerçek hayat problemi literatürde esnek atölye tipi çizelgeleme problemi olarak modellenmiş ve çözümler üretilmiştir. Li vd. (2009), dikişsiz demir boruların üretimi için karar destek sistemi çizelgelemede doğrulanmış modifiye edilmiş bir GA kullanılmıştır. Silah üretim fabrikasında toplam gecikme, toplam makine boşta kalma süresi ve tamamlanma zamanı çoklu performans ölçütlerini en aza indirmek amacıyla GA ve Gruplayıcı GA (GGA)'ya dayanan yeni bir algoritma geliştirilmiştir (Chen ve diğerleri, 2012). Gomes (2013), esnek atölye tipi çizelgeleme problemini çözmek üzere, kesikli ve sipariş üzerine üretim endüstrisinde kullanılmak üzere yeni bir karışık tamsayılı doğrusal programlama modeli sunmuştur. Alvarez-Valdez vd. (2005), cam fabrikasında esnek atölye tipi çizelgeleme problemi çözmek için sezgisel bir yöntem geliştirmiştir. Tanev vd. (2004), plastik enjeksiyon makineleri fabrikalarındaki siparişler için, önceliklendirme kurallarını GA ile birleştirerek hibrit bir evrimsel algoritmayı (HEA) geliştirilmiştir.

Hosseinabadi vd. (2015), küçük ve orta ölçekli işletmelerdeki çok amaçlı esnek atölye tipi çizelgeleme problemi çözümü için, yerçekimi benzetim yerel arama algoritması geliştirmiştir. Birgin vd. (2014), bir işin operasyonları arasındaki öncelik kurallarının lineer bir düzen yerine keyfi yönlendirilmiş döngüsel bir grafik tarafından verildiği,

geliştirilmiş bir esnek atölye tipi çizelgeleme problemi üzerine çalışmıştır: yeni bir karışık tamsayılı doğrusal programlama modeli formülize edip, baskı endüstrisindeki gerçek verilerden oluşturulmuş örnekleri çözmüşlerdir.

Hansmann vd. (2014), vagonların yönetim merkezlerinde esnek atölye tipi çizelgeleme probleminin karışık tamsayılı programlama modeli üzerine çalışmışlar ve Dal Sınır algoritması kullanmışlardır. Grobler vd. (2010), çok amaçlı esnek atölye tipi çizelgeleme problemini sıraya bağlı setup süreleri, yardımcı kaynaklar ve makine arıza süresi ile çözmek için dört tür PSO tabanlı sezgisel yöntem geliştirmiştir. Gerçek müşteri verileriyle yapılan karşılaştırma sonuçları bu problem için, öncelik tabanlı PSO algoritmasının, yaygın olarak kullanılan mevcut kural tabanlı algoritmalardan daha iyi performans gösterdiğini ortaya koymuşlardır. Baskın olmayan çözümler kümesi elde etmek adına, baskı endüstrisinden esnek atölye tipi çizelgeleme problemi için iki TA algoritması önermişlerdir.

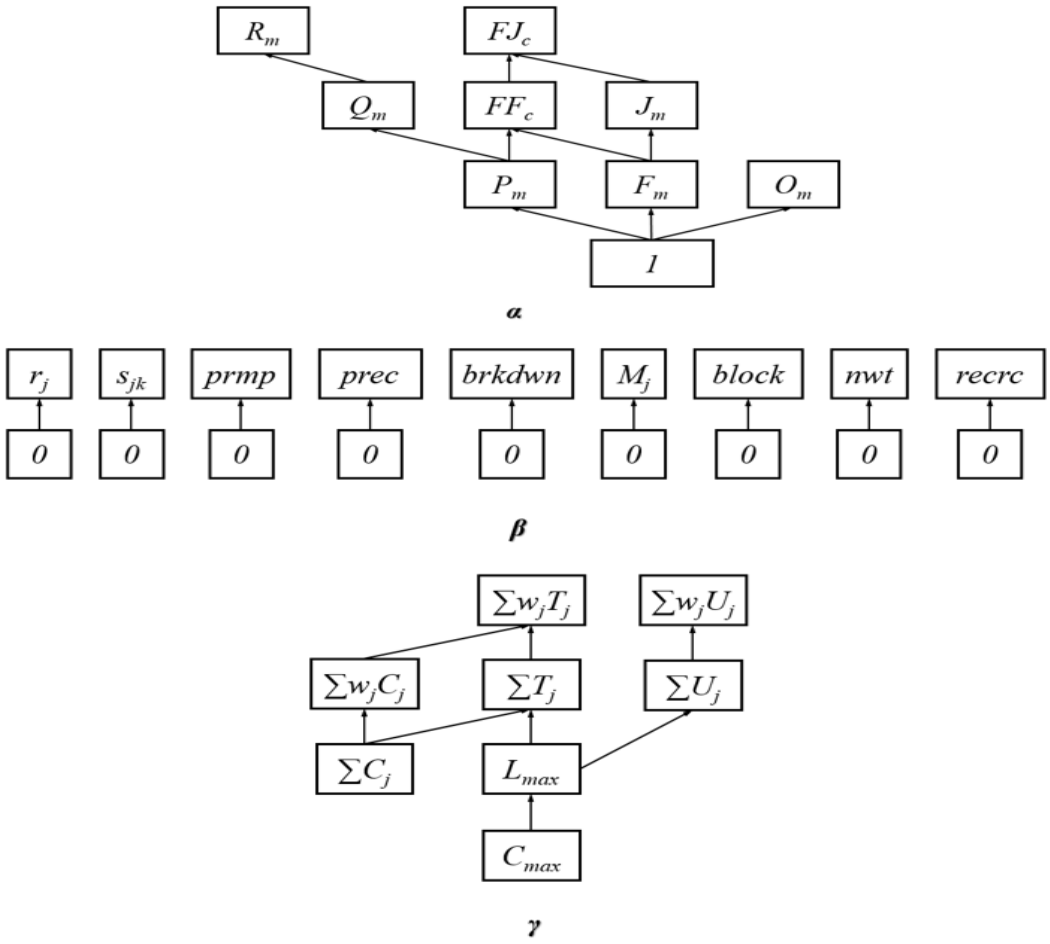
Çizelgeleme problemleri literatürde araştırmacılar tarafından yoğun olarak çalışılmaktadır; ancak, veri yapılarının süreye etkisi üzerine çalışma bulunmamaktadır. NP-Zor olmasından dolayı da literatürde yeni çözüm algoritmalarıyla çözülmeye devam edecek bu kombinatoryal probleme önerdiğimiz ileri veri yapısı yukarıda adı geçen tüm algoritmalara entegre edilip bunların daha iyi performans vermesini sağlayabilir.

2.2. Çizelgeleme Problemlerinin Karmaşıklık Sınıflandırması

Çizelgeleme problemlerinin ve çözüm tekniklerinin ilişkisine dair faydalı bir bakış açısı, karmaşıklık teorisi olarak bilinen bilgisayar biliminin bir dalındaki gelişmelerden gelmektedir. Karmaşıklık kavramı, bir çözüm algoritmasının gerektirdiği hesaplama çabasını ifade etmektedir (Baker ve Trietsch, 2009). Algoritmanın zaman karmaşıklığı Büyük-O notasyonu ile tanımlanır. Karmaşıklık teorisi, pratikte karşılaşılan problemlerin algoritmaları ve önemli örneklerin her ikisinin de hesaplamalarının sınıflandırılmasını konu alır.

Çizelgeleme problemleri kombinatoryal optimizasyon problemidir. Küçük boyutlu problemler için problemin çözümü zor görünmeyebilir fakat iş ve makine sayısı arttıkça yani problem büyüdükçe muhtemel çizelge sayısı arttığından dolayı hesaplama zorlaşmaktadır. Karmaşıklık teorisi ile çoğu çizelgeleme probleminin NP-Zor olduğu ispatlanmıştır (Garey ve Johnson, 1976; Sun ve diğerleri, 2021).

Çizelgeleme problemlerinin tanımında "α | β | γ" şeklinde bir gösterim kullanılmaktadır. "α" o probleme ait makine ortamını, "β" kısıtları ve "γ" ise amaç fonksiyonunu göstermektedir (Graham ve diğerleri, 1979). Deterministik çizelgeleme problemlerinin karmaşıklık hiyerarşisi Şekil 2.1'de yer almaktadır. Yukarıdan aşağıya doğru gittikçe problemler karmaşıklaşmaktadır (Sirkeci, 2015).



Şekil 2.1. Karmaşıklık hiyerarşisi

Algoritma için gerekli olan hesaplamaların sayısı, n'nin bir fonksiyonu ile yukarıdan sınırlandırılır. Eğer, n büyüdükçe fonksiyonun büyüklüğü polinom ise, algoritma polinomdur. Polinom zaman kavramı, algoritmaların hesaplama gereksinimleri ve karşılaşılan zorluklara göre karmaşıklık teorisinde birkaç karmaşıklık sınıfına yol açmaktadır. Bu sınıfa göre problemler zor veya kolay olarak ayrıştırılmaktadır. Problemin zorluk derecesinin bilinmesi problemin çözümü için en iyi yöntemin uygulanmasını sağlar (Polatlı, 2011).

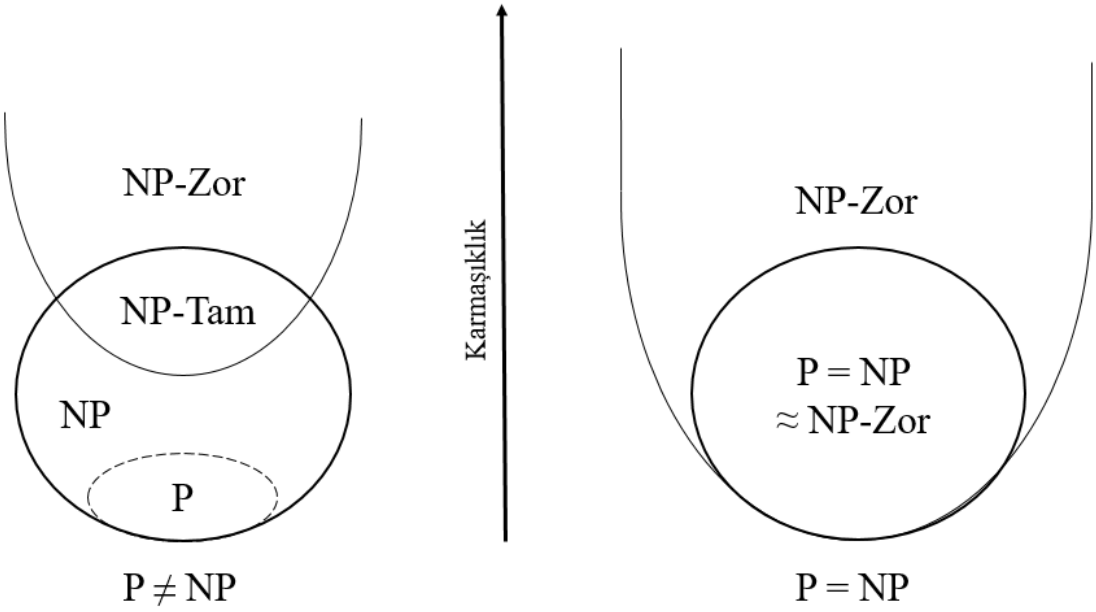
Eğer belirli bir problemin her örneğini çözecek bir polinom zaman algoritması geliştirilebilirse, kombinatoryal optimizasyon problemlerinin bu sınıfı kolay olarak adlandırılır ve P ile gösterilir. Polinom zaman algoritmalarının çözümlenmesi, incelenmesi kolaydır ve bu algoritmalar sorunu kısa sürede çözüme ulaştırır (Biroğul, 2005).

Eğer bir problem için o problemi çözecek etkili algoritmalar bulunamazsa, bu problem zor olarak adlandırılır. Polinom zaman algoritması ile çözülemeyen zor problemler veya üstel işlem zamanı gerektiren problemler, üstel zaman algoritması ile ele alınır. Deterministik olmayan algoritmalar yardımıyla polinom zamanda çözülebilen karar problemlerinin sınıfı NP (Non-Deterministic Polynomial Time) sınıfı olarak adlandırılır. NP, NP-Tam (NP-Complete) olarak adlandırılan problemlerin bir alt kümesini içerir. Bu alt kümedeki, her bir problem, NP sınıfına aittir. Eğer bu problem için etkili bir algoritma mevcutsa, NP sınıfındaki her bir problem için etkili bir algoritma mevcuttur. Bunun anlamı, NP-Tam sınıfı problemlerinin NP sınıfındaki en zor problem sınıfı olmasıdır. Eğer NP sınıfındaki bütün problemler polinom olarak bir probleme indirgenebilirse, bu problem NP-Zor (NP-Hard) sınıfına aittir denir. Başka bir deyişle, eğer bir problem NP-Zor sınıfına aitse, sadece P=NP olan bir polinom zaman algoritması ile çözülebilir (Yurttakal, 2014).

Eğer NP sınıfındaki her problem polinomiyal zamanda, bir P problemine indirgenebiliyorsa, ilgili P problemine NP-Zor problem denir. Eğer P probleminin kendisi de NP sınıfında bir problemse bu durumda P problemine NP-Tam problem denir (Reeves, 1995).

NP-Tam sınıfındaki herhangi bir problem için polinom zamanlı bir algoritma bulunabilirse, bu sınıftaki tüm problemler için de polinom zamanlı bir algoritma bulunabilir. Bu durumda P=NP olduğu gösterilmiş olacaktır (Kellegöz, 2006).

Kombinatoriyel optimizasyon problemlerinin çoğu NP-tam, polinom zaman sınırı olmayan problemler sınıfına girmektedir. NP problemlerine, polinom algoritma geliştirilememiştir. NP kapsamında yer alan problemler için asıl optimum çözüm yerine yakın çözümler tercih edilir.



Şekil 2.2. P-NP arasındaki ilişki

Araştırmacılar, NP içerisinde P'ye asla taşınamayacak problemlerin olduğunu göstermişlerdir. Yani, çok zor olmasından dolayı bazı problemlere ait polinom zaman karmaşıklık fonksiyonuna sahip algoritmaların hiçbir zaman bulunamayacağına inanırlar. "P=NP midir?" sorusuna hala cevap aranmaktadır. Eğer P=NP olduğu ispat edilirse çözümü olmayan problemlere de çözümler bulunacaktır (Yurttakal, 2014). Bu süreçte de çizelgeleme problemleri için yeni algoritmalar geliştirilecektir.

2.3. Veri Yapıları

Çizelgeleme problemlerinin birçok formunun çözümlerinde dal-sınır, dinamik programlama veya tam sayılı programlama gibi yöntemlerin kullanılmasına sıklıkla

rastlanmaktadır. Ancak bu yöntemler güçlü bir bilgisayara ve çok fazla hesaplama zamanına ihtiyaç duymaktadır ve büyük boyutlu problemlerin çözümünde kullanılmaları uygun olmamaktadır. Bu nedenle yaygın olarak çözüm yöntemlerinde sezgisel algoritmalar geliştirilmektedir. Bu algoritmaların sıklıkla kullanılanlarına 2.1 numaralı bölümde kısaca değinilmişti.

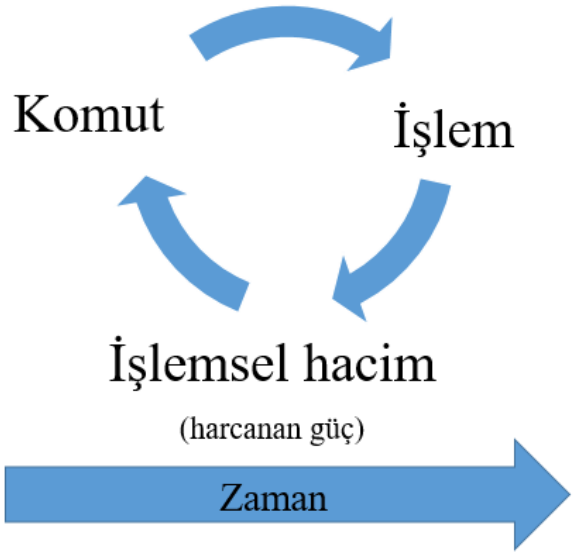
Bir problemi çözmek üzere tasarlanan yola ya da problemi kaynağından çözüm noktasına taşıyan süreç ve metotların bütününe algoritma adı verilir. Algoritmalar düzenlenirken problemi ifade etmek için giriş ve çıkış unsurlarına ihtiyaç vardır. Hazırlanan programın amacı doğrultusunda elde edilen bilgiye çıkış verisi, karşılaştığımız probleme ya da elde etmek istediğimiz sonuca göre algoritmamızca değişikliğe uğrattığımız bilgiyeyse giriş verisi adı verilir.

Bilgisayarlar ve yazılımlar kullanarak artan problem çözme kapasitemiz sonucunda, problemlerin çözümü için algoritmalar oluşturulma aşamasında bilgi birikimimizin artış niteliğine bağlı olarak problemleri yapılandırabilme yöntemlerimiz de çeşitlilik göstermiştir. Bir algoritmayı oluşturabilmemiz için öncelikle problemde kullanılacak bilgilerin hangi veri türü cinsinden ifade edilmesi gerektiğinin tespit edilmesi gerekir. Farklı veri tipleri farklı türde mantıksal bağlamlarla bir araya gelip farklı işlemlerle ilişkilendirilebilir. Algoritmaları anlamlandırıp, problemlerin çözümsel niteliklerine dair yorum yapabilmemiz için veri tiplerini ve veri yapılarını incelememiz gerekmektedir.

Veri yapısı, verinin veya bilginin bellekte tutulma şeklini veya düzenini gösterir. Tüm programlama dillerinin, genel olarak, tamsayı, kesirli sayı, karakter ve sözcük saklanması için temel veri yapıları vardır. Bir program değişkeni bile basit bir veri yapısı olarak kabul edilebilir. Temel olarak üç farklı veri yapısı vardır. Bunlar; temel, basit ve birleşik olarak gruplandırılabilir. Temel veri yapıları başlıca algoritma oluşturmada kullanılabildiği gibi diğer veri yapılarının oluşturulmasında da kullanılabilir.

Veri yapılarını kabaca tanımladıktan sonra algoritmayla arasındaki ilişkiyi inceleyebiliriz. Bir algoritma, bir problemi işlemsel ve mantıksal olarak kendisine sunulan bilgiden yola çıkarak amaçlandığı dizayna göre sistematik olarak adım adım

ilerletip sonuca ulaştırır. Gelen verinin işlendiği veri yapısı algoritmanın çalışma prensibi içerisinde, farklı işlem boyutu ve kaynak hacmi gerektirir. Bir tasarımcı, ele aldığı problemi hangi yönüyle ve hangi verim açısından çözmesi gerektiğini de belirlemeli ve buna uygun veri yapısı seçmelidir. Algoritmalar temel olarak; komut, işlem, işlem hacmi, ve bu üçlünün birlikte işleyişi esnasında geçen zaman ögeleriyle tanımlanabilir (Şekil 2.3).



Şekil 2.3. Algoritmaların işleyişi

Sonsuz olmayan bir süre içerisinde, belirli komutlarla yapılan belirli işlemler ve bu işlemlerin yapılmasına olanak tanıyan işlem gücü neticesinde bir algoritma çalışma prensibini tamamlamış olur. Bu çalışma prensibi içerisinde iyileştirilmek ya da elde edilmek istenen niteliğe uygun veri yapısı seçimi bu yüzden önemlidir. Bir tasarımcı, ele aldığı problemi hangi yönüyle ve hangi verim açısından çözmesi gerektiğini de belirlemeli ve buna uygun veri yapısı seçmelidir.

Bir algoritmanın çalışma süresi algoritmanın adımları ile ölçülür. Bu ise T(n) şeklinde bir fonksiyon ile ifade edilebilir. Oluşan fonksiyonun en hızlı büyüyen terimi algoritmanın işlem zamanını gösterir. Bu ise T(n)=O(g(n)) şeklinde gösterilir(Tanyıldızı, 2012).

Bilgisayar uygulamasında, bir yazılım geliştirirken birçok algoritmaya ihtiyaç duyulur. Örneğin, arama algoritması, sıralama algoritması, matris veya vektörel işlem algoritması,

graf algoritması, bir matematiksel modelin çözülmesi algoritması, gibi birçok algoritma türü vardır ve uygulama geliştirirken, bunların biri veya birkaçı her zaman kullanılır.

Literatürde algoritmalar ve veri yapılarına etkileri üzerinde çok fazla bir çalışma mevcut değildir. Esen (2018) tez çalışmasında, Sudoku bulmacasında kuyruk liste veri yapısı tabanlı paralel önce-derine arama yöntemiyle çözüm önermiş ve geleneksel önce-derine arama yöntemine göre daha hızlı çözdüğünü göstermiştir. İnner (2004) tez çalışmasında, sekizli ağaç yapısı ve sıkıştırılmış sekizli ağaç yapısı kullanarak Barnes-Hut ve Hızlı çok-kutup algoritmalarını çalışmışlar ve her iki veri yapısının performanları, çalışma zamanları ve doğruluk oranlarını karşılaştırmışlardır. Veri yapıları ve algoritmalar arasındaki ilişkilerin irdelendiği çalışmaların az olması; tezdeki çalışma konusunun yeniliği üzerine bir gösterge olup; bu çalışmada da esnek atölye tipi çizelgeleme problemi üzerinden çifte bağlı doğrusal liste veri yapısıyla geliştirilen algoritmanın çalışma süresi performansını iyileştirmesi çalışılmıştır.

3. MATERYAL ve YÖNTEM

Bu bölümde, tez çalışmasında uygulamada kullanılan problem hakkında bilgiler verilmiştir ve problemin çözümüne yönelik farklı veri yapılarıyla geliştirilen algoritmalar detaylı olarak açıklanmıştır. İki alt bölümden oluşmaktadır.

3.1. Uygulama Aşamasında Kullanılan Problemin Tanımı ve Varsayımları

Bu tez çalışmasında, esnek atölye tipi çizelgeleme problemi üzerinden veri yapılarının çalışma sürelerine etkisi incelenmiştir. Bu çizelgeleme problemi çeşitli amaç fonksiyonlarıyla ele alınsa da işlerin operasyonlarının en uygun makinelere atandığı ve en uygun sıralamanın belirlendiği problemdir. Yani, önceki bölümlerde belirtildiği gibi esnek atölye tipi çizelgeleme problemi atama ve sıralama olmak üzere iki alt problemden oluşmaktadır. Fakat bu çalışmada önerilen veri yapıları sıralama alt problemini dikkate almadan atama alt problemine yönelik geliştirilmiştir. Sıralama alt probleminin çalışma süresine etkisi göz ardı edilmiştir. Bunun nedeni, çalışmamızda veri yapılarının süreye etkilerini daha net görmek içindir. Sıralamada çeşitli algoritmalar kullanılabilir ve sıralama algoritmalarının CPU zamanları çok değişiklik göstermektedir. Farklı senaryolarda sıralama algoritmalarımız da değişebileceğinden dolayı sıralama alt problemine harcanan zaman göz ardı edilmiştir. İşler sıralandıktan sonra programda atama alt probleminin deneyleri yapılmıştır.

Operasyonlar sıralanırken önce önşartları olmayan operasyonlar atanmıştır. Daha sonra bütün operasyonlar atanana kadar önşartlarında bu atanan işleri içeren operasyonlar bulunmuş ve içlerinden rastgele seçilen bir tanesi sıraya konmuştur. Bu sıralamanın yapılmış hali bir json dosyasına kaydedilmiştir. Atama alt problemi çalıştırılırken bu json dosyasından veri okunur, böylece sıralanmış operasyonlar atama algoritmasında çalıştırılmaktadır.

Esnek atölye tipi çizelgeleme probleminde, | J | sayıda iş, $|M|$ sayıda makinede önceden ($o1$, $o2$, . . . , $o|Oj|$) olarak belirlenmiş üretim rotasını izleyecek şekilde ve her bir operasyon O_J = {1,2, … , $|O_j|$ } uygun makine grubundaki Mj,o makinelerinden herhangi

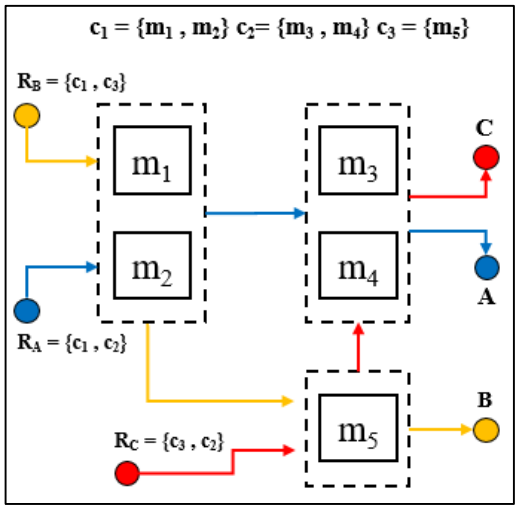
birinde işlenmektedir. Burada $|O_j|$ j işi için gerekli operasyonların sayısını ifade etmektedir.

Eğer tüm makineler bütün operasyonları işleyebilecek kapasiteye sahipse, problem tam esnek olan esnek atölye tipi çizelgeleme problemi olarak, eğer M makine kümesinde yer alan bazı makineler bazı operasyonları işleyemiyorsa bu tür problemler ise kısmi esnek atölye tipi çizelgeleme problemi olarak adlandırılmaktadır (Kacem ve diğerleri, 2002).

Bir esnek atölye tipi çizelgeleme probleminde genel olarak aşağıdaki varsayımlar dikkate alınmaktadır: (Kaya ve Fığlalı, 2016)

- Problem deterministik yapıdadır.
- Her bir iş farklı operasyonlardan oluşabilir.
- Her operasyon, işlem süresince kesintiye uğramadan gerçekleşmektedir.
- Her bir operasyon için en az bir tane olmak üzere birden fazla makine alternatifi vardır.
- Bütün işler sıfır anında işlenmeye hazırdır.
- İşlem sürelerinin önceden bilindiği ve hazırlık zamanlarını da içerdiği kabul edilmektedir.
- Her bir operasyonun bütün makine alternatiflerindeki işlem süreleri aynıdır.
- Makineler çizelgeye başlarken boştur, çizelgeleme periyodu boyunca bozulmaz ve bakım istemez.
- Makineler arası taşıma sürelerinin ihmal edildiği varsayılmaktadır.

Şekil 3.1'de görüldüğü gibi her işin farklı rotası ve her makine grubunda paralel makineler vardır. A işi iki operasyondan oluşmaktadır ve her bir operasyon sırasıyla c1 ve c2 makine grubundaki özdeş makinelerden birinde işlenmek zorundadır. Bir işin bir operasyonu bitmeden diğer operasyon başlayamaz. Bir makine grubundaki tüm makineler paraleldir (örneğin c1 makine grubundaki m1 ve m2 makineleri).



Şekil 3.1. Esnek atölye tipi üretim

3.2. Performans Değerlendirmesi için Geliştirilen Test Senaryoları

Geliştirilen algoritmaların performanslarını test ederken çeşitlilik olması için geliştirilen senaryolar bu bölümde anlatılmaktadır. Senaryolar, sayısal örnekler üzerinden detaylı olarak anlatılmaktadır. Sayısal örnekler senaryoların anlaşılması için rastgele üretilmiştir.

3.2.1. Senaryo 1

İşlere ait operasyonların tüm makine gruplarında işlenebildiği yani tam esnekliğin olduğu ve operasyonların birbiri ardına gelmesi yani bir işin ikinci operasyonu ilk operasyonu tamamlanmadan bitemez şeklindeki ön şartlar dışında operasyonlara ek ön şartların verilmediği durumdur.

Çizelge 3.1'de makine sınıfı tanımlanmıştır. Örnekte altı adet paralel makine yer almaktadır. Operasyonların tümü bu makinelerden birinde işlenebilmektedir (tam esneklik). Operasyonların işlem süresi ve ön şartları Çizelge 3.2'de yer almaktadır. Operasyonlar, ön şartlarının sağlandığı, yani sıralamasının ön şartlarına göre başta yapıldıktan sonra, atanabileceği makinelerde ilk başlama zamanı küçük olan yeri bulduğunda atama gerçekleştirilir. Bu kurala göre operasyonların hangi makinede hangi zaman aralığında işlendiği Çizelge 3.3'te verilmiştir. Bu sayısal örneğe ait Gantt Şeması Şekil 3.2'de görülmektedir. Aşağıda verilen sayısal örnekte çizelge uzunluğu 120 dakikadır.

Çizelge 3.1. Senaryo 1 sayısal örneği makinelerin tanımı

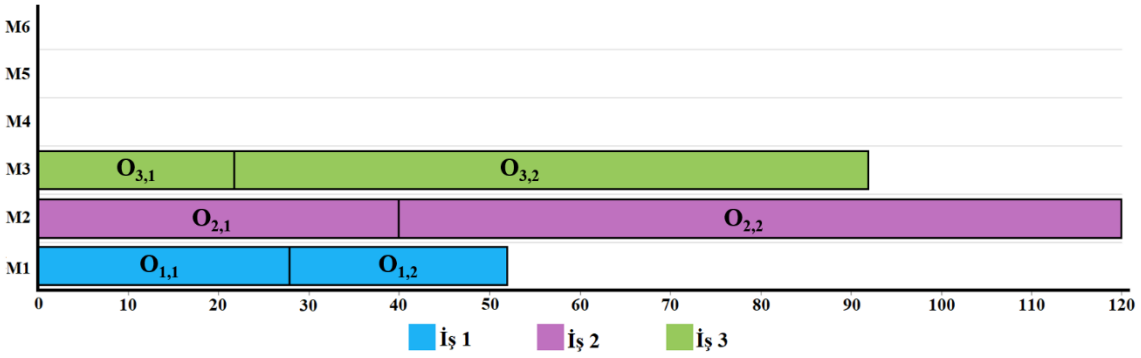
Makine ID	Makine Grubu
1	1
2	1
3	1
4	1
5	1
6	1

Çizelge 3.2. Senaryo 1 sayısal örneği operasyonların tanımı

Operasyon ID	İş ID	İşin Operasyonu	Makine Grubu	Süre(dk)	Önşart ID
1	1	1	1	28	-
2	1	2	1	24	1
3	2	1	1	40	-
4	2	2	1	80	3
5	3	1	1	22	-
6	3	2	1	70	5

Çizelge 3.3. Senaryo 1 sayısal örneği için uygun çözüm

Operasyon ID	Atandığı Makine	Başlama Zamanı	Bitiş Zamanı
1	1	0	28
2	1	28	52
3	2	0	40
4	2	40	120
5	3	0	22
6	3	22	92



Şekil 3.2. Senaryo 1 sayısal örneği için Gantt Şeması

3.2.2. Senaryo 2

Bu senaryoda ise operasyonların sadece belirli makinelerde işlenebildiği kısmi esneklik dikkate alınmıştır. Her makine bir makine grubunda yer almaktadır. Ayrıca, operasyon sırasına ek ön şartlar verilmiştir. Aşağıda verilen sayısal örnekte çizelge uzunluğu 120 dakikadır. İlgili senaryo için makineler Çizelge 3.4'te tanımlanmıştır. Örnekte altı adet makine vardır ve üç makine grubu bulunmaktadır. Ve örnek Örnek verecek olursak 1 ve 2 paralel makinelerdir. Operasyonların tümü bu makine gruplarında yer alan makinelerden birinde işlenebilmektedir. Operasyonların işlem süresi ve ön şartları Çizelge 3.5'te yer almaktadır. Veri seti oluşturulurken ön şartları sağlayacak şekilde operasyonlar sıralanmıştır, fakat bu sıralama için geçen süre veri yapılarının çalışma süresine etkisi dikkate alınmamıştır. Atama algoritmasında çalışma süresine etki etmemesi için veri oluşturma aşamasında çalıştırılan sıralama algoritmasının sözde şu şekildedir:

```
PUBLIC METHOD Sıralama()
        BEGIN
            WHILE (Tüm operasyonlar sıralanıncaya kadar)
                    Önşartlarıyla, sıralanan önşartları eşit olan operasyonları bul.
                    Bu operasyonlardan rastgele birini seç.
                    Sıralama listesine bu operasyonu ekle.
                    Sıraya eklenen bu operasyonu, önşartlarında bulunduran diğer tüm
            operasyonların sıralanan ön şartlarına bu operasyonu ekle.
            END WHILE
        END
```

Sözde koda göre, ilk önce tüm operasyonlar rastgele oluşturulmuştur. Bu oluşturulan operasyonlar daha sonra sıralanırken, operasyonların hem ön şart listesi hem de sıralanan ön şart listesine bakılır. Önşartları sağlanan operasyonlar bulunur ve içlerinden rastgele biri seçilir ve sıralama listesine eklenir. Sıraya eklenen bu oeprasyonu ön şartlarında içeren tüm operasyonların da sıralanan ön şartları listesine bu operasyon eklenir. Bu işlem tüm operasyonlar sıralanınca kadar devam eder.

Ön şartlar dikkate alınarak oluşturulan bu sıralamaya göre operasyonlar atanabileceği makine grubunda yer alan makinelerden en erken atanabileceği zamanı küçük olanına atanmaktadır. Verilen sayısal örneğin uygun bir çözümü Çizelge 3.6'da verilmiştir. Çizelge 3.6'da yer alan sonuçlara göre oluşturulan Gantt şeması Şekil 3.3'te görülmektedir. Aşağıda verilen sayısal örnekte çizelge uzunluğu 120 dakikadır.

Çizelge 3.4. Senaryo 2 sayısal örneği makinelerin tanımı

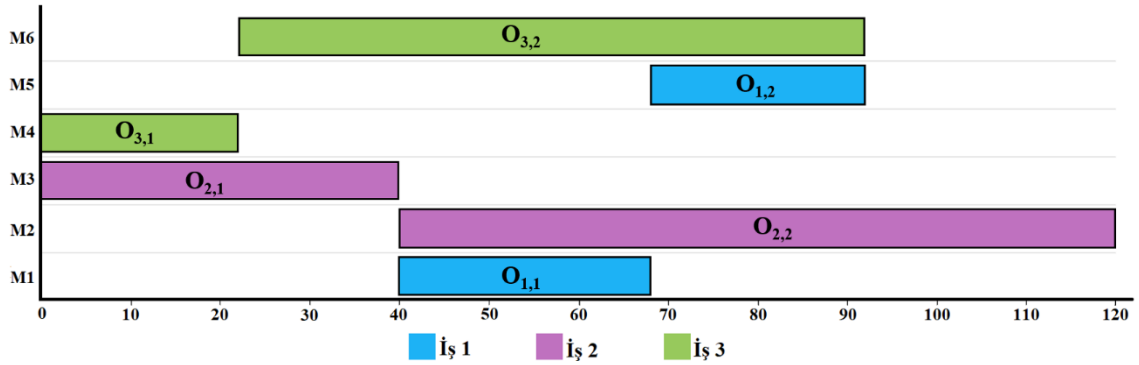
Makine ID	Makine Grubu
1	1
2	1
3	2
4	2
5	3
6	3

Çizelge 3.5. Senaryo 2 sayısal örneği operasyonların tanımı

Operasyon ID	İş ID	İşin Operasyonu	Makine Grubu	Süre(dk)	Ön şart ID
3	2	1	2	40	-
1	1	1	1	28	3
2	1	2	3	24	1
4	2	2	1	80	3
5	3	1	2	22	-
6	3	2	3	70	5

Çizelge 3.6. Senaryo 2 sayısal örneği için uygun çözüm

Operasyon ID	Atandığı Makine	Başlama Zamanı	Bitiş Zamanı
3	3	0	40
1	1	40	68
2	5	68	92
4	2	40	120
5	4	0	22
6	6	22	92



Şekil 3.3. Senaryo 2 sayısal örneği için Gantt Şeması

3.2.3. Senaryo 3

Ele aldığımız bu senaryoda da Senaryo 2'de olduğu gibi kısmi esneklik ve işlerin operasyon sırası dışında ek ön şartlar dikkate alınmıştır. Her makine bir makine grubunda yer almaktadır. Ancak, bir operasyon birden fazla makine grubunda aynı zamanda işlem görebilmektedir. Makine olarak operatör gibi kaynaklar da düşünülerek bu senaryoda birden fazla makine grubuna aynı zamanda atama yapılmaktadır. Aşağıda verilen sayısal örnekte çizelge uzunluğu 140 dakikadır. İlgili senaryo için makineler Çizelge 3.7'de tanımlanmıştır. Örnekte altı adet makine vardır ve üç makine grubu bulunmaktadır. Operasyonların tümü bu makine gruplarında yer alan makinelerden birinde işlenebilmektedir. Operasyonların işlem süreleri, ön şartları ve aynı anda atanması gereken makine grupları Çizelge 3.8'de yer almaktadır. Önceki senaryodaki gibi veri üretilirken sıralama yapılmış ve bunun için geçen zaman veri yapılarının çalışma süresine etkisi etmemiştir. Ön şartlar dikkate alınarak oluşturulan bu sıralamaya göre operasyonların atanması gereken her bir makine grubundaki makinelerden birine boş oldukları ilk uygun aynı zaman aralığında atanmaktadır.

Çizelge 3.7. Senaryo 3 sayısal örneği makinelerin tanımı

Makine ID	Makine Grubu
1	1
2	1
3	2
4	2
5	3
6	3

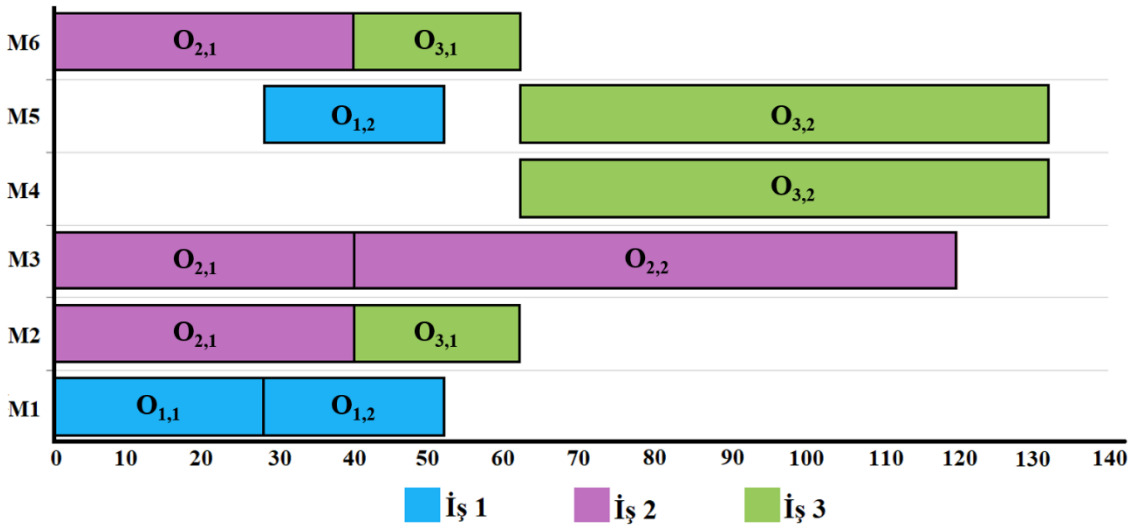
Çizelge 3.8. Senaryo 3 sayısal örneği operasyonların tanımı

Operasyon ID	İş ID	İşin Operasyonu	Makine Grubu	Süre(dk)	Önşart ID
1	1	1	1	28	-
2	1	2	1,3	24	1
3	2	1	1,2,3	40	-
4	2	2	2	80	3
5	3	1	1,3	22	-
6	3	2	2,3	70	5

Çizelge 3.9. Senaryo 3 sayısal örneği için uygun çözüm

Operasyon ID	Atandığı Makine	Başlama Zamanı	Bitiş Zamanı
1	1	0	28
2	1,5	28	52
3	2,3,6	0	40
4	3	40	120
5	2,6	40	62
6	4,5	62	132

Çizelge 3.9'da verilen sayısal örneğin uygun bir çözümü ve bu çözüme yönelik Gantt şeması Şekil 3.4'te yer almaktadır.



Şekil 3.4. Senaryo 3 sayısal örneği için Gantt Şeması

3.3. Farklı Veri Yapıları ile Geliştirilen Algoritmalar

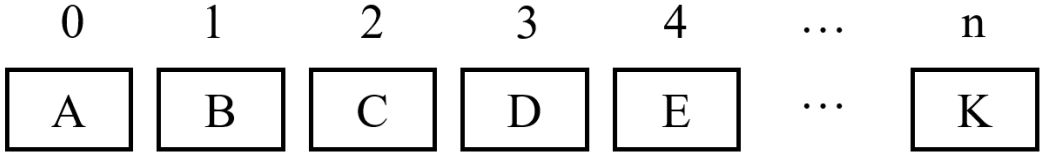
Bu tez çalışmasında, veri yapılarının çalışma sürelerine etkisini incelemek için liste ve bağlı liste olmak üzere iki farklı veri yapısıyla algoritmalar geliştirmiştir. Bu algoritmalar detaylı olarak iki alt bölümde açıklanmıştır.

3.3.1. Liste Veri Yapısı ile Geliştirilen Algoritma

Çalışmamızda bağlı liste yapısını karşılaştırmak için sezgisellerde algoritmaları geliştirirken yaygın olarak kullanıldığı için liste veri yapısını seçtik.

Listeler en önemli temel ve ilkel veri yapısıdır. Listeler için birçok farklı uygulama mümkündür. Bazı programlama dillerinde listeler diziler olarak da uygulanabilir. Bir liste hemen hemen her şeyi içerebilir, örneğin bir tamsayı listesi [1; 2; 3; 4; 5], bir metin listesi [a; b; c; d; e], veya tanımlanmış bir sınıfın listesi olabilir. Örnek verecek olursak Makine Grup ismi ve Makine isimlerinden oluşan bir makine grubu listesi şu şekilde gösterilebilir : [{“Makine Grubu” : “G1”, “Makineler” : [“M1”, “M2”]}, {“Makine Grubu” : “G2”, “Makineler” : [“M3”, “M4”]}]. Yani listeler, bilgisayarlarla iletişim kurarken anlatmak istediğimizi soyut bir düzeyde tanımlayarak konuşmamızı sağlamaktadır(Storer, 2012).

Listelerin yapısı Şekil 3.5’te gösterilmiştir. Şekilde n boyutlu bir listede her bir düğüm bir veri tutmaktadır. Örneğin, Listenin 1 indeksli elemanı B değerini tutmaktadır. Listelerle işlem yapmak basittir ve her bir işlem O(n) zamanda çalışır.



Şekil 3.5. n indeksli bir liste yapısı

Programımızda, Liste yapısıyla makinelerdeki boşluklar şu şekilde tutulmaktadır:

```
CLASS Makine
BEGIN
    PUBLIC String MakineID
    PUBLIC List<Int> BoşlukListesi = new List<Int>()
END CLASS
```

Makinenin boşluk listesi başlangıçta çizelge uzunluğu boyunca 100 değeri ile doldurulur. Bu değer rastgele seçilmiştir. Değeri 100 olan liste indeksi makinenin boş olduğu zamanı temsil eder. Makinenin çalıştığı zaman indeksinde listede 0 değeri yer alır. Bu 0 değeri de rastgele seçilmiştir. Makinenin hangi zaman dilimlerinde dolu veya boş olduğu 0 ve 100 değerlerinden anlaşılmaktadır.

Makine ID M1 olan, boşluk listesi {100,100,100,100,100,100,100,100,100,100} olan bir makine sınıfı çizelge uzunluğu 10 birim süre olan henüz iş atanmamış bir makineyi temsil etmektedir (Şekil 3.6).

Makine										
Zaman	1	2	3	4	5	6	7	8	9	10
Liste Değeri	100	100	100	100	100	100	100	100	100	100

Şekil 3.6. İş atanmamış makine için boşluk listesi

Makine ID M1 olan, boşluk listesi = {0,0,100,100,100,100,100,100,100,100} olan bir makine sınıfı çizelge uzunluğu 10 birim süre olan ilk iki birim süresine iş atanmış bir makineyi temsil etmektedir (Şekil 3.7).

Makine										
Zaman	1	2	3	4	5	6	7	8	9	10
Liste Değeri	0	0	100	100	100	100	100	100	100	100

Şekil 3.7. Belli zaman aralığına iş atanmış bir makine için boşluk listesi

Senaryo 1 için oluşturulan algoritmanın sözde kodunda makinelerin bilgisi, makine sınıfında tutulmaktadır. Makine grupları da MakineGrup sınıfındaki gibi tanımlanmıştır.

```
CLASS MakineGrup
BEGIN
    PUBLIC String MakineGrupID;
    PUBLIC List<String> MakineIDs;
END CLASS

CLASS Makine
BEGIN
    PUBLIC String MakineID
    PUBLIC List<Int> BoşlukListesi = new List<Int>()
END CLASS
```

Data, verilerin üretildikten sonra doldurulduğu ve operasyon atamalarının çağırıldığı sınıftır.

```
CLASS Data
BEGIN
    PUBLIC List< Operasyon > Operasyonlar = new List< Operasyon >();
    PUBLIC List< Makine > Makineler = new List<Machine2>();
    PUBLIC List< MakineGrup > MakineGrupları = new List< MakineGrup >();
    PUBLIC METHOD İşAtama()
        BEGIN
            FOR i=1: Operasyonlar
                    OperasyonAtama() metodunu çalıştır
            END FOR
        END
END CLASS
```

Operasyon sınıfında her operasyonun özellikleri tutulur ve OperasyonAtama() metodu ile operasyonların atamaları gerçekleştirilir. Atama algoritmasının temel mantığında, operasyonun atanacağı makinelerden birinde, en erken başlama zamanı kısıtını da sağlayan işlem süresi kadar makinenin boş olduğu bir zaman aralığına yerleştirilmesi söz

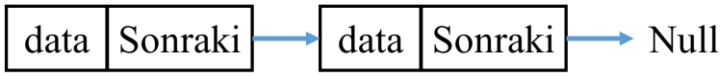
konusudur. Makinelerin boşluğu listelerde “100” değeriyle tutulduğundan “100” değerini işlem süresince aralıksız bulunduran indislere operasyon atanır ve listede ilgili indislerdeki “100” değeri “0”’a çevrilir.

```
CLASS Operasyon
  BEGIN
    PUBLIC String ID;
    PUBLIC String OperasyonID;
    PUBLIC String İşID;
    PUBLIC Double OperasyonSüresi;
    PUBLIC String Makine_GrupID;
    PUBLIC List<String> ÖnşartIDs;
    PUBLIC Double BaşlangıçZamanı;
    PUBLIC Double BitişZamanı;
    PUBLIC String AtananMakine;
    PUBLIC Double EnErkenBaşlamaZamanı;
    PUBLIC METHOD OperasyonAtama()
        BEGIN
            Önşartlarına göre Operasyonun en erken başlayabileceği zaman bulunur
            Operasyonun makine grubundan atanabileceği makineler getirilir
            FOR i=0: Atanabileceği makineler
                Makine i üzerinde, en erken başlama zamanından büyük ve operasyonun işlem süresince boş olduğu bir zaman aralığına operasyon atanır
            END FOR
        END
  END
```

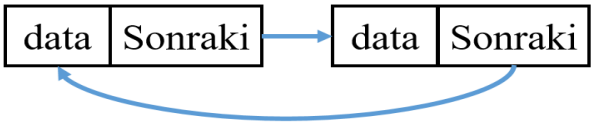
3.3.2. Bağlı Liste Veri Yapısı ile Geliştirilen Algoritma

Çalışmamızda, ileri veri yapısı olarak çift bağlı doğrusal listeyi makine boşluklarını tanımlarken kullanan bir algoritma geliştirdik. Ele aldığımız probleme uygulanabilir ve daha anlaşılır olduğu için liste veri yapısına göre daha komplike olan çift bağlı doğrusal liste yapısı tercih edilmiştir.

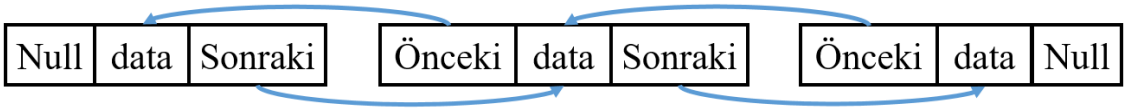
Yapısı itibariyle, listeler için mümkün olan birçok farklı uygulama vardır ve çeşitli ilişkilerle listeler birbirine de bağlanabilir. Listeler, tek bağlı ve çift bağlı olarak bir de doğrusal ve dairesel olarak da gruplandırılabilir. Tek bağlı liste yapısında düğümler bir sonraki düğüme bağlanırken, çift bağlı liste veri yapısında düğümler hem önceki hem sonraki düğümlere bağlanmaktadır. Doğrusal listelerin sonundaki düğümün ileri düğüm bağı bulunmaz (Şekil 3.8, Şekil 3.10). Dairesel liste veri yapısındaysa son düğümün ilk düğüme ileri bağı bulunmaktadır (Şekil 3.9, Şekil 3.11). (Kurutucu, 2014)



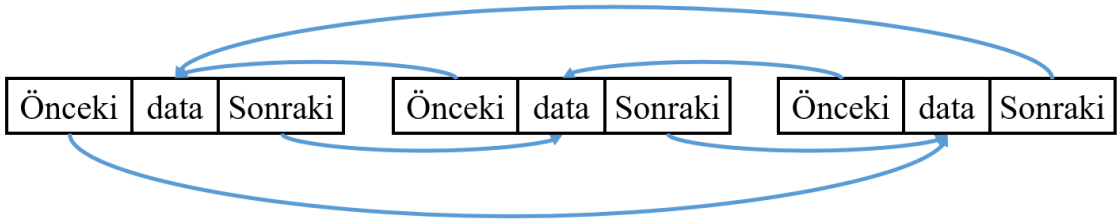
Şekil 3.8. Tek bağlı doğrusal liste yapısı



Şekil 3.9. Tek bağlı dairesel liste yapısı



Şekil 3.10. Çift bağlı doğrusal liste yapısı



Şekil 3.11. Çift bağlı dairesel liste yapısı

Çift bağlı listelerin genel tanımlaması da şu şekildedir:

```
CLASS Düğüm
BEGIN
    String data
    Düğüm öncekidüğüm
    Düğüm sonrakidüğüm
END CLASS
```

Bağlı listeler ile listeleri karşılaştırdığımızda avantaj ve dezavantajlar bulunmaktadır. Bağlı listelerde listelere göre her bir işleme harcanan zaman artmaktadır. Bağlı listeler de liste uygulayan bir listedir fakat bağlı listelerde ilgili düğümü bulmak zamanı arttırmaktadır. İlgili düğüme erişildikten sonra işlemler yine sabit zaman alacaktır. Ancak bağlı listelerin daha dinamik olması da güçlü olmasını sağlayan bir avantajdır (Morin, 2016)

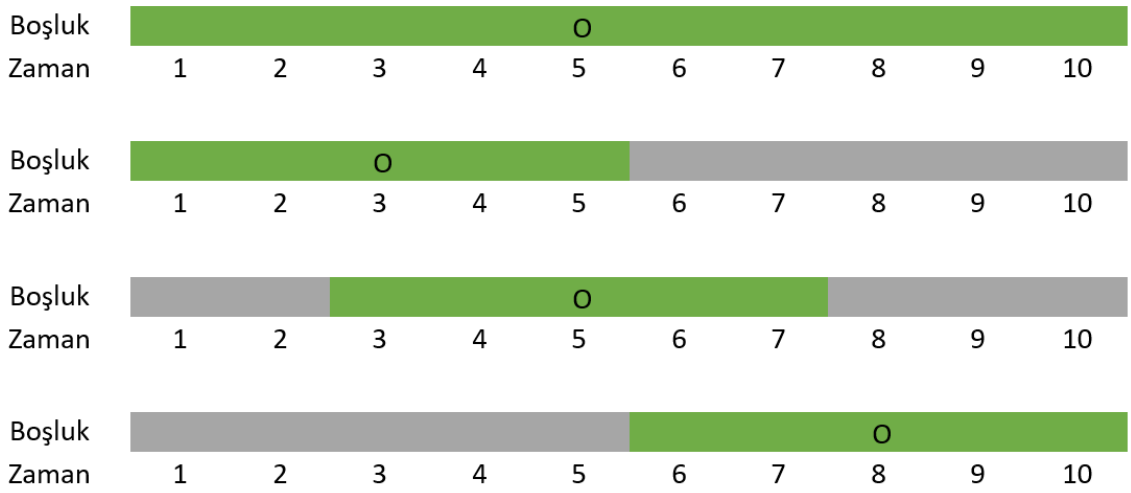
Çift bağlı doğrusal listeler için mantıksal yapı Şekil 3.10'da gösterilmiştir. Her düğümün data adında bir değişkende verileri tutmaktadır. Ayrıca önceki ve sonraki düğümlerin adreslerini tutmakta olan önceki ve sonraki isminde iki referansı bulunmaktadır. Listenin ilk elemanının önceki referansı herhangi bir yeri göstermediğinden değer almamaktadır. İlk düğümün sonraki referansı ise bir sonraki düğümün adres bilgisini içermektedir. İkinci düğümün önceki referansı ilk düğümün adresini tutarken, sonraki referansı ise sonraki düğümün adresini tutmaktadır. Son düğümde ise önceki referans kendinden önceki düğümün adresini tutarken sonraki referansı ise değer almamaktadır.

Programımızda, çifte bağlı doğrusal liste veri yapısıyla makinelerdeki boşluklar aşağıdaki sözde kodda gösterildiği gibi şu şekilde tutulmaktadır:

```
CLASS Boşluk
BEGIN
    PUBLIC Double BoşlukSüresi
    PUBLIC Double BoşlukBaşlangıçZamanı
    PUBLIC Boşluk ÖncekiBoşluk
```

```
PUBLIC Boşluk SonrakiBoşluk
PUBLIC METHOD Atama(Operasyon operasyon)
    BEGIN
        IF Operasyon bu boşluğa sığıyorsa
            IF operasyon.EnErkenBaşlamaZamanı <= BoşlukBaşlangıçZamanı
                IF operasyon.OperasyonSüresi = BoşlukSüresi
                Operasyon boşluğa tam sığar, atama yap
                ELSE
                Operasyon boşluğun başına ata, boşluğun sonunda boşluk kalır
                END IF
            ELSE
                IF Operasyon atanırsa bitecek zaman <= Boşluğun bitiş zamanı
                Operasyonu boşluğun arasında bir yere ata, boşluğun sonunda
                boşluk kalır
                ELSE
                Operasyonu boşluğun sonuna ata
                END IF
            END IF
        END IF
    END
END CLASS
```

Boşluk sınıfında atama yapılırken operasyonun ön şartıyla ilgili başlangıç zamanı ayarlamalarının daha önceden yapıldığı varsayılır. Operasyon, makinedeki bir boşluğa atanırken dört durum söz konusudur. Operasyon boşluğa tam sığabilir. Operasyon, boşluğun başına atanabilir ve boşluğun tamamını kaplamaz. Operasyon, boşluğun başından başlamayıp boşluğun sonuna değmeden bitebilir. Ya da, boşluğun arasında başlayıp boşluğun sonunda bitebilir. Bu durumlar sırasıyla Şekil 3.12'de gösterilmiştir.



Şekil 3.12. Operasyonun boşluğa atandığı durumlar

Operasyon boşluğa tam sığarsa boşluk silinir. Önceki ve sonraki boşlukları düzenlenir. Operasyon boşluğun başına ve sonuna atanırsa boşluğun başlangıç ve süre değerleri güncellenir. Boşluğun ortasında bir yere atandığında ise boşluğa sonraki boşluk düğümü eklenir.

Makinelerin bilgisi, Makine sınıfında tutulmaktadır:

```
CLASS Makine
BEGIN
    PUBLIC String MakineID
    PUBLIC Boşluk MakineBaslangıçBoşluk
END CLASS
```

Makine grupları da MakineGrup sınıfındaki gibi tanımlanmıştır:

```
CLASS MakineGrup
BEGIN
    PUBLIC String MakineGrupID;
    PUBLIC List<String> MakineIDs;
END CLASS
```

Data, verilerin üretildikten sonra doldurulduğu ve her bir operasyonun atamasını çağıran sınıftır:

```
CLASS Data
BEGIN
    PUBLIC List< Operasyon > Operasyonlar = new List< Operasyon >();
    PUBLIC List< Makine > Makineler = new List<Machine2>();
    PUBLIC List< MakineGrup > MakineGrupları = new List< MakineGrup >();
    PUBLIC METHOD İşAtama()
        BEGIN
            FOR i=1: Operasyonlar
                    OperasyonAtama() metodunu çalıştır
            END FOR
        END
END CLASS
```

Operasyon sınıfında her operasyonun özellikleri tutulur ve OperasyonAtama() metodu ile operasyonların atamaları gerçekleştirilir. Burada atama algoritmasının temel mantığında, operasyonun atanacağı makinelerden birinde, makinenin başlangıç boşluğuna bakılır. Eğer bu boşluk operasyonun en erken başlama zamanı kısıtını sağlıyorsa ve boşluk, işlem süresinden büyük veya eşitse operasyon bu boşluğa atanır. Eğer operasyon bu boşluğa atanamıyorsa, ilgili makinede bu boşluktan sonraki boşluğa bakılır. Operasyonun ataması gerçekleşene kadar bu işlem devam ettirilir. Atama gerçekleştiğinde eğer boşluk, operasyon işlem süresinden büyükse, operasyonun boşlukta atandığı yere göre boşluk güncellenir ve gerekirse makine boşluklarına yeni düğümler eklenir. Eğer operasyon boşluğu tamamen kaplıyorsa da ilgili boşluk düğümü silinir, silinen düğümün önceki ve sonraki boşlukları düzenlenir.

```
CLASS Operasyon
  BEGIN
    PUBLIC String ID;
    PUBLIC String OperasyonID;
    PUBLIC String İşID;
    PUBLIC Double OperasyonSüresi;
    PUBLIC String Makine_GrupID;
    PUBLIC List<String> ÖnşartIDs;
    PUBLIC Double BaşlangıçZamanı;
    PUBLIC Double BitişZamanı;
    PUBLIC String AtananMakine;
    PUBLIC Double EnErkenBaşlamaZamanı;
    PUBLIC METHOD OperasyonAtama()
        BEGIN
            Önşartlarına göre Operasyonun en erken başlayabileceği zaman bulunur
            Operasyonun makine grubundan atanabileceği makineler getirilir
            FOR i=0: Atanabileceği makineler
                Makine i üzerinde, en erken başlama zamanından büyük ve
                operasyonun işlem süresince boş olduğu bir zaman aralığına
                operasyon atanır.
                Çift bağlı listeden dolayı şu şekilde atama gerçekleştirilir:
                MevcutBoşluk ← Makine[i]. MakineBaslangıçBoşluk
                WHILE (Makine[i]. MakineBaslangıçBoşluk.SonrakiBoşluk != Null
AND Makine[i].MakineBaslangıçBoşluk.Atama = False)
                MevcutBoşluk ← Makine[i]. MakineBaslangıçBoşluk
                END WHILE
            END FOR
        END
   END
```

Senaryo 1 ve Senaryo 2 için geliştirilen programın kodları aynıdır ve Ek 10'da yer almadır. Senaryo 3 için geliştirilen programın kodları ise Ek 11'de yer almaktadır.

4. BULGULAR ve TARTIŞMA

Bu bölüm iki alt bölümden oluşmaktadır. İlk bölümde, deneylerde kullanılmak için oluşturulan veri setlerinin parametreleri verilmiştir. Diğer bölümde ise algoritmaların ele alınan senaryolar üzerinden elde edilen deney sonuçları yer almaktadır.

4.1. Veri Setinin Oluşturulması

Behnke ve Geiger (2012), çalışmalarında esnek atölye tipi çizelgeleme probleminin farklı spesifikasyonları için literatürdeki yaygın test örneklerini incelemiştir. Literatürdeki veri setlerini kullanmanın, bu tez çalışmasında esnek atölye tipi çizelgeleme problemi çözümüne yönelik değil, veri yapısının algoritma performansına yönelik bir uygulama gerçekleştirildiğinden dolayı sağlıklı sonuç vermeyeceğini düşünülmüştür. Gerçek hayat problemi boyutunda verilerle çalışmak, hızlı ve kolay bir şekilde aynı problem tipinde çeşitli test örneklerini oluşturabilmek için sentetik veri seti kullanmaya karar verilmiştir.

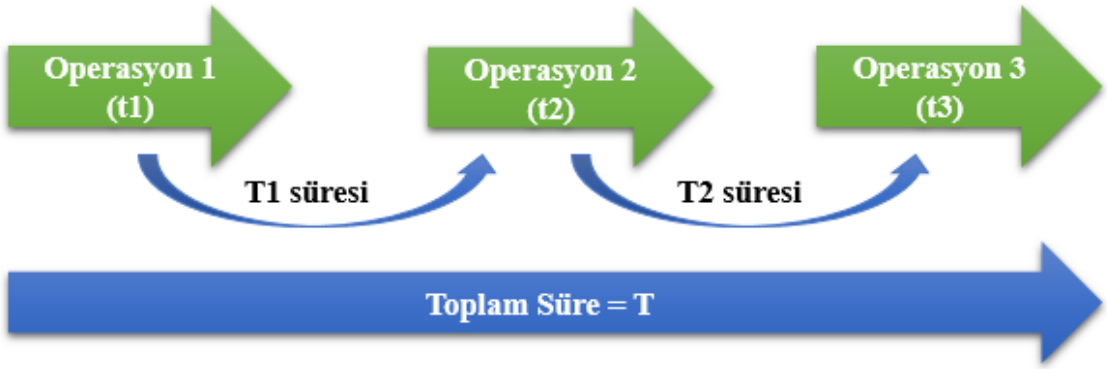
Test örneklerinin parametreleri, hesaplama süresini ve çözüm yöntemlerinin performansını etkiler. Bu yüzden, bu tez çalışmasında rastgele test örnekleri oluştururken, geliştirilen algoritmaların performansını daha iyi anlayabilmek için aşağıdaki gibi mümkün olduğunca çok parametre dikkate alınmıştır:

- Farklı sayıda iş
- Farklı sayıda makine
- Her işin maksimum operasyon sayısı
- Her operasyonun işlem süresinin dağılımı
- Farklı sayıda makine grup sayısı
- Her operasyonun maksimum ön şart sayısı
- Makinelerin esnekliği

Bir problemin boyutu, problemin tanımı ve sürecin akış dizgisinden ileri gelen etmenlere bağlıdır. Problem boyutu; iş sayısı, makine sayısı, maksimum operasyon sayısı ve maksimum grup sayısı ile belirlenmektedir. Problemin hesaplanması için gereken süre, problemin boyutuyla doğru orantılıdır. Bir örneğin iş sayısı ve makine sayısı, problem

boyutunu ve dolayısıyla gerekli hesaplama süresini belirler. Ancak geçen süre yalnızca problemin boyutuyla ilişkili değildir. Test örneklerinde kullandığımız diğer bir önemli parametre, operasyonların işlem süresi dağılımıdır. Operasyonların sonucunda ortaya çıkan işin tamamlanması için geçen süre, operasyonların gerektirdiği farklı işlem sürelerinin toplamı olarak ifade edilebilir. Geçen zaman niteliksel olarak incelendiğinde, bu zamanın operasyonun gerçekleşme ve farklı operasyonların birbirlerine olan etkilerine bağlı olduğu analiz edilmektedir. Yani operasyonun süresiyle beraber diğer operasyonlarla ilişkisi de ayırt edici niteliktedir. Çünkü operasyonların işlem sürelerinin büyük veya küçük olması hesaplamaya ayrılan süreyi etkilediğinden farklı sonuçlar meydana getirebilir.

Yukarıdaki paragrafın daha açıklayıcı olması için verilen Şekil 4.1'de Operasyon1 için geçen zaman; t, Operasyon2 için geçen zaman; t2, Operasyon3 için geçen zaman; t3 olmak üzere T1 ve T2 süreleri; t1, t2 ve t3 sürelerine bağlıdır. t1, t2 ve t3'te yaşanan değişmeler T1 ve T2'yi etkiler. t1, t2, t3, T1, T2 sürelerinde yaşanan değişmeler toplam T süresini etkiler.(Şekil 4.1)



Şekil 4.1. Birbiriyle ilişkisi olan operasyonların sürelerinin etkisi

Her işteki maksimum operasyon sayısı da problem boyutunu etkileyen başka bir parametredir. Gerçekleştirilen her operasyon işlem süreci esnasında işlemsel önceliğe sahip ya da bir başka operasyonun ön koşulu olabileceğinden her bir operasyon öncelik kısıtlaması anlamına gelmektedir. Esnek atölye tipi çizelgeleme problemi için, bir işin daha az operasyonu olduğunda, daha az öncelik kısıtlaması vardır. Öncelik sayısının artması özellikle birden fazla kaynak kullanımını gerektiren senaryo 3'te aynı anda atamak için aramayı zorlaştıracağından çalışma süresini etkiler.

Makinelerin esnekliği de test örneklerinde ele aldığımız problem boyutunu etkileyen diğer bir özelliktir. Eğer bir operasyon bütün makinelerde işlenebiliyorsa tam esneklik, belli makine gruplarında işlenebiliyorsa kısmi esneklik söz konusudur. Biz makine esnekliklerini makine grup sayısı ile sağladık.

Farklı veri yapılarıyla geliştirilen algoritmaları test etmek ve çalışma süresi performanslarını değerlendirebilmek amacıyla her bir senaryo için 270 adet gerçek hayat problemi büyüklüğünde örnekler üretilmiştir. Küçük, orta ve büyük boyutlu problem setleri oluşturulmuştur. Büyük ve orta büyüklükteki problemlerin iş sayısını gerçek hayat problemlerine daha uygun olması için yoğun büyük skalada seçilmiştir. Küçük boyutlu problem için ise iş sayısı literatürdeki gerçek hayat problemlerine de uygun olması için 100 alınmıştır.

Bütün senaryolarda ortak olan parametre değerleri aşağıdaki gibi alınır:

- İş sayısı, j, 100, 5000 veya 10000'e eşittir.
- Makine sayısı, m, 10, 50 veya 100'e eşittir.
- Her bir işteki operasyon sayısı, o, kesikli U(1,3), U(1,6) veya U(1,9) olarak dağılmıştır.
- Operasyonların işlem süreleri 10 ile 90 dakika arasında kesikli düzgün olarak dağılmıştır.

Senaryolara göre farklılaşan parametre değerleri aşağıdaki gibidir:

- Operasyonların ön şart sayısı 1, 2 veya 3'e eşittir.
- Operasyonların kaynak gereksinimi 1 veya 2'ye eşittir.
- Makine grubu sayısı(makine esneklikleri) 1, 2 veya 5'e eşittir.

Farklı veri yapılarıyla geliştirilen algoritmaları test etmek ve çalışma süresi performanslarını değerlendirebilmek amacıyla geçerli bir deney seti oluşturulması gerekmektedir ve bu amaçla İş Sayısı, Makine Sayısı ve Operasyon sayısı parametrelerinin her birinin üçer seviyesi de tam faktöryel tasarıma tabi tutularak 27 noktalı bir tasarım şablonu oluşturulmuştur. Bu şablona göre de her bir senaryo için her bir test örneğinden 10 örnek olmak üzere toplamda 270 adet gerçek hayat problemi

boyutunda örnekler üretilmiştir. Her bir senaryonun test örnek nitelikleri Çizelge 4.1'de verilmiştir. İş sayısı, operasyon sayısı ve makine sayısına göre senaryo test örneklerinin problem boyutları Çizelge 4.2'de yer almaktadır.

Çizelge 4.1. Senaryo test örnekleri için parametreler

SENARYO 1

İş Sayısı	Maksimum Operasyon Sayısı	Operasyon Süresi(dk)	Makine Sayısı	Makine Grup Sayısı
10000	3	U(10,90)	10	1
100	6		50	1
5000	9		100	1

SENARYO 2

İş Sayısı	Maksimum Operasyon Sayısı	Operasyon Süresi(dk)	Makine Sayısı	Makine Grup Sayısı	Maksimum Ön Şart Sayısı
10000	3	U(10,90)	10	2	3
100	6		50	2	
5000	9		100	2	

SENARYO 3

İş Sayısı	Maksimum Operasyon Sayısı	Operasyon Süresi(dk)	Makine Sayısı	Makine Grup Sayısı	Maksimum Ön Şart Sayısı	Maksimum Çoklu Kaynak Sayısı
10000	3	U(10,90)	10	5	3	2
100	6		50	5		
5000	9		100	5		

Çizelge 4.2. Senaryo test örnekleri için problem boyutları

Problem No	İş Sayısı	Maksimum Operasyon Sayısı	Makine Sayısı
P1	100	3	10
P2	100	3	50
P3	100	3	100
P4	100	6	10
P5	100	6	50
P6	100	6	100
P7	100	9	10
P8	100	9	50
P9	100	9	100
P10	5000	3	10
P11	5000	3	50
P12	5000	3	100
P13	5000	6	10
P14	5000	6	50
P15	5000	6	100
P16	5000	9	10
P17	5000	9	50
P18	5000	9	100
P19	10000	3	10
P20	10000	3	50
P21	10000	3	100
P22	10000	6	10
P23	10000	6	50
P24	10000	6	100
P25	10000	9	10
P26	10000	9	50
P27	10000	9	100

4.2. Hesaplama Sonuçları

Çalışmada algoritmalar C#.Net ortamında geliştirilmiş ve üretilen veri setleri kullanılarak çalışma süreleri karşılaştırılmıştır. Deneyler, Intel Core i7 2.59 GHz işlemci ve 8 GB Ram özelliklerine sahip bir bilgisayarda çalıştırılmıştır. Önceki bölümde açıklanan her üç senaryo için liste ve bağlı liste veri yapılarıyla geliştirilen algoritmaların çalışma zamanları bu bölümde birbirleriyle karşılaştırılmış ve alt bölümlerde ayrıntılı olarak sonuçlar verilmiştir. Senaryoların her biri oluşturulan problem setlerindeki 10 sonucun

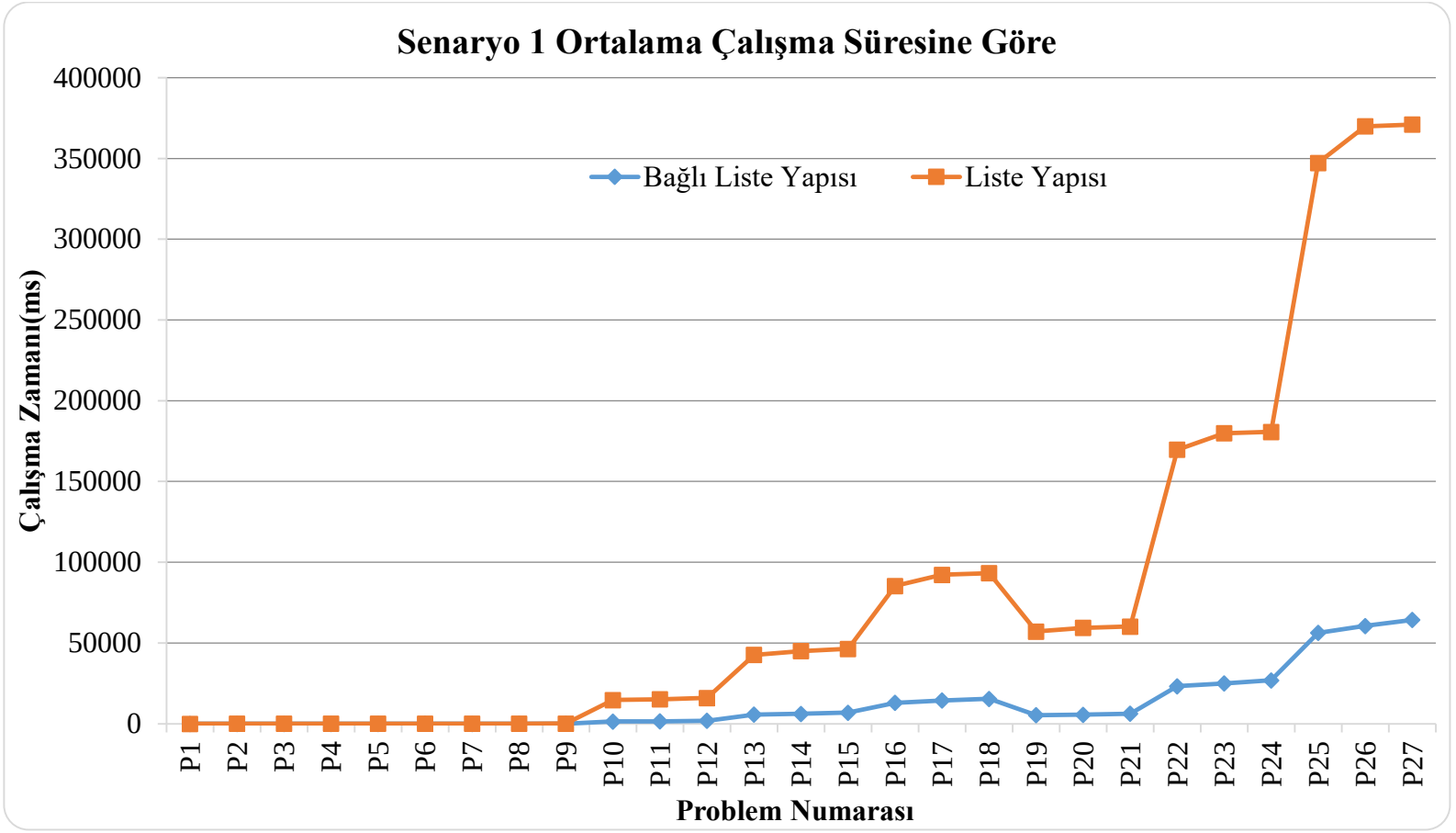
ortalaması olarak hesaplanmıştır. Çalışma zamanlarını tamamı saniye cinsinden verilmiştir.

Senaryo 1 için deney sonuçları Çizelge 4.2'de her bir problem boyutundaki 10 problemin çalışma sürelerinin ortalamasına göre liste ve bağlı liste veri yapısıyla geliştirilen algoritmaların performansı görülmektedir. Ek 1'de yer alan çizelgede Senaryo 1 için liste ve bağlı liste veri yapısıyla geliştirilen algoritmaların performansı daha ayrıntılı incelenebilir. Ek 4'te her bir problem setindeki 10 problemin minimum çalışma süresine göre sonuç grafiği, Ek 7'de ise maksimum çalışma süresine göre sonuç grafiği yer almaktadır.

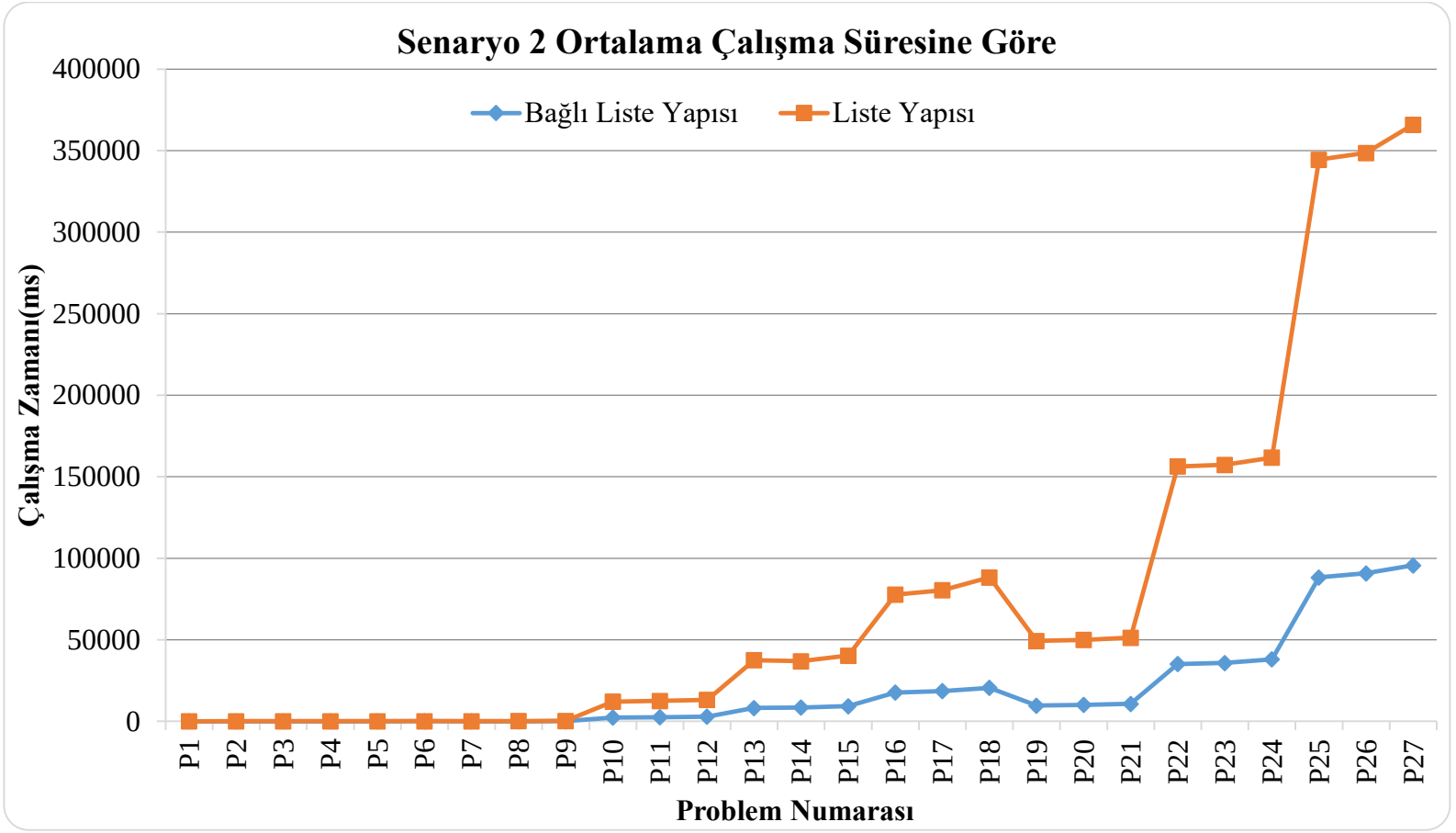
Senaryo 2 için deney sonuçları grafiği Çizelge 4.3'te her bir problem boyutundaki 10 problemin çalışma sürelerinin ortalamasına göre liste ve bağlı liste veri yapısıyla geliştirilen algoritmaların performansı görülmektedir. Ek 2'de yer alan çizelgede senaryo 2 için liste ve bağlı liste veri yapısıyla geliştirilen algoritmaların performansı daha ayrıntılı incelenebilir. Ek 5'te her bir problem setindeki 10 problemin minimum çalışma süresine göre sonuç grafiği, Ek 8'de ise maksimum çalışma süresine göre sonuç grafiği yer almaktadır.

Senaryo 3 için deney sonuçları Çizelge 4.4'te her bir problem boyutundaki 10 problemin çalışma sürelerinin ortalamasına göre liste ve bağlı liste veri yapısıyla geliştirilen algoritmaların performansı görülmektedir. Ek 3'te yer alan çizelgede senaryo 3 için liste ve bağlı liste veri yapısıyla geliştirilen algoritmaların performansı daha ayrıntılı incelenebilir. Ek 6'da her bir problem setindeki 10 problemin minimum çalışma süresine göre sonuç grafiği, Ek 9'da ise maksimum çalışma süresine göre sonuç grafiği yer almaktadır.

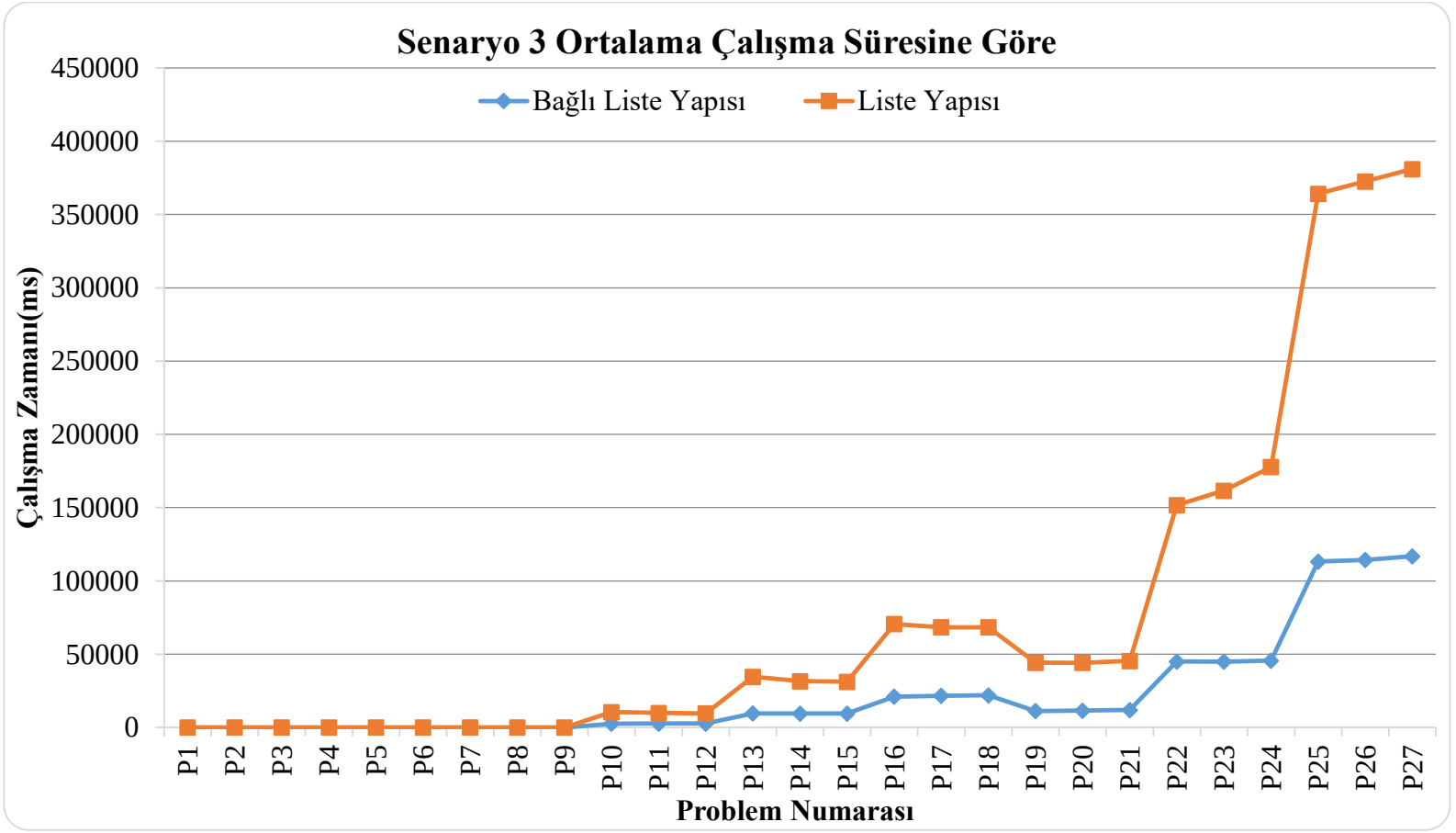
Şekil 4.2. Senaryo 1 deney sonuçlarının ortalama çalışma süresine göre grafiği



Şekil 4.3. Senaryo 2 deney sonuçlarının ortalama çalışma süresine göre grafiği



Şekil 4.4. Senaryo 3 deney sonuçlarının ortalama çalışma süresine göre grafiği



Tüm senaryolar için çalıştırılan 270 deneyin sonuçlarının bibirinden bağımlı örneklem t testi sonuçları Çizelge 4.3'te yer almaktadır. T testi için kurulan H_0 hipotezi, çalışma süresi performansı açısından çift bağlı liste yapısıyla geliştirilen atama algoritmasının, liste veri yapısıyla geliştirilen atama algoritmasından anlamlı düzeyde daha hızlıdır iken, H_1 hipotezi ise eşit veya daha yavaştır şeklinde tanımlanmıştır. Çizelge 4.3'te de görüldüğü gibi elde edilen p değeri 0,01'den küçük olduğu için H0 hipotezi kabul edilmiş ve çift bağlı liste veri yapılı algoritmanın, liste veri yapılı algoritmadan her senaryo için daha hızlı çalıştığı görülmüştür.

H_0: Çift bağlı liste yapısıyla geliştirilen atama algoritmasının çalışma süresi < liste yapısıyla geliştirilen atama algoritmasının çalışma süresi
H_1: Çift bağlı liste yapısıyla geliştirilen atama algoritmasının çalışma süresi ≥ liste yapısıyla geliştirilen atama algoritmasının çalışma süresi

Çizelge 4.3. %99 güvenilirlik seviyesi için bağımlı örneklem T testi sonuçları

	T Değeri	**t Değeri(Tek Kuyruk)**	**P Değeri**
Senaryo 1	12,29	2,34	3,85E-28
Senaryo 2	11,84	2,34	1,3E-26
Senaryo 3	11,03	2,34	6,72E-24

Sonuçlara bakarak her bir örnek niteliğini dikkate alarak etkilerini analiz etmeye çalışırsak: İş, operasyon ve makine sayısı arttıkça tüm senaryolarda çözüm süresi uzamaktadır. Operasyon sayısı arttıkça çözüm süresi daha da uzamaktadır. Çünkü her bir operasyon için makinelerde arama yapılıyor işe ait operasyon sayısı da arttınca bu arama da artacağından süre de çok artmaktadır.

Senaryolardaki farklılıklar nedeniyle iki veri yapısının da çalışma sürelerinde değişmeler olmuştur. Senaryo 1 ve Senaryo 2'de bağlı liste veri yapısının süreleri benzerdir. İki veri yapısı arasındaki çalışma süreleri birbirine yakındır. Fakat, Senaryo 3'te operasyonun çoklu kaynak ihtiyacından dolayı bağlı liste veri yapısının çözüm süresinde değişiklik olmuştur, artmıştır. Liste veri yapısının çözüm süreleri tüm senaryolarda yakın çıkmıştır. Çünkü liste veri yapısında her işlem sabit zaman alırken, bağlı liste veri yapısında düğüm bulmak ciddi zaman harcanmakta daha sonraki işlemler sabit zaman almaktadır.

Senaryo 1 de problem tam esnek olduğu için bütün makinelerde uygun yer arandığından makine sayısı arttıkça her bir problemi setinde çalışma süresinin makine sayısına orantılı arttığı görülmektedir.

Senaryo 2'de kısmi esneklik söz konusu olduğundan tüm makinelerde arama yapılmasına gerek yoktur ama operasyonların ek ön şartı olduğundan en erken başlama zamanları değişmektedir. Dolayısıyla en erken başlama zamanını sağlayacak uygun boşluğu bulmak için için bağlı liste veri yapısında yine düğüme (boşluğa) gidilmesi gerektiğinden bağlı liste veri yapısının çözüm süresi Senaryo 1'e göre biraz artarken, liste veri yapısının süresi Senaryo 1'dekine yaklaşıktır. Ama yine de bağlı liste yapısının çalışma süresinin daha hızlı olduğu görülmektedir.

Senaryo 3'te birden fazla makineye aynı operasyonu aynı zaman aralığına atayabilmek için operasyonun atanması gereken bütün makine gruplarındaki bütün makinelere bakıldığından bağlı liste veri yapılı algoritmanın performansı düşmüş liste veri yapılı algoritmanın performansı artmıştır. Çünkü bağlı liste veri yapısında düğümleri aramak vakit alan masraflı işlemdir. Bağlı veri yapılı algoritmada da operasyonun atanması gereken makine gruplarındaki makinelerde aynı zaman aralığında uygun olan boşluğu(düğüm) bulmak zorundayız. Bu da çalışma hızında yavaşlamaya sebep olmuştur. Fakat buna rağmen liste veri yapılı algoritmadan hala daha hızlı sonuç vermektedir.

Deney sonuçlarına göre, Senaryo 1'de bağlı liste veri yapısı algoritması ile liste veri yapısı algoritmasına göre çalışma süresinde ortalama %81,5 iyileşme elde edilmiştir. Senaryo 2'de bağlı liste veri yapısı algoritması, liste veri yapısı algoritmasına göre ortalama %76,3 daha hızlı çalışmıştır. Senaryo 3'te çalışma süresindeki bu iyileşme oranı ortalama %72,1 olarak kaydedilmiştir.

Bağlı liste veri yapısıyla, 10000 iş maksimum 9 operasyondan ortalama 72000 operasyon olan bir örnek bağlı liste veri yapısıyla yaklaşık 1 dakikada atanırken, liste veri yapısıyla bu süre 6 dakikayı bulmaktadır. Bu çok büyük bir farktır.

200 popülasyon büyüklüğüne sahip 5000 iterasyonlu popülasyon tabanlı yöntemle bir test örneği 1000000 kere çözüldüğü düşünülürse geliştirdiğimiz bağlı liste veri yapısı kullanan algoritmanın çözüm süresini çok daha düşüreceği ön görülmektedir.

İşlem süreleri açısından iki veri yapısını karşılaştırıldığında bağlı liste veri yapısının daha iyi performans gösterdiği deneylerde görülmüştür. Bağlı liste veri yapısıyla geliştirilen algoritmanın çalışma süresi tüm deney sonuçlarında liste ile geliştirilen algoritmaya göre daha kısa sürmüştür.

Çalışmanın sonucunda, amacımıza ulaşarak, çizelgeleme algoritmalarının ileri veri yapılarıyla çözüm sürelerinin iyileştirileceğini söyleyebiliriz.

5. SONUÇ

İşletmeler için günümüzün artan rekabet koşullarında çizelgeleme büyük önem arz etmektedir. Çizelgeleme problemlerinin NP-Zor sınıfta yer alması nedeniyle kesin çözümlerini bulmak zordur ve çözüm süreleri üstsel olarak artmaktadır. Bu nedenle bu problemin çözümüne yönelik sezgisel yöntemler yani algoritmalar giderek önem kazanmıştır. Araştırmacılar çizelgeleme algoritmaları geliştirirken amaç fonksiyonlarını en iyilemeye odaklansalar da yöntemlerinin çözüm süreleri de kritiktir.

Bu sebeplerle bu tez çalışmasında, esnek atölye tipi çizelgeleme problemi için liste ve bağlı liste veri yapılarıyla algoritmalar oluşturulmuş ve ele alınan çizelgeleme problemi çözülmüştür. Uygulama bölümünde deneyler yapılmış ve veri yapılarının çözüm sürelerinin etkisi birbirleriyle karşılaştırılmıştır. Üç farklı senaryo üzerinden performansları değerlendirilmiştir. Veri yapılarının etkisini daha net görebilmek için gerçek hayat problemi büyüklüğünde örneklerle çalışılmak istendiğinden her senaryo için sentetik veriler üretilmiştir. Deney sonuçlarına göre, bağlı liste veri yapısı algoritmasının, liste veri yapısı algoritmasından Senaryo 1'de ortalama %81.5, Senaryo 2'de %76.3 %72.1 oranında iyileşmeyle daha hızlı olduğu gösterilmiştir. Dolayısıyla, bağlı liste veri yapısı tüm senaryolarda atama sonuçlarını çok daha kısa sürede üretmiştir. Sonuçlar istatistik testler ile teyid edilmiştir.

Çalışmanın motivasyonu geliştirilen veri yapısı algoritmasının tüm çizelgeleme algoritmalarıyla da entegre çalışabilir olmasıydı. Bu tez çalışmasında veri yapıları incelendiğinden, amaç fonksiyonunu optimize etmeyi dikkate alınmadan sadece ön şartların sağlanarak işlerin makinelere atandığı uygun bir çözüm oluşturulmuştur. Yapılan bu çalışma literatürdeki esnek atölye tipi çizelgeleme problemlerine yönelik bir alternatif değildir. Tüm çizelgeleme algoritmalarına entegre edilebilir liste ve bağlı liste veri yapılarını kullanarak algoritmaların geliştirildiği ama veri yapılarının süre performanslarının incelendiği bir çalışmadır. Çizelgeleme problemlerinin çözümünde amaç fonksiyonu çok önemli olduğundan dolayı literatürde bu problemin çözümü için çeşitli yöntemler mevcuttur. Literatürdeki çözüm yöntemlerine bakıldığında popülasyon tabanlı geliştirilen algoritmaların çokluğu ve çeşitliliği görülmektedir. Geliştirdiğimiz

bağlı liste veri yapısı popülasyon tabanlı diğer çizelgeleme algoritmalarına da entegre edildiğinde çözüm süresinin çok daha fazla düşeceği düşünülmektedir. Uygulamada her bir örneği birer kez çözerek liste ve bağlı liste veri yapılarının ortaya çıkardığı süre farkının, popülasyon tabanlı bir algoritmada hem popülasyon büyüklüğü hem iterasyon kadar çözüleceği için daha da fark oluşturması beklenmektedir.

Literatürde bilindiği kadarıyla çizelgeleme algoritmalarına veri yapılarının etkisi incelenmemiştir. Bu yüzden çalışmamız önemlidir. Gelecekte de çalışılacağı düşünülerek aşağıdaki konular çalışılabilir:

- Esnek atölye tipi çizelgeleme problemi dışında bir çizelgeleme problemi üzerinde veri yapılarının performansı değerlendirilebilir.
- Entegre edilebilir olduğu için NP-Zor sınıfında yer alan başka bir problem üzerinde benzer liste ve bağlı liste veri yapıları geliştirilebilir.
- Esnek atölye tipi çizelgeleme probleminin sıralama alt problemi dikkate alınarak deneyler yapılabilir.
- Bu veri yapılarıyla popülasyon tabanlı bir meta sezgisel geliştirilip süreleri karşılaştırılabilir.
- Çoklu amaç fonksiyonları dikkate alınarak performansları karşılaştırılabilir.
- Farklı veri yapıları ile karşılaştırılabilir.

KAYNAKLAR

Alvarez-Valdés, R., Fuertes, A., Tamarit, J. M., Giménez, G., ve Ramos, R. (2005). A heuristic to schedule flexible job-shop in a glass factory, *European Journal of Operational Research*, 165(2): 525-534.

Amjad, M. K., Butt, S. I., Kousar, R., Ahmad, R., Agha, M. H., Faping, Z. ve Asgher, U. 2018. Recent research trends in genetic algorithm based flexible job shop scheduling problem, *Mathematical Problems in Engineering*. https://doi.org/10.1155/2018/9270802

Azzouz, A., Ennigrou, M., Said, L. B. 2017. A self-adaptive evolutionary algorithm for solving flexible job-shop problem with sequence dependent setup time and learning effects, *IEEE 2017 Congress On Evolutionary Computation (CEC),* (pp. 1827-1834).

Baker, K. R. ve Trietsch, D. 2009. *Principles of sequencing and scheduling*, John Wiley & Sons.

Baykasoglu, A. 2002. Linguistic-based meta-heuristic optimization model for flexible job shop scheduling, *International Journal of Production Research*, *40*(17): 4523-4543.

Baykasoğlu, A., Madenoğlu, F. S. ve Hamzadayı, A. 2020. Greedy randomized adaptive search for dynamic flexible job-shop scheduling, *Journal of Manufacturing Systems*, *56*(1): 425-451.

Behnke, D. ve Geiger, M. J. 2012. Test instances for the flexible job shop scheduling problem with work centers. Helmut-Schmidt Üniversitesi, Logistics Management Department, Hamburg, Almanya. https://doi.org/10.24405/436

Birgin, E. G., Feofiloff, P., Fernandes, C. G., De Melo, E. L., Oshiro, M. T. Ve Ronconi, D. P. 2014. A MILP model for an extended version of the flexible job shop problem, *Optimization Letters*, *8*(4): 1417-1431.

Biroğul, S., 2005. *Genetik Algoritma Yaklaşımıyla Atölye Çizelgeleme* [Yüksek Lisans Tezi, Gazi Üniversitesi]. Fen Bilimleri Enstitüsü, Ankara.

Brandimarte, P. 1993. Routing and scheduling in a flexible job shop by tabu search, *Annals Of Operations Research*, *41*(3): 157-183.

Bruker, P. ve Schlie, R. 1990. Job-shop scheduling with multi-purpose machines, *Computing*, *45*(4): 369-375.

Chang, H. C., Chen, Y.-P. ve Liu, T. 2015. Solving the flexible job shop scheduling problem with makespan optimization by using a hybrid Taguchi genetic algorithm, IEEE access *3*(2015): 1740-1754.

Chaudhry, I. A., ve Khan, A. A. 2016. A research survey: review of flexible job shop scheduling techniques, *International Transactions in Operational Research*, *23*(3): 551-591.

Chen, J. C., Wu, C. C., Chen, C. W. ve Chen, K. H. 2012. Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm, *Expert Systems with Applications, 39*(11): 10016-10021.

Çınar, D., Oliveira, J. A., Topçu, Y. I. ve Pardalos, P. M. 2016. A priority-based genetic algorithm for a flexible job shop scheduling problem, *Journal of Industrial & Management Optimization, 12*(4): 1391-1415.

Çınar, D., Topçu, Y. I. ve Oliveira, J. A. 2015. A taxonomy for the flexible job shop scheduling problem. *In Optimization, Control, and Applications in the Information Age, Vol. 130.* (pp. 17-37), Springer, Cham.

Defersha, F. M., ve Chen, M. 2010. A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups, *The International Journal Of Advanced Manufacturing Technology, 49*(1): 263-279.

Defersha, F. M. ve Rooyani, D. 2020. An efficient two-stage genetic algorithm for a flexible job-shop scheduling problem with sequence dependent attached/detached setup, machine release date and lag-time, *Computers & Industrial Engineering, Vol. 147.* (106605).

Esen, Z. F. (2020). *Sudoku Bulmacasının Kuyruk Liste Veri Yapısı Tabanlı Paralel Önce-derine Arama Yöntemiyle Çözülmesi* (Tez No. 535422) [Yüksek Lisans Tezi, Anadolu Üniversitesi]. YÖK Tez Merkezi.

Gao, K., Cao, Z., Zhang, L., Chen, Z., Han, Y. ve Pan, Q. 2019. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems, *IEEE/CAA Journal of Automatica Sinica, 6*(4): 904-916.

Garey, M. R., Johnson, D. S. ve Sethi, R. 1976. The complexity of flowshop and jobshop scheduling, *Mathematics of Operations Research, 1*(2): 117-129.

Gomes, M.C., Barbosa-Póvoa, A.P., Novais, A.Q. 2013. Reactive scheduling in a make-to-order flexible job shop with re-entrant process and assembly: a mathematical programming approach', *International Journal of Production Research, 51*(17): 5120-5141.

Graham, R. L., Lawler, E. L., Lenstra, J. K. ve Kan, A. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics, Vol. 5.* (pp. 287–326).

Grobler, J., Engelbrecht, A. P., Kok, S., ve Yadavalli, S. 2010. Metaheuristics for the multiobjective FJSP with sequence-dependent set-up times, auxiliary resources and machine down time, *Annals of Operations Research, 180*(1): 165–196.

Gu, X., Huang, M. ve Liang, X. 2017. The improved simulated annealing genetic algorithm for flexible job-shop scheduling problem, *IEEE 2017 6th International Conference on Computer Science and Network Technology (ICCSNT)*, (pp. 22–27).

Hansmann, R. S., Rieger, T. ve Zimmermann, U. T. (2014). Flexible job shop scheduling with blockages, *Mathematical Methods of Operations Research, 79*(2): 135-161.

Huang, M., Mingxu, W. ve Xu, L. 2016a. An improved genetic algorithm using opposition-based learning for flexible job-shop scheduling problem, *2016 2nd International Conference on Cloud Computing and Internet of Things (CCIOT),* (pp. 8–15).

Huang, M., Wang, L.-M. ve Liang, X. 2016b. An improved adaptive genetic algorithm in flexible job shop scheduling, *2016 2nd International Conference on Cloud Computing and Internet of Things (CCIOT),* (pp. 177–184).

Hosseinabadi, A. A. R., Siar, H., Shamshirband, S., Shojafar, M. ve Nasir, M. H. N. M. 2015. Using the gravitational emulation local search algorithm to solve the multi-objective flexible dynamic job shop scheduling problem in Small and Medium Enterprises, *Annals of Operations Research, 229*(1): 451-474.

İnner, B. 2004. *Hiyerarşik N-cisim algoritmalarının performans karşılaştırılması ve veri yapılarının incelenmesi* (Tez No. 150217) [Yüksek Lisans Tezi, Gebze Yüksek Teknoloji Enstitüsü]. YÖK Tez Merkezi.

Kacem, I., Hammadi, S. ve Borne, P. 2002. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems, *IEEE Transactions on Systems, Man, and Cybernetics, Part C, 32*(1): 1–13.

Kaplanoğlu, V. 2016. An object-oriented approach for multi-objective flexible job-shop scheduling problem, *Expert Systems with Applications, Vol. 45*. (pp. 71-84).

Kaya, S. ve Fığlalı, N. 2016. Esnek atölye tipi çizelgeleme problemlerinin meta sezgisel yöntemler ile çözümüne yönelik bir inceleme, *Sakarya University Journal of Science, 20*(2): 223-244.

Kaya, S. ve Fığlalı, N. 2018. Çok amaçlı esnek atölye tipi çizelgeleme problemlerinin çözümünde meta sezgisel yöntemlerin kullanımı, *Harran Üniversitesi Mühendislik Dergisi, 3*(3): 222-233.

Kellegöz, T. 2006. *Toplam geç bitirme zamanının en küçüklenmesi performans ölçütlü permütasyon akış tipi çizelgeleme problemlerinin çözümünde genetik algoritma yaklaşımı* [Yüksek Lisans Tezi, Kırıkkale Üniversitesi]. Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, Kırıkkale.

Kurutucu, H. 2014. *Veri Yapıları*. Karabük Üniversitesi, Mühendislik Fakültesi, Ders Notları, Karabük, 92 s.

Leung, J. Y. 2004. *Handbook of scheduling: algorithms, models, and performance analysis*, Chapmann & Hall/CRC Press.

Li, L. ve Huo, J. Z. 2009. Multi-objective flexible job-shop scheduling problem in steel tubes production, *Systems Engineering-Theory & Practice, 29*(8): 117-126.

Li, X. ve Gao, L. 2016. An effective hybrid genetic algorithm and TS for flexible job shop scheduling problem, *International Journal of Production Economics, Vol. 174*. (pp. 93–110).

Liu, H., Abraham, A., Wang, Z. 2009. A multi-swarm approach to multi-objective flexible Job-shop scheduling problems, *Fundamental Informatics, Vol. 95*. (pp. 465-489).

Marzouki, B. ve Driss, O.B. 2015. Multi agent model based on chemical reaction optimization for flexible job shop problem, *IEEE 7th International Conference on Computational Collective Intelligence (ICCCI)*, (pp. 29–38).

Marzouki, B. ve Driss, O.B., Ghédira, K. 2018. Multi-agent model based on combination of chemical reaction optimisation metaheuristic with TS for flexible job shop scheduling problem, *International Journal of Intelligent Engineering Informatics, 6*(3-4): 242-265.

Mati, Y., Rezg, N. ve Xie, X. 2001. An integrated greedy heuristic for a flexible job shop scheduling problem, *IEEE International Conference on Systems, Man and Cybernetics e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236), Vol. 4*. (pp. 2534–2539).

Morin, P. 2016. *Açık Veri Yapıları (Java ile)*, lulu.com

Pezzella, G.M., ve Ciaschetti, G. 2008. A genetic algorithm for the flexible job-shop scheduling problem, *Computers & Operations Research, 35*(10): 3202-3212.

Pinedo, M. 2012. *Scheduling theory, algorithms, and systems*, Springer.

Polatlı, E. 2015. *Hücresel İmalatta Çizelgeleme* [Yüksek Lisans Tezi, Sakarya Üniversitesi]. Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, Sakarya.

Purnomo, M.R.A. 2016. A knowledge-based genetic algorithm for solving flexible job shop scheduling problem, *International Business Management*, *10*(19): 4708–4712

Reeves, C.R., 1995. *Modern Heuristic Techniques for Combinatorial Problems*, McGraw-Hill Book Company.

Ren, H., Xu, H. ve Sun, S. 2016. Immune genetic algorithm for multi-objective flexible job-shop scheduling problem, *IEEE 2016 Chinese control and decision conference (CCDC)*, (pp. 2167–2171).

Romero, M. A. F., García, E. A. R., Ponsich, A. ve Gutiérrez, R. A. M. 2018. A heuristic algorithm based on tabu search for the solution of flexible job shop scheduling problems with lot streaming, *In Proceedings of the Genetic and Evolutionary Computation Conference*, (pp. 285-292).

Storer, J. A. (2012). *An introduction to data structures and algorithms*, Springer Science & Business Media.

Sirkeci, E. 2015. *Esnek Atölye Tipi Çizelgeleme Problemi İçin Çözüm Yaklaşımları: Savunma Sanayinde Bir Uygulama* [Yüksek Lisans Tezi, Gazi Üniversitesi]. Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, Ankara.

Sun, J., Zhang, G., Lu, J. ve Zhang, W. (2021). A hybrid many-objective evolutionary algorithm for flexible job-shop scheduling problem with transportation and setup times, *Computers & Operations Research, Vol. 132*. https://doi.org/10.1016/j.cor.2021.105263

Tamssaouet, K., Dauzère-Pérès, S., Knopp, S., Bitar, A. ve Yugma, C. 2021. Multiobjective Optimization for Complex Flexible Job-Shop Scheduling Problems. *European Journal of Operational Research*, *296*(1): 87-100.

Tanev, I. T., Uozumi, T. ve Morotome, Y. 2004 . Hybrid evolutionary algorithm-based real-world flexible job shop scheduling problem: application service provider approach, *Applied soft computing*, *5*(1): 87-100.

Tanyıldızı, E. 2012. *Veri Yapıları*. Fırat Üniversitesi, Teknoloji Fakültesi, Ders Notları Elazığ.

Türkyılmaz, A., Şenvar, Ö., Ünal, İ. ve Bulkan, S. 2020. A research survey: heuristic approaches for solving multi objective flexible job shop problems. *Journal of Intelligent Manufacturing, 31*(8): 1-35.

Cedimoğlu, İ.H. 2008. *Veri Yapıları ve Programlama*. Sakarya Üniversitesi, Adapazarı Meslek Yüksekokulu, Ders Notları, Sakarya, 7 s.

Wang, L., Cai, J., Li, M. ve Liu, Z. 2017. Flexible job shop scheduling problem using an improved ant colony optimization, *Scientific Programming, Vol. 2017*. 9016303, (pp. 1-11).

Xie, J., Gao, L., Peng, K., Li, X. ve Li, H. 2019. Review on flexible job shop scheduling, *IET Collaborative Intelligent Manufacturing, 1*(3): 67-77.

Yi, W., Li, X. ve Pan, B. 2016. Solving flexible job shop scheduling using an effective memetic algorithm, *International Journal of Computer Applications in Technology, 53*(2): 157-163

Yuan, Y., ve Xu, H. 2013. An integrated search heuristic for large-scale flexible job shop scheduling problems, *Computers & Operations Research, 40*(12): 2864-2877.

Yuan, Y. ve Xu, H. 2015. Multi objective flexible job shop scheduling using memetic algorithms, *IEEE Transactions on Automation Science and Engineering, 12*(1): 336-353.

Yurttakal, A. H. 2014. *İş Akışı Çizelgeleme Probleminin Yapay Bağışıklık Sistemi ile Optimizasyonu* (Tez No. 380918) [Yüksek Lisans Tezi, Selçuk Üniversitesi]. YÖK Tez Merkezi.

Zhang, G., Shao, X., Li, P. ve Gao L. 2009. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, *Computers & Industrial Engineering*, *56*(4): 1309-1318.

Zhang, Z., Wu, L., Peng, T., ve Jia, S. 2019. An improved scheduling approach for minimizing total energy consumption and makespan in a flexible job shop environment, *Sustainability*, *11*(1): 179.

Zhang, G., Zhang, L., Song, X., Wang, Y. ve Zhou, C. 2019. A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem, *Cluster Computing*, *22*(5): 11561-11572.

Zhang, G., Hu, Y., Sun, J. ve Zhang, W. 2020. An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints, *Swarm and Evolutionary Computation*, *Vol. 54*. https://doi.org/10.1016/j.swevo.2020.100664

EKLER

EK 1 Senaryo 1 için elde edilen deney sonuçları

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
1	100	3	10	4,02	12,68
2	100	3	10	4,19	12,1
3	100	3	10	5,67	15,1
4	100	3	10	4,27	14,02
5	100	3	10	5,78	21,36
6	100	3	10	4,3	13,62
7	100	3	10	4,26	13,28
8	100	3	10	4,17	14,94
9	100	3	10	4,37	13,42
10	100	3	10	4,22	12,69
11	100	3	50	5,95	16,55
12	100	3	50	6,75	18,43
13	100	3	50	6,05	18,6
14	100	3	50	6,05	16,68
15	100	3	50	4,66	12,17
16	100	3	50	6,52	19,22
17	100	3	50	6,02	17,5
18	100	3	50	6,14	16,91
19	100	3	50	5,75	17,09
20	100	3	50	6,97	17,65
21	100	3	100	12,2	30,58
22	100	3	100	12,07	30,75
23	100	3	100	12,32	31,34
24	100	3	100	13,15	31,29
25	100	3	100	13,54	32,63
26	100	3	100	12,13	30,03
27	100	3	100	12,38	30,99
28	100	3	100	11,84	29,51
29	100	3	100	12,17	31,3
30	100	3	100	12,81	32,41

EK 1'in devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
31	100	6	10	6,07	28,64
32	100	6	10	6,11	30,66
33	100	6	10	6,65	27,2
34	100	6	10	6,31	27,61
35	100	6	10	6,31	27,89
36	100	6	10	6,22	25,75
37	100	6	10	6,33	29,31
38	100	6	10	9,4	29,75
39	100	6	10	7,21	27,31
40	100	6	10	6,76	28,96
41	100	6	50	10,83	42,22
42	100	6	50	8,97	33,89
43	100	6	50	9,35	34,29
44	100	6	50	10,81	41,95
45	100	6	50	10,35	38,31
46	100	6	50	10,13	41,27
47	100	6	50	9,5	35,08
48	100	6	50	9,35	36,81
49	100	6	50	9,61	34,49
50	100	6	50	9,19	32,38
51	100	6	100	19,4	63,02
52	100	6	100	20,55	68,04
53	100	6	100	20,8	70,74
54	100	6	100	22,03	69,28
55	100	6	100	21,19	73,01
56	100	6	100	19,37	59,56
57	100	6	100	19,68	59,27
58	100	6	100	19,5	63,77
59	100	6	100	22,03	69,73
60	100	6	100	19,1	61,33

EK 1'in devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
61	100	9	10	12	52,19
62	100	9	10	10,91	61,18
63	100	9	10	10,42	53,87
64	100	9	10	10,39	53,18
65	100	9	10	8,91	46,01
66	100	9	10	9,71	55,52
67	100	9	10	9,8	47,86
68	100	9	10	12,55	66,69
69	100	9	10	9,52	47,05
70	100	9	10	9,65	46,12
71	100	9	50	14,73	69,4
72	100	9	50	17,13	76,66
73	100	9	50	14,32	67,69
74	100	9	50	15,03	71,57
75	100	9	50	15,85	76,21
76	100	9	50	16,76	78,03
77	100	9	50	15,28	69,58
78	100	9	50	15,57	71,4
79	100	9	50	14,53	66,48
80	100	9	50	16,71	79,54
81	100	9	100	29,61	122,04
82	100	9	100	27,45	105,13
83	100	9	100	32,31	127,41
84	100	9	100	29,65	112,07
85	100	9	100	32,4	119,99
86	100	9	100	32,08	125,34
87	100	9	100	29,14	116,39
88	100	9	100	33,16	120,57
89	100	9	100	32,41	130,01
90	100	9	100	28,09	108,59

EK 1'in devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
91	5000	3	10	1403,24	14579,79
92	5000	3	10	1478,42	14456,42
93	5000	3	10	1434,43	14884,82
94	5000	3	10	1451,59	14600,52
95	5000	3	10	1505,94	14849,29
96	5000	3	10	1462,91	14505,45
97	5000	3	10	1400,54	14572,46
98	5000	3	10	1633,36	15213,65
99	5000	3	10	1424,33	14773,38
100	5000	3	10	1460,5	14783,58
101	5000	3	50	1531,39	15264,04
102	5000	3	50	1578,81	15330,46
103	5000	3	50	1531,34	15295,61
104	5000	3	50	1490,5	14880,41
105	5000	3	50	1598,03	15376,7
106	5000	3	50	1494,36	15069,89
107	5000	3	50	1537,84	15392,48
108	5000	3	50	1533,87	15133,5
109	5000	3	50	1493,3	14847,68
110	5000	3	50	1555,86	15468,83
111	5000	3	100	1862,26	15651,71
112	5000	3	100	1864,17	15894,49
113	5000	3	100	1869,66	15898,69
114	5000	3	100	1848,36	15873,83
115	5000	3	100	1880,54	16034,1
116	5000	3	100	1829,67	15642,29
117	5000	3	100	1883,68	16281,14
118	5000	3	100	1921,92	16312,2
119	5000	3	100	1876,43	16016,07
120	5000	3	100	1881,72	15920,28

EK 1'in devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
121	5000	6	10	6100,19	43908,49
122	5000	6	10	5767,14	41838,56
123	5000	6	10	5470,17	42689,85
124	5000	6	10	5772,31	43305,43
125	5000	6	10	5876,36	42733,12
126	5000	6	10	5597,4	42290,26
127	5000	6	10	5747,79	42629,06
128	5000	6	10	5633,78	42165,34
129	5000	6	10	5776,84	42244,47
130	5000	6	10	5682,07	42948,54
131	5000	6	50	6288,09	44644,07
132	5000	6	50	6101,31	44498,37
133	5000	6	50	6148,58	45249,9
134	5000	6	50	6018,9	44456,16
135	5000	6	50	6168,92	45170,29
136	5000	6	50	6196,98	45856,42
137	5000	6	50	6466,02	45705,37
138	5000	6	50	6197,54	45916,74
139	5000	6	50	5964,68	44161,24
140	5000	6	50	6244,44	44347,52
141	5000	6	100	6693,77	45424,1
142	5000	6	100	6812,56	45660,29
143	5000	6	100	7045,06	47197,86
144	5000	6	100	6862,77	46401,29
145	5000	6	100	7088,52	48273,56
146	5000	6	100	6610,88	45235,37
147	5000	6	100	6925,45	46393,26
148	5000	6	100	6712,7	45971,7
149	5000	6	100	6954,49	46761,63
150	5000	6	100	6857,2	46396,93

EK 1'in devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
151	5000	9	10	12760,54	84130,04
152	5000	9	10	13105,34	85133,1
153	5000	9	10	13028	85393,78
154	5000	9	10	13167,47	85699,44
155	5000	9	10	12757,89	85041,52
156	5000	9	10	12982,44	86212,09
157	5000	9	10	12640,92	83855,02
158	5000	9	10	13197,98	86285,74
159	5000	9	10	12762,98	84484,25
160	5000	9	10	13121,89	86706,08
161	5000	9	50	14108,51	91015,22
162	5000	9	50	15151,81	95209,75
163	5000	9	50	13992,16	90859,82
164	5000	9	50	14303,66	90042,09
165	5000	9	50	15345,15	96396,56
166	5000	9	50	14472,98	92810,79
167	5000	9	50	14368,32	92432,37
168	5000	9	50	14078,47	90155,53
169	5000	9	50	14584,85	93194,38
170	5000	9	50	14151,51	90503,73
171	5000	9	100	15494,38	94182,22
172	5000	9	100	15994,18	94588,09
173	5000	9	100	15077,06	91696,05
174	5000	9	100	14933,96	91180,17
175	5000	9	100	15096,78	92228,43
176	5000	9	100	15298,81	93140,27
177	5000	9	100	16061,55	95175,65
178	5000	9	100	15651,9	94830,38
179	5000	9	100	15008,78	90971,83
180	5000	9	100	15564,92	94140,52

EK 1'in devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
181	10000	3	10	5365,24	57823
182	10000	3	10	5240,26	57230,29
183	10000	3	10	5311,95	56951,19
184	10000	3	10	5263,4	56464,57
185	10000	3	10	5287,12	57566,15
186	10000	3	10	5151,62	56495,87
187	10000	3	10	5323,56	57730,77
188	10000	3	10	5088,48	55953,45
189	10000	3	10	5463,68	57401,63
190	10000	3	10	5477,97	57450,8
191	10000	3	50	5667,59	59104,88
192	10000	3	50	5552,32	58850,9
193	10000	3	50	5705,14	59918,89
194	10000	3	50	5533,23	58326,21
195	10000	3	50	5524,72	59526,98
196	10000	3	50	5586,4	59163,42
197	10000	3	50	5763,82	60134,18
198	10000	3	50	5721,43	60168,74
199	10000	3	50	5531,36	59282,23
200	10000	3	50	5498,82	59186,96
201	10000	3	100	6311,36	61335,64
202	10000	3	100	6146,25	59334,6
203	10000	3	100	6218,11	59927,14
204	10000	3	100	6252,09	59199,16
205	10000	3	100	6202,06	59732,42
206	10000	3	100	6227,7	60567,34
207	10000	3	100	6173,62	59337,84
208	10000	3	100	6562,46	61117,04
209	10000	3	100	6275,72	60732
210	10000	3	100	6246,09	60587,16

EK 1'in devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
211	10000	6	10	24129,98	172519,14
212	10000	6	10	23052,71	170403,14
213	10000	6	10	23543,7	168942,78
214	10000	6	10	23755,31	172320,21
215	10000	6	10	23486,31	170279,27
216	10000	6	10	23594,52	170575,53
217	10000	6	10	22950,11	167336,85
218	10000	6	10	22562,18	165605,22
219	10000	6	10	22375,16	167343,03
220	10000	6	10	22988,18	171545,29
221	10000	6	50	25523,38	180487,68
222	10000	6	50	24509,19	178796,43
223	10000	6	50	25241,83	180879,32
224	10000	6	50	25099,6	181477,71
225	10000	6	50	24642,43	179123,04
226	10000	6	50	24922,7	178874,23
227	10000	6	50	24578,8	178420,89
228	10000	6	50	25559,99	182204,5
229	10000	6	50	24622,67	179437,95
230	10000	6	50	24944,82	179437,04
231	10000	6	100	27609,7	180600,08
232	10000	6	100	27280,25	181481,16
233	10000	6	100	26415,97	180131,26
234	10000	6	100	26755,93	181157,14
235	10000	6	100	26229,54	179366,66
236	10000	6	100	26969,14	184639,45
237	10000	6	100	25841,97	177715,51
238	10000	6	100	29380,46	185261,29
239	10000	6	100	26302,86	178265,05
240	10000	6	100	26164,77	178745,7

EK 1'in devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
241	10000	9	10	57300,01	352659,3
242	10000	9	10	56816,53	346465,73
243	10000	9	10	57090,46	348302,91
244	10000	9	10	56693,17	343529,32
245	10000	9	10	54916,48	345478,85
246	10000	9	10	55088,41	343725,07
247	10000	9	10	56894,66	347559,22
248	10000	9	10	55526,77	342907,93
249	10000	9	10	55186,13	342144,05
250	10000	9	10	57561,84	359917,77
251	10000	9	50	59115,34	364508,02
252	10000	9	50	58508,08	360365,11
253	10000	9	50	59818,71	363606,89
254	10000	9	50	60604,23	369431,74
255	10000	9	50	61361,07	369083,72
256	10000	9	50	61070,91	366572,44
257	10000	9	50	62435,37	376974,86
258	10000	9	50	59939,07	371105,15
259	10000	9	50	62253,56	383649,15
260	10000	9	50	60747,61	373391,63
261	10000	9	100	63561,2	369759,66
262	10000	9	100	62116,21	363935,85
263	10000	9	100	64718,51	373294,49
264	10000	9	100	64083,49	371241,23
265	10000	9	100	63825,13	367825,53
266	10000	9	100	64315,53	370525,12
267	10000	9	100	64491,58	370913,44
268	10000	9	100	62696,27	368503,09
269	10000	9	100	64640,18	374220,83
270	10000	9	100	68231,78	379509,23

EK 2 Senaryo 2 için elde edilen deney sonuçları

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
1	100	3	10	4,97	11,21
2	100	3	10	4,84	11,09
3	100	3	10	4,61	11,5
4	100	3	10	4,53	11,98
5	100	3	10	5,41	11,66
6	100	3	10	7,97	15,02
7	100	3	10	5,18	12,14
8	100	3	10	4,62	11,61
9	100	3	10	5,06	14,99
10	100	3	10	5,68	12,23
11	100	3	50	6,58	19,82
12	100	3	50	6,41	20,69
13	100	3	50	6,88	20,82
14	100	3	50	6,73	19,64
15	100	3	50	6,77	20,28
16	100	3	50	7,21	26,27
17	100	3	50	6,47	23,09
18	100	3	50	6,41	22,01
19	100	3	50	6,59	24,79
20	100	3	50	6,15	18,45
21	100	3	100	12,05	35,76
22	100	3	100	12,05	44,24
23	100	3	100	12,5	40,77
24	100	3	100	12,03	37,85
25	100	3	100	12,26	37,9
26	100	3	100	12,49	39,24
27	100	3	100	12,14	36,58
28	100	3	100	12,33	41,87
29	100	3	100	12,06	41,75
30	100	3	100	13,54	42,52

EK 2'nin devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
31	100	6	10	7,13	27
32	100	6	10	7,28	23,03
33	100	6	10	7,52	23,16
34	100	6	10	8,52	29,17
35	100	6	10	7,34	24,81
36	100	6	10	12,21	27,13
37	100	6	10	7,63	25,17
38	100	6	10	9,49	23,92
39	100	6	10	10,41	29
40	100	6	10	7,85	33,69
41	100	6	50	11,86	48,38
42	100	6	50	9,56	42,77
43	100	6	50	10,97	51,95
44	100	6	50	11,85	59,9
45	100	6	50	10,61	46,19
46	100	6	50	11,59	61,7
47	100	6	50	9,98	39,82
48	100	6	50	10,33	54,65
49	100	6	50	10,59	48,43
50	100	6	50	9,98	45,41
51	100	6	100	20,56	99,24
52	100	6	100	21,29	115,76
53	100	6	100	22,25	120,12
54	100	6	100	21,22	151,3
55	100	6	100	21,76	87,88
56	100	6	100	18,84	105,74
57	100	6	100	19,85	100,51
58	100	6	100	20,47	82,51
59	100	6	100	22,11	104,6
60	100	6	100	20,68	73,98

EK 2'nin devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
61	100	9	10	15,03	60,87
62	100	9	10	12,75	58,88
63	100	9	10	13,2	60,74
64	100	9	10	12,79	49,64
65	100	9	10	10,8	38,34
66	100	9	10	12,53	46,37
67	100	9	10	11,69	44,64
68	100	9	10	13,65	51,67
69	100	9	10	11,74	48,31
70	100	9	10	15,5	54,1
71	100	9	50	17,43	101,84
72	100	9	50	17,6	134,79
73	100	9	50	15,87	101,23
74	100	9	50	16,96	118,42
75	100	9	50	18,79	140,37
76	100	9	50	19,14	141,78
77	100	9	50	17,01	97,91
78	100	9	50	15,69	110,87
79	100	9	50	16,68	120,49
80	100	9	50	18,57	135,83
81	100	9	100	29,93	222,33
82	100	9	100	27,75	145,45
83	100	9	100	32,69	259,05
84	100	9	100	29,9	246,14
85	100	9	100	33,34	210,53
86	100	9	100	32,48	187,08
87	100	9	100	31,96	239,29
88	100	9	100	38,18	238,47
89	100	9	100	34,68	264,46
90	100	9	100	29,23	203,42

EK 2'nin devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
91	5000	3	10	2284,05	11943,17
92	5000	3	10	2301,31	11985,63
93	5000	3	10	2340,61	12099,9
94	5000	3	10	2447,02	12150,31
95	5000	3	10	2318,05	11982,24
96	5000	3	10	2297,63	11857,18
97	5000	3	10	2320,9	11946,4
98	5000	3	10	2309,35	12229,12
99	5000	3	10	2382,79	12283,86
100	5000	3	10	2391,25	12313,24
101	5000	3	50	2514,1	13614,57
102	5000	3	50	2544,51	12362,46
103	5000	3	50	2501,16	12248,55
104	5000	3	50	2487,76	12249,62
105	5000	3	50	2603,65	12650,63
106	5000	3	50	2443,13	12283,81
107	5000	3	50	2542,71	12430,22
108	5000	3	50	2479,51	12385,16
109	5000	3	50	2434,92	12134,1
110	5000	3	50	2463,85	12460,1
111	5000	3	100	3069,75	13333
112	5000	3	100	2762,35	12872,16
113	5000	3	100	2782,43	12932,49
114	5000	3	100	2808,97	13073,17
115	5000	3	100	2832,57	13387,31
116	5000	3	100	2709,05	12636,43
117	5000	3	100	2915,29	13583
118	5000	3	100	2847,86	13473,93
119	5000	3	100	2792,31	13026,39
120	5000	3	100	2930,25	13250,18

EK 2'nin devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
121	5000	6	10	8567,67	38342,47
122	5000	6	10	8024,53	37027,09
123	5000	6	10	8223,46	37717,94
124	5000	6	10	8364,61	38231,21
125	5000	6	10	8096,8	36996,97
126	5000	6	10	8108,23	37232,51
127	5000	6	10	8195,47	37409,07
128	5000	6	10	7820,28	36359,64
129	5000	6	10	8164,57	37525,76
130	5000	6	10	8235,69	37826,02
131	5000	6	50	8223,63	26969,53
132	5000	6	50	8272,99	37164,67
133	5000	6	50	8406,72	37559,16
134	5000	6	50	8352,38	37512,43
135	5000	6	50	8482,65	38330,53
136	5000	6	50	8403,04	38596,99
137	5000	6	50	8472,96	38441,28
138	5000	6	50	8449,18	38258,48
139	5000	6	50	8611,56	37934,06
140	5000	6	50	8813,64	37486,02
141	5000	6	100	8987,29	39138,27
142	5000	6	100	8939,22	39018,75
143	5000	6	100	9411,67	41244,38
144	5000	6	100	9335,83	40853,08
145	5000	6	100	9724,3	40644,4
146	5000	6	100	9034,89	39810,7
147	5000	6	100	9141,8	40526,47
148	5000	6	100	9206,32	39658,21
149	5000	6	100	9162,92	41740,9
150	5000	6	100	9205,5	40527,89

EK 2'nin devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
151	5000	9	10	17665,51	76504,56
152	5000	9	10	17892,38	77521,6
153	5000	9	10	18266,14	78953,15
154	5000	9	10	17410,02	77896,31
155	5000	9	10	17210,2	76957,88
156	5000	9	10	17411,36	77742,75
157	5000	9	10	17322,16	76709,08
158	5000	9	10	17405,82	77845,59
159	5000	9	10	17003,95	77476,52
160	5000	9	10	18907,63	79450,61
161	5000	9	50	18630,38	80533,51
162	5000	9	50	18992,96	82052,61
163	5000	9	50	18036,06	79474,03
164	5000	9	50	18007,47	78229,29
165	5000	9	50	18734,14	81617,28
166	5000	9	50	18743,81	80626,17
167	5000	9	50	18844,91	81043,62
168	5000	9	50	18244,83	79725,14
169	5000	9	50	18768,15	81189,4
170	5000	9	50	18023,27	79296,96
171	5000	9	100	22290,7	92914,93
172	5000	9	100	21186,23	88707,03
173	5000	9	100	19598,99	84946,52
174	5000	9	100	19530,8	84639,83
175	5000	9	100	19401,27	89657,27
176	5000	9	100	19596,78	88493,24
177	5000	9	100	20000,63	86315,49
178	5000	9	100	20360,87	88761,68
179	5000	9	100	19249,57	83563,71
180	5000	9	100	24038,48	94087,19

EK 2'nin devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
181	10000	3	10	10112,53	50726,88
182	10000	3	10	9472,17	48378,35
183	10000	3	10	9670,03	49600,46
184	10000	3	10	9475,95	49012,71
185	10000	3	10	9585,43	49128,25
186	10000	3	10	9429,58	48816,64
187	10000	3	10	9497,33	49114,51
188	10000	3	10	9344,63	48615,1
189	10000	3	10	9877,17	49640,37
190	10000	3	10	9720,63	49547,82
191	10000	3	50	9895,56	49509,1
192	10000	3	50	10328,11	50676,67
193	10000	3	50	10127,46	49946,16
194	10000	3	50	9795,71	49103,08
195	10000	3	50	10198,87	49810,23
196	10000	3	50	10010,31	49542,56
197	10000	3	50	10177,29	50796,12
198	10000	3	50	9936,51	50005,66
199	10000	3	50	9978,36	49729,26
200	10000	3	50	10092,53	49973,23
201	10000	3	100	10636,31	51184,19
202	10000	3	100	10848,19	51265,49
203	10000	3	100	10691,63	50967,35
204	10000	3	100	10550,86	50443,03
205	10000	3	100	10729,62	51009,62
206	10000	3	100	10620,69	51283,37
207	10000	3	100	10590,76	50799,89
208	10000	3	100	10782,97	51824,71
209	10000	3	100	10794,97	52089,2
210	10000	3	100	10897,2	51410,1

EK 2'nin devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
211	10000	6	10	35948,66	158393,34
212	10000	6	10	35204,27	157403,96
213	10000	6	10	34856,79	156063,91
214	10000	6	10	35177,29	157305,6
215	10000	6	10	35502	156265,92
216	10000	6	10	35552,82	157952,86
217	10000	6	10	34322,7	153413,3
218	10000	6	10	34479,8	155281,19
219	10000	6	10	34916,78	154788,74
220	10000	6	10	35615,96	156504,79
221	10000	6	50	35328,22	154434,69
222	10000	6	50	35128,3	154767,37
223	10000	6	50	36422,1	157928,05
224	10000	6	50	35827,57	157993,66
225	10000	6	50	35872,18	156913,22
226	10000	6	50	35923,91	155606
227	10000	6	50	35911,34	158530,87
228	10000	6	50	35155,02	162089,2
229	10000	6	50	36106,19	158761,45
230	10000	6	50	35603,93	155941,28
231	10000	6	100	37763,57	160392,05
232	10000	6	100	36936,57	158374,13
233	10000	6	100	38450,57	163983,47
234	10000	6	100	38601,62	162406,93
235	10000	6	100	36961,03	159753,75
236	10000	6	100	38751,66	165582,51
237	10000	6	100	37474,25	160865,89
238	10000	6	100	38682,81	165180,48
239	10000	6	100	38795,85	160307,45
240	10000	6	100	37707,25	160594,91

EK 2'nin devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
241	10000	9	10	87420,44	345543,3
242	10000	9	10	86850,41	341506,22
243	10000	9	10	89494,47	346960,34
244	10000	9	10	88526,71	345492,94
245	10000	9	10	87670,99	344381,49
246	10000	9	10	89029,49	342730,38
247	10000	9	10	85384,34	338948,26
248	10000	9	10	90390,93	346839,41
249	10000	9	10	88489,07	344952,91
250	10000	9	10	88817,33	346996,89
251	10000	9	50	90071,37	346489,3
252	10000	9	50	89675,83	341172,66
253	10000	9	50	88431,71	341200,42
254	10000	9	50	95105,67	353749,57
255	10000	9	50	89099,39	345302,24
256	10000	9	50	90169,31	347072,11
257	10000	9	50	92826,85	357379,23
258	10000	9	50	89924,03	348175,09
259	10000	9	50	93753,79	360338,8
260	10000	9	50	88152,49	345224,04
261	10000	9	100	101357,7	374688,47
262	10000	9	100	92214,57	349822,95
263	10000	9	100	93551,25	365278,55
264	10000	9	100	96724,37	367237,65
265	10000	9	100	97843,7	373033,21
266	10000	9	100	94558,31	365113,17
267	10000	9	100	94237,23	360691,33
268	10000	9	100	94517,29	368918,35
269	10000	9	100	95277,41	362013
270	10000	9	100	95979,86	372212,05

EK 3 Senaryo 3 için elde edilen deney sonuçları

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
1	100	3	10	1,91	6,67
2	100	3	10	1,42	6,13
3	100	3	10	1,44	5,69
4	100	3	10	1,6	4,93
5	100	3	10	1,4	4,93
6	100	3	10	1,59	7,87
7	100	3	10	1,88	6,66
8	100	3	10	1,48	5,67
9	100	3	10	1,34	5,35
10	100	3	10	1,43	5,51
11	100	3	50	5,23	12,83
12	100	3	50	2,29	8,13
13	100	3	50	2,06	7,84
14	100	3	50	2,2	7,84
15	100	3	50	2,34	8,79
16	100	3	50	2,79	10,51
17	100	3	50	2,28	9,12
18	100	3	50	2,23	8,18
19	100	3	50	2,22	8,97
20	100	3	50	2,17	7,92
21	100	3	100	7,28	17,79
22	100	3	100	4,39	16,45
23	100	3	100	4,61	15,41
24	100	3	100	4,57	13,74
25	100	3	100	4,45	14,27
26	100	3	100	4,29	14,14
27	100	3	100	3,91	12,52
28	100	3	100	4,55	15,37
29	100	3	100	4,56	15,25
30	100	3	100	4,64	15,27

EK 3'ün devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
31	100	6	10	7,24	19,18
32	100	6	10	3,49	13,91
33	100	6	10	4,25	14,76
34	100	6	10	3,85	14,57
35	100	6	10	3,79	14,49
36	100	6	10	3,96	14,55
37	100	6	10	3,64	13,71
38	100	6	10	3,41	13,13
39	100	6	10	3,83	14,83
40	100	6	10	4,41	16,31
41	100	6	50	8,62	25,61
42	100	6	50	4,58	17,53
43	100	6	50	4,55	20,16
44	100	6	50	6,21	25,8
45	100	6	50	4,64	18,96
46	100	6	50	5,8	24,78
47	100	6	50	4,53	17,1
48	100	6	50	5,77	24,63
49	100	6	50	5,39	21,22
50	100	6	50	4,83	19
51	100	6	100	11,69	40,05
52	100	6	100	9,76	41,64
53	100	6	100	9,93	45,51
54	100	6	100	9,73	54,81
55	100	6	100	8,44	30,33
56	100	6	100	11,7	46,39
57	100	6	100	8,13	35,78
58	100	6	100	8,1	30,15
59	100	6	100	9,37	38,59
60	100	6	100	8,04	28,37

EK 3'ün devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
61	100	9	10	11,78	36,83
62	100	9	10	8,71	32,86
63	100	9	10	9,89	33,9
64	100	9	10	8,98	34,81
65	100	9	10	6,95	24,82
66	100	9	10	8,17	30,54
67	100	9	10	8,84	31,7
68	100	9	10	9,94	34,45
69	100	9	10	7,58	28,19
70	100	9	10	7,79	29,36
71	100	9	50	14,42	48,19
72	100	9	50	14,8	59,22
73	100	9	50	9,08	41,36
74	100	9	50	9,05	49,06
75	100	9	50	11,04	54,48
76	100	9	50	12,53	58,95
77	100	9	50	9,99	44,37
78	100	9	50	8,91	42,17
79	100	9	50	9,49	47,43
80	100	9	50	12,15	55,62
81	100	9	100	17,61	91,4
82	100	9	100	12,75	54,49
83	100	9	100	17,63	98,46
84	100	9	100	14,93	89,43
85	100	9	100	16,52	78,78
86	100	9	100	18,81	75,82
87	100	9	100	15,02	83,14
88	100	9	100	18,2	90,43
89	100	9	100	17,44	92,9
90	100	9	100	14,72	74,35

EK 3'ün devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
91	5000	3	10	2609,4	10370,74
92	5000	3	10	2742,01	10604,09
93	5000	3	10	2651,98	10491,35
94	5000	3	10	2544,96	10174,76
95	5000	3	10	2661,08	10407,25
96	5000	3	10	3138,4	11387,79
97	5000	3	10	2609,15	10520,22
98	5000	3	10	2576,6	10436,62
99	5000	3	10	2682,86	10639,43
100	5000	3	10	2590,31	10387,42
101	5000	3	50	2738,36	10245,96
102	5000	3	50	2815,4	10041,85
103	5000	3	50	2766,09	9799,49
104	5000	3	50	2688,39	9755,93
105	5000	3	50	2823,24	10114,86
106	5000	3	50	2701,87	9761,56
107	5000	3	50	2744,15	9912,43
108	5000	3	50	2724,93	9855,55
109	5000	3	50	2675,47	9696,85
110	5000	3	50	2686,39	10009,42
111	5000	3	100	2913,78	9758,03
112	5000	3	100	2802,68	9648,49
113	5000	3	100	2683,77	9454,82
114	5000	3	100	2794,33	9785,51
115	5000	3	100	2770,98	9622,53
116	5000	3	100	2682,98	9291,3
117	5000	3	100	2821,09	9773,75
118	5000	3	100	2850,67	9859,39
119	5000	3	100	2741,66	9589,22
120	5000	3	100	2829,38	9772,17

EK 3'ün devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
121	5000	6	10	10882	36798,66
122	5000	6	10	9700,46	34486,67
123	5000	6	10	9775,98	35255,59
124	5000	6	10	9762,54	35564,6
125	5000	6	10	9399,32	34494,74
126	5000	6	10	9179,21	33037,24
127	5000	6	10	9144,2	33627,34
128	5000	6	10	9415,07	34012,81
129	5000	6	10	9456,56	34376,52
130	5000	6	10	9453,72	34190,87
131	5000	6	50	9445,12	31071,84
132	5000	6	50	9548,04	31570,72
133	5000	6	50	9582,52	31879,2
134	5000	6	50	9298,23	31210,13
135	5000	6	50	9520,32	31832,2
136	5000	6	50	10189,29	32532,51
137	5000	6	50	9143,36	31326,93
138	5000	6	50	10029,77	32649,59
139	5000	6	50	9742,08	31376,4
140	5000	6	50	9240,99	30705,62
141	5000	6	100	9817,64	31241,12
142	5000	6	100	9576,48	30808,01
143	5000	6	100	9685,46	31593,68
144	5000	6	100	9646,15	31219,75
145	5000	6	100	9879,17	31815,6
146	5000	6	100	9712,22	31106,94
147	5000	6	100	9412,85	30889,2
148	5000	6	100	9506,91	30864,4
149	5000	6	100	9358,28	31229,85
150	5000	6	100	9531,96	31480,43

EK 3'ün devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
151	5000	9	10	21754,65	69075,15
152	5000	9	10	20451,76	70105,99
153	5000	9	10	20840,49	71670,37
154	5000	9	10	21276,36	69784,31
155	5000	9	10	21593,09	70415,7
156	5000	9	10	20814,35	70733,17
157	5000	9	10	20263,72	71424,43
158	5000	9	10	21726,58	71315,27
159	5000	9	10	21338,77	70562,32
160	5000	9	10	21352,86	70866,99
161	5000	9	50	22900,04	69462,83
162	5000	9	50	21816,62	68412,76
163	5000	9	50	20807,63	66142,77
164	5000	9	50	20785,53	67130,34
165	5000	9	50	22824,47	71125,23
166	5000	9	50	21372,31	68815,51
167	5000	9	50	22150,21	68641,56
168	5000	9	50	21175,81	67198,16
169	5000	9	50	21822,47	69487,19
170	5000	9	50	20957,84	67421,78
171	5000	9	100	23387,92	71634,48
172	5000	9	100	24864,71	70817,48
173	5000	9	100	20748,88	65143,62
174	5000	9	100	21986,26	69067,1
175	5000	9	100	20982,77	68267,97
176	5000	9	100	21107,26	68035,06
177	5000	9	100	22132,57	67343,12
178	5000	9	100	21764,28	69585,51
179	5000	9	100	21686,34	66536,23
180	5000	9	100	21291,51	67488

EK 3'ün devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
181	10000	3	10	11240,41	43767,29
182	10000	3	10	11869,63	45143,81
183	10000	3	10	11700,84	45190,48
184	10000	3	10	11097	43473,46
185	10000	3	10	11394,5	44271,3
186	10000	3	10	10958,96	43353,94
187	10000	3	10	11088,8	44234,26
188	10000	3	10	11118,67	43516,35
189	10000	3	10	11294,21	44857,6
190	10000	3	10	11247,9	44442,23
191	10000	3	50	11290,1	44360,97
192	10000	3	50	11614,56	44889,41
193	10000	3	50	11614,06	45466,97
194	10000	3	50	11827,31	45535,81
195	10000	3	50	11346,95	44117,59
196	10000	3	50	11497,69	42870,3
197	10000	3	50	11697,07	44398,89
198	10000	3	50	11358,21	43469,16
199	10000	3	50	12240,4	43671,44
200	10000	3	50	11507,02	42989,12
201	10000	3	100	11620,1	45323,44
202	10000	3	100	11825,7	45276,31
203	10000	3	100	11627,28	44371,8
204	10000	3	100	11463,8	44493,3
205	10000	3	100	13098,96	46944,78
206	10000	3	100	12327,76	45909,3
207	10000	3	100	12089,4	45240,89
208	10000	3	100	11585,67	45845,02
209	10000	3	100	12014,93	45844,59
210	10000	3	100	12023,64	45399,78

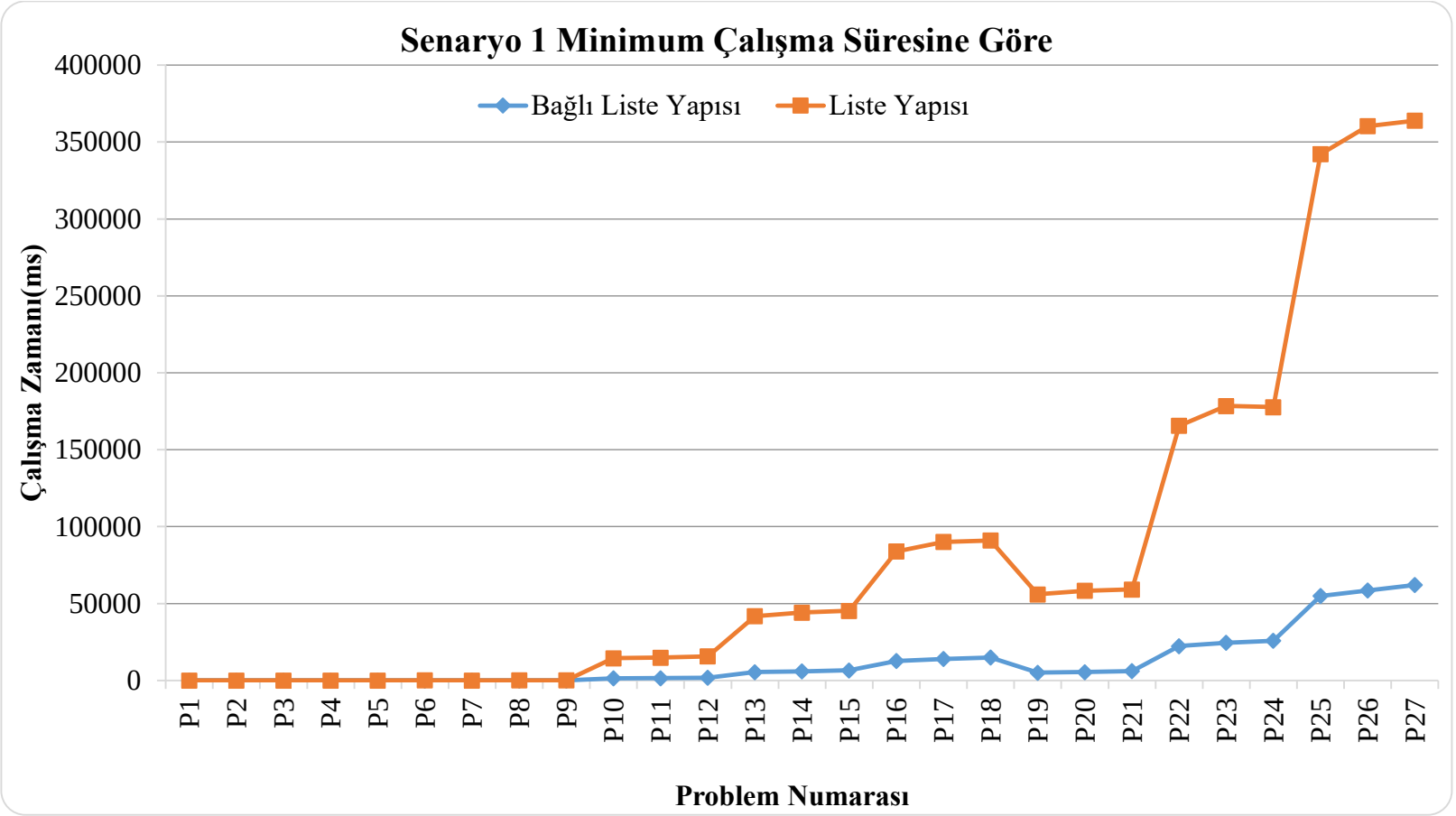
EK 3'ün devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
211	10000	6	10	44510,48	153155,14
212	10000	6	10	44369,08	150069,27
213	10000	6	10	45170,86	151250,33
214	10000	6	10	44727,49	153822,34
215	10000	6	10	47020,31	155080,83
216	10000	6	10	45261,36	154609,06
217	10000	6	10	43155,92	147289,68
218	10000	6	10	44326,27	148613,24
219	10000	6	10	45623,32	149828,6
220	10000	6	10	46651,63	153436,69
221	10000	6	50	44866,04	161097,9
222	10000	6	50	45107,58	162621,36
223	10000	6	50	45545,6	162209,57
224	10000	6	50	45666,81	163599,35
225	10000	6	50	44823,99	159567,4
226	10000	6	50	44043,25	159151,6
227	10000	6	50	44696,35	160199,11
228	10000	6	50	46283,92	168669,04
229	10000	6	50	44876,41	160795,93
230	10000	6	50	43723,79	157502,62
231	10000	6	100	45761,9	176997,53
232	10000	6	100	44867,45	174023,69
233	10000	6	100	45620,42	178428
234	10000	6	100	46377,64	180069,64
235	10000	6	100	44949,04	177258,68
236	10000	6	100	46578,47	179872,99
237	10000	6	100	44697,11	175735,87
238	10000	6	100	45672,21	177957,94
239	10000	6	100	45581,2	177066,61
240	10000	6	100	46922,07	179726,09

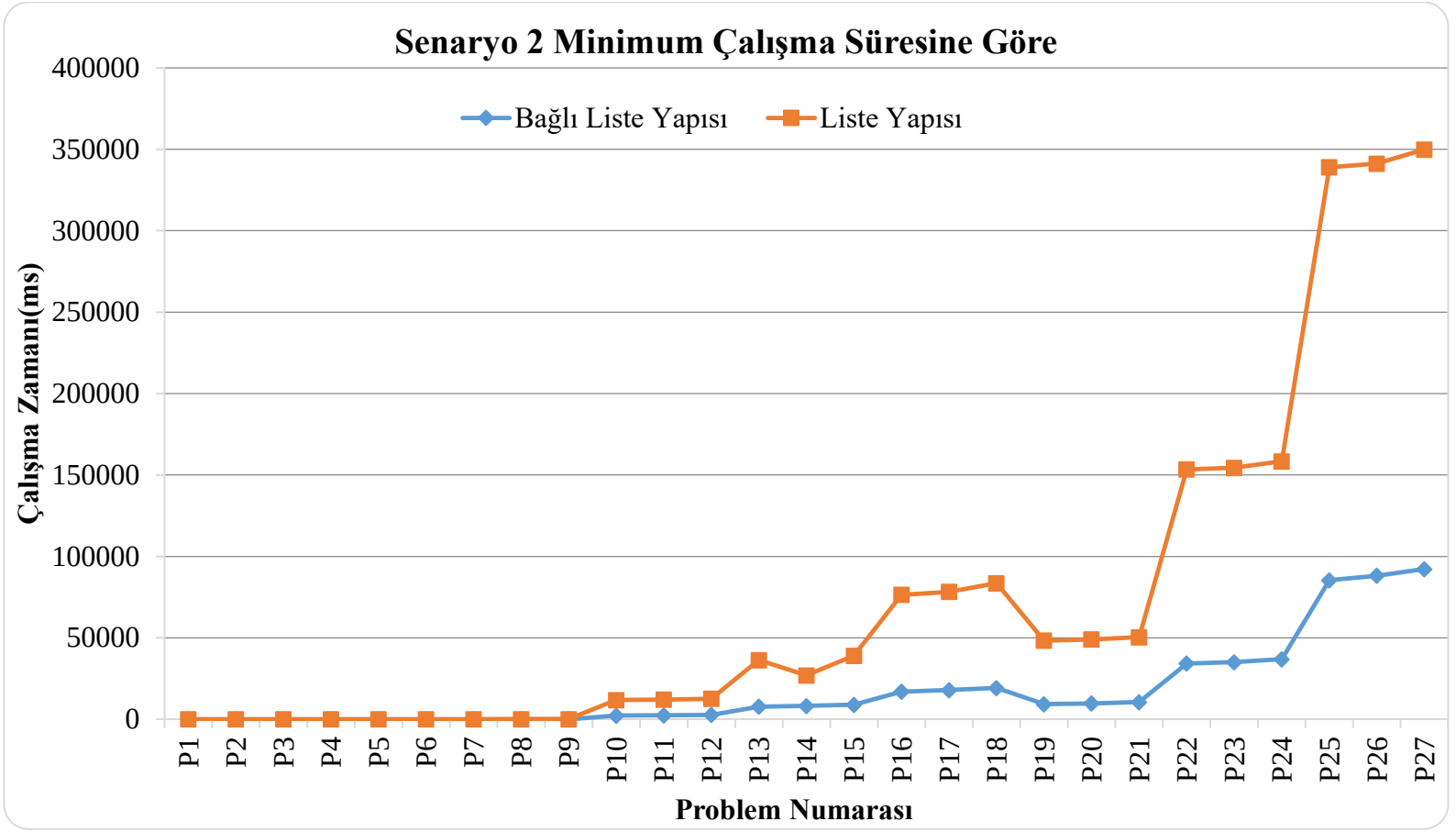
EK 3'ün devamı

Test Örnek No	İş Sayısı	Operasyon Sayısı	Makine Sayısı	Bağlı Liste Yapısı Çalışma Süresi	Liste Yapısı Çalışma Süresi
241	10000	9	10	114732,54	363845,55
242	10000	9	10	112126,38	358840,34
243	10000	9	10	112484,76	362899,3
244	10000	9	10	115209,68	378179,47
245	10000	9	10	114622,23	377711,2
246	10000	9	10	111729,36	357362,31
247	10000	9	10	111374,73	353500,71
248	10000	9	10	112512,76	363874,62
249	10000	9	10	114777,47	364632,29
250	10000	9	10	112631,98	360487,46
251	10000	9	50	111338,57	365811,13
252	10000	9	50	111866,2	359843,21
253	10000	9	50	113483,6	374326,06
254	10000	9	50	116135,26	376396,16
255	10000	9	50	116806,46	373973,87
256	10000	9	50	113886,93	373980,34
257	10000	9	50	114193,52	376521,24
258	10000	9	50	114702,27	367342,82
259	10000	9	50	116758,2	379467,63
260	10000	9	50	114281,61	377551,68
261	10000	9	100	116163,28	383274,84
262	10000	9	100	114211,78	372723,02
263	10000	9	100	115774,34	380804,16
264	10000	9	100	117237,1	381018,67
265	10000	9	100	114339,65	382238,63
266	10000	9	100	118909,92	382861,79
267	10000	9	100	117439,92	379458,54
268	10000	9	100	119092,26	381322,06
269	10000	9	100	117355,98	380444,49
270	10000	9	100	117672,34	385892,5

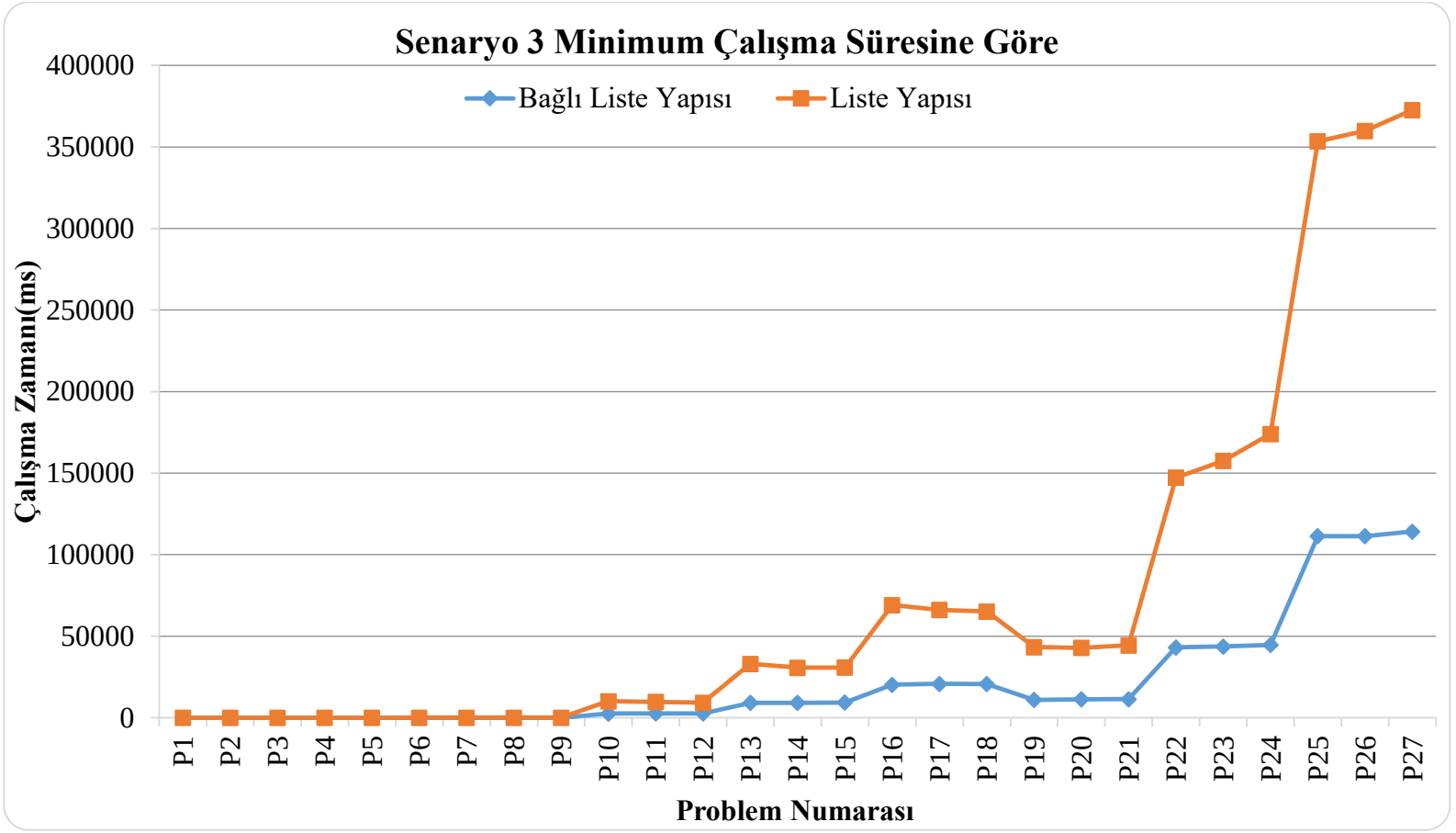
EK 4 Senaryo 1 minimum çalışma süresine göre sonuç grafiği



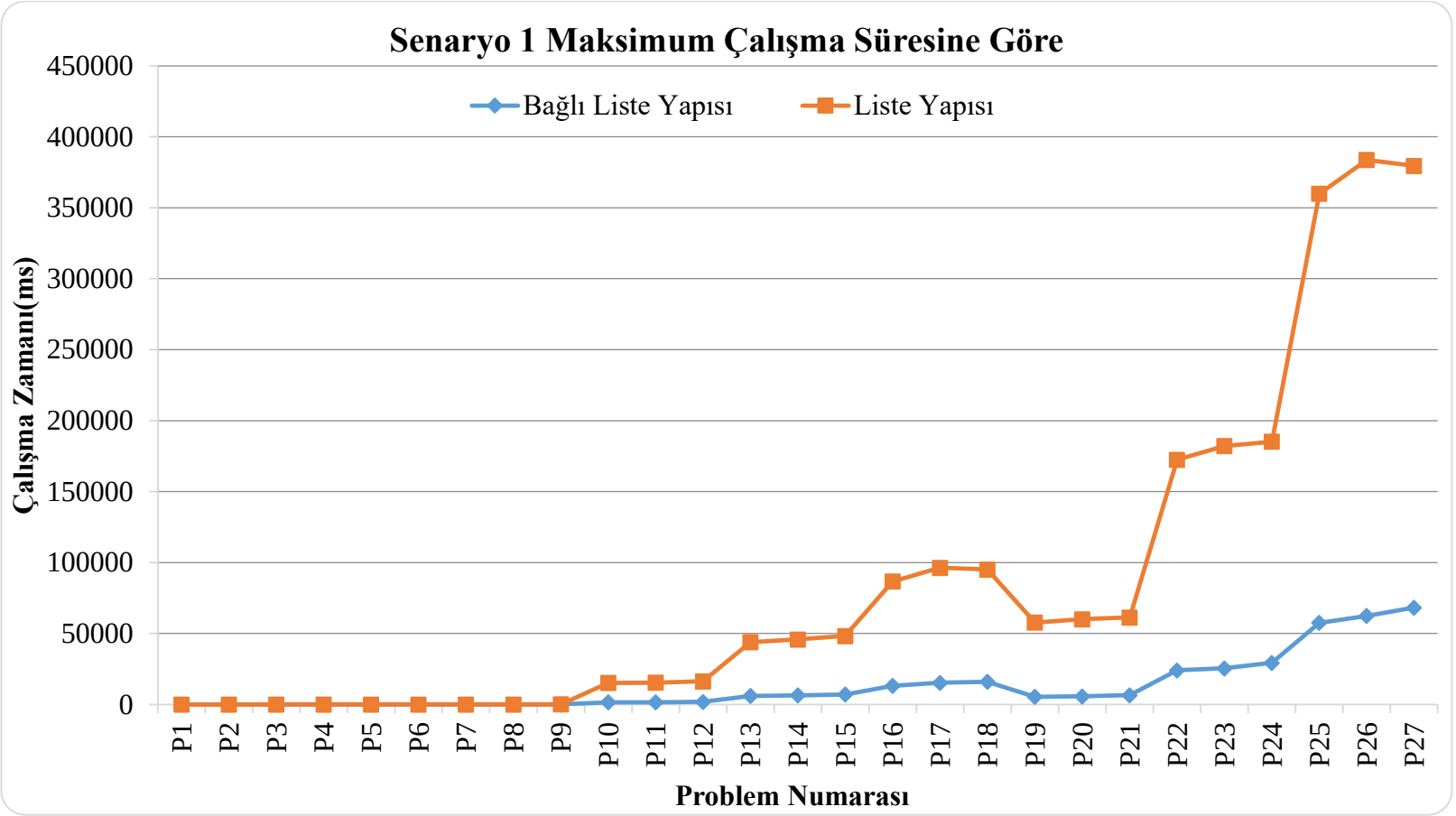
EK 5 Senaryo 2 minimum çalışma süresine göre sonuç grafiği



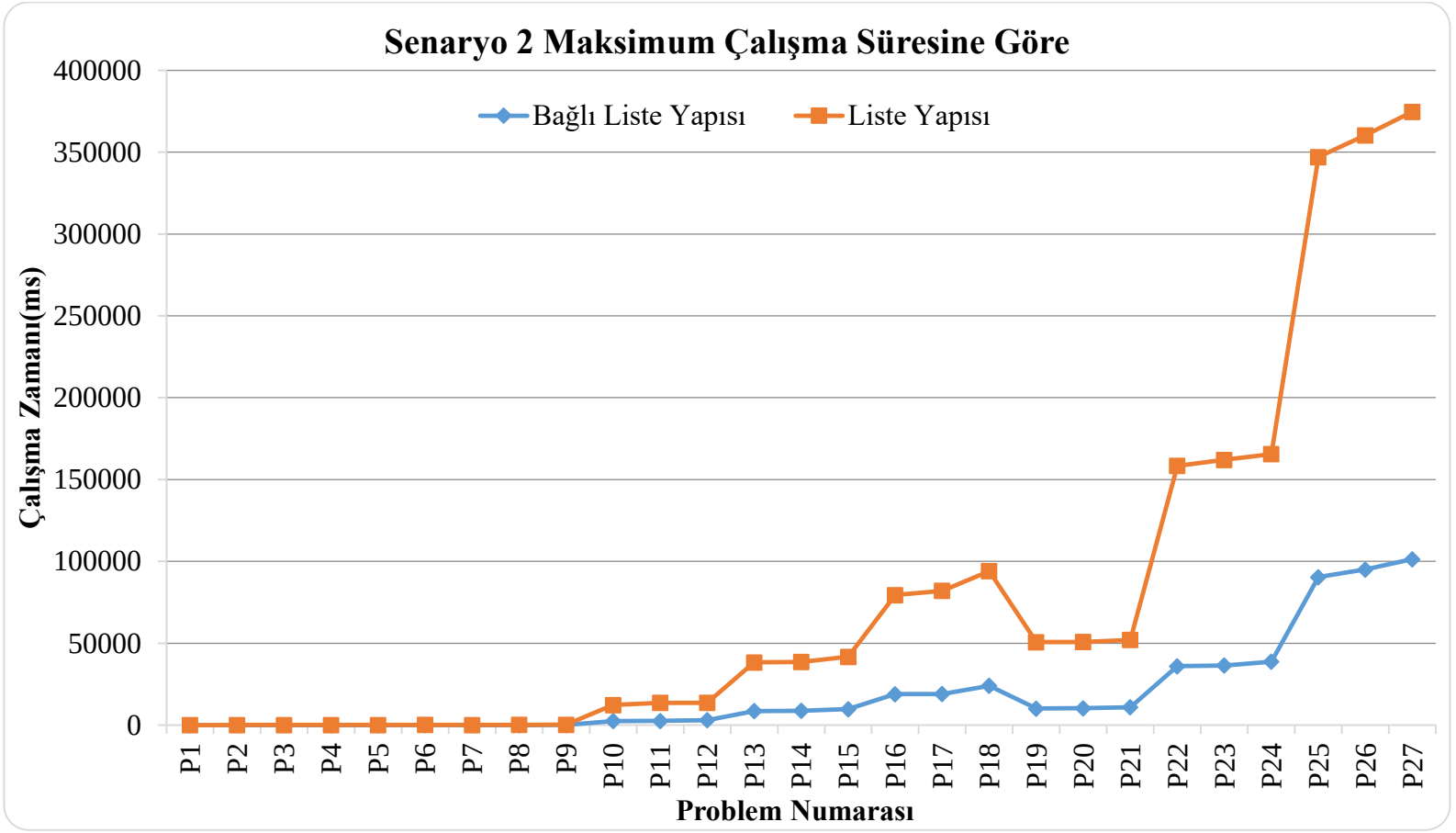
EK 6 Senaryo 3 minimum çalışma süresine göre sonuç grafiği



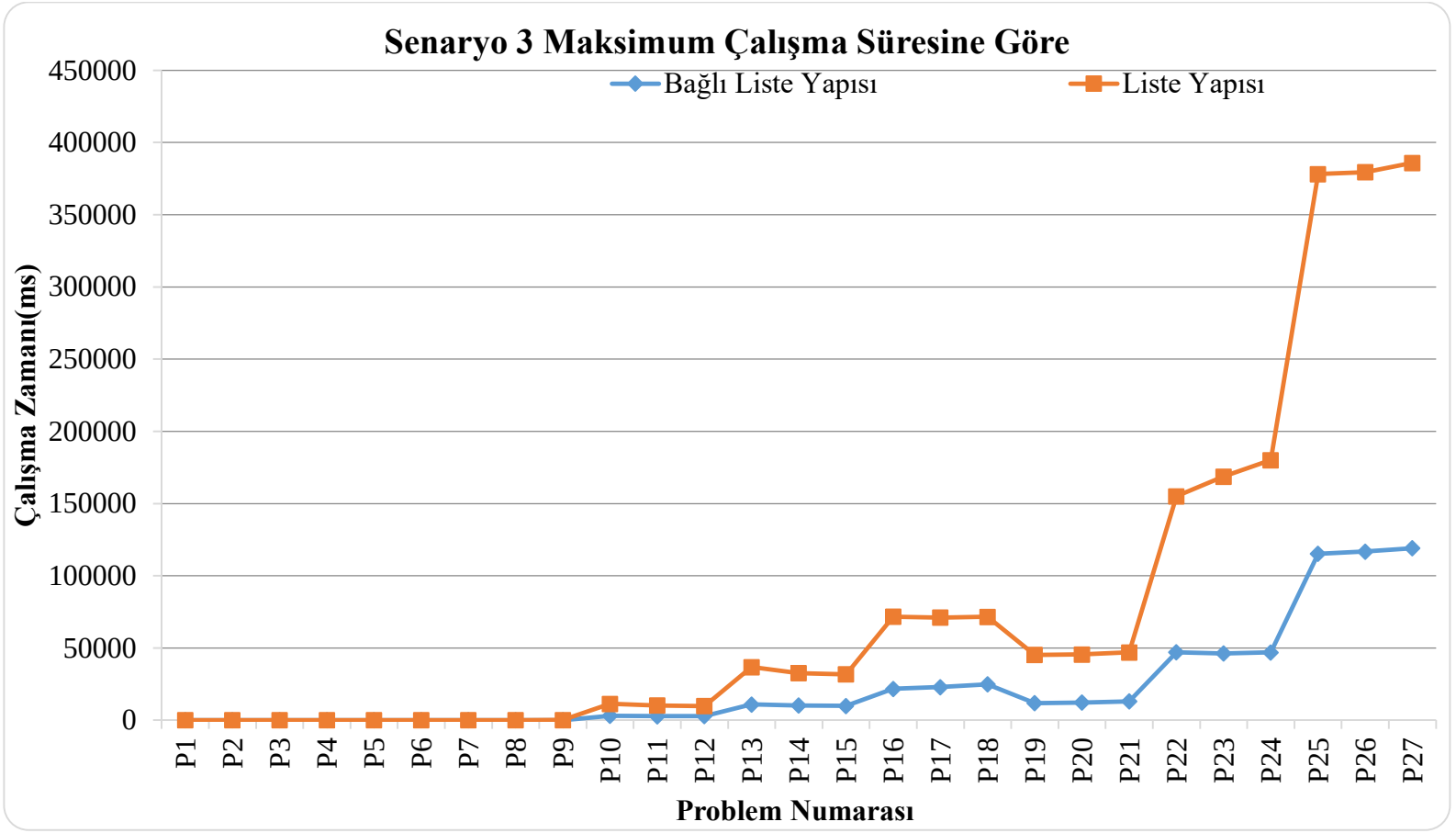
EK 7 Senaryo 1 maksimum çalışma süresine göre sonuç grafiği



EK 8 Senaryo 2 maksimum çalışma süresine göre sonuç grafiği



EK 9 Senaryo 3 maksimum çalışma süresine göre sonuç grafiği



EK 10 Senaryo 1 ve Senaryo 2 için C# kodu

Data.cs

```
class Data
   {
      public static List<Operation> operations = new List<Operation>();
      public static List<Machine> machines = new List<Machine>();
      public static List<Makine_Grup> makine_grupları = new List<Makine_Grup>();
      public static void is_atama()
      {
         foreach (var op_item in operations)
         {
            op_item.op_atama();
         }
      }
   }
   class Data2
   {
      public List<Operation> operations = new List<Operation>();
      public List<Machine> machines = new List<Machine>();
      public List<Makine_Grup> makine_grupları = new List<Makine_Grup>();
      public void kopyala()
      {
         operations = Data.operations;
         machines = Data.machines;
         makine_grupları = Data.makine_grupları;
      }
   }

   class Data_2
   {
      public static List<Operation> operations = new List<Operation>();
```

```
    public static List<Machine2> machines = new List<Machine2>();
    public static List<Makine_Grup> makine_grupları = new List<Makine_Grup>();

    public static void is_atama2()
    {
        foreach (var op_item in operations)
        {
            op_item.op_atama2();
        }

    }
}
class Data2_2
{
    public List<Operation> operations = new List<Operation>();
    public List<Machine2> machines = new List<Machine2>();
    public List<Makine_Grup> makine_grupları = new List<Makine_Grup>();
    public void kopyala()
    {
        operations = Data_2.operations;
        machines = Data_2.machines;
        makine_grupları = Data_2.makine_grupları;
    }
}

class Makine_Grup
{
    public string mak_grup_id;
    public List<string> makine_ids;
    public Makine_Grup()
    {
        mak_grup_id = "-1";
```

```
            makine_ids = new List<string>();
        }
    }
    class Operation
    {
        public string id;
        public string op_id;
        public string is_id;
        public double op_suresi;
        public string mak_grup_id;
        //public List<string> makine_ids;
        public List<string> onsart_ids;
        public List<string> onsart_ids_temp; //count eşit olana kadar serbest bırakmayacağız
        public double baslangıc_zamanı;
        public double bitis_zamanı;
        public string atanan_makine;
        public List<string> atanan_makineler;
        public double en_erken_baslama_zamanı;
        //liste tipi olanları initialize ediyoruz hata almamak ve kontrol etmek için
        public Operation()
        {
            id = "-1";
            op_id = "-1";
            is_id = "-1";
            op_suresi = -1;
            mak_grup_id = "-1";
            //makine_ids = new List<string>();
            //eğer parçalı esneklik varsa 53teki yapı, tam esneklik varsa 52deki yapı
            onsart_ids = new List<string>();
            baslangıc_zamanı = -1;
            bitis_zamanı = -1;
```

```
        atanan_makine = "-1";
        en_erken_baslama_zamanı = -1;
    }
    public void op_atama()
    {
        string atanabilecek_makine = "-1";
        double atanabilecek_baslama_zamanı = 0;
        double opt_baslama_zamani = double.MaxValue;
        bool conti = true;

        if (onsart_ids.Count != 0) en_erken_baslama_zamanı = Data.operations.FindAll(x => onsart_ids.Contains(x.id)).Select(x => x.bitis_zamanı).ToList().Max();
        else en_erken_baslama_zamanı = 0;

        List<string> alt_mak = new List<string>();
        alt_mak = Data.makine_grupları.Find(x => x.mak_grup_id == this.mak_grup_id).makine_ids;

        foreach (var mach_item in alt_mak)
        {
            var atanacak_makine = Data.machines.Find(x => x.machine_id == mach_item);
            Bosluk mevcut_bosluk = atanacak_makine.makine_ilk_bosluk;
            conti = true;
            var atama_sonucu = mevcut_bosluk.atama(this, mach_item, conti, ref atanabilecek_baslama_zamanı, ref opt_baslama_zamani, ref atanabilecek_makine);
            while (!mevcut_bosluk.atama(this, mach_item, conti, ref atanabilecek_baslama_zamanı, ref opt_baslama_zamani, ref atanabilecek_makine) && mevcut_bosluk.sonraki_bosluk != null)
            {
                mevcut_bosluk = mevcut_bosluk.sonraki_bosluk;
```

```
            }
        }
        conti = false;
        Bosluk mevcut_bosluk2 = Data.machines.Find(x => x.machine_id == atanabilecek_makine).makine_ilk_bosluk;
        while (!mevcut_bosluk2.atama(this, atanabilecek_makine, conti, ref atanabilecek_baslama_zamanı, ref opt_baslama_zamani, ref atanabilecek_makine) && mevcut_bosluk2.sonraki_bosluk != null)
        {
            mevcut_bosluk2 = mevcut_bosluk2.sonraki_bosluk;
        }

    }
    public void op_atama2()
    {
        string atanabilecek_makine = "-1";
        double atanabilecek_baslama_zamanı = 0;
        double opt_baslama_zamani = double.MaxValue;
        bool conti = false;
        if (onsart_ids.Count != 0)
        {
            en_erken_baslama_zamanı = Data_2.operations.FindAll(x => onsart_ids.Contains(x.id)).Select(x => x.bitis_zamanı).ToList().Max();
        }
        else en_erken_baslama_zamanı = 0;
        List<string> alt_mak = new List<string>();
        alt_mak = Data_2.makine_grupları.Find(x => x.mak_grup_id == this.mak_grup_id).makine_ids;

        foreach (var mach_item in alt_mak)
        {
            var mak = Data_2.machines.Find(x => x.machine_id == mach_item);
```

```
            conti = false;
            for (int i = 0; i < mak.bosluk_listesi.Count; i++)
            {
               if (i < en_erken_baslama_zamanı) continue;
               if (atanabilecek_baslama_zamanı <= i && atanabilecek_baslama_zamanı >
0) break;

               if (mak.bosluk_listesi[i] == 100)
               {
                  for (int j = i; j < i + op_suresi; j++)
                  {
                     if (mak.bosluk_listesi[j] == 100)
                     {
                        conti = true;
                     }
                     else
                     {
                        conti = false;
                        break;
                     }
                  }
               }
               if (conti == true)
               {
                  atanabilecek_baslama_zamanı = i;
                  break;
               }
            }
            if (conti == true && atanabilecek_baslama_zamanı < opt_baslama_zamani)
            {
               opt_baslama_zamani = atanabilecek_baslama_zamanı;
               atanabilecek_makine = mach_item;
```

```
            }
        }
        var mak2 = Data_2.machines.Find(x => x.machine_id == atanabilecek_makine);
        atanan_makine = mak2.machine_id.ToString();
        baslangıc_zamanı = opt_baslama_zamani;
        bitis_zamanı = opt_baslama_zamani + op_suresi;
        for (int k = (int)atanabilecek_baslama_zamanı; k < (int)atanabilecek_baslama_zamanı + op_suresi; k++)
        {
            mak2.bosluk_listesi[k] = 0;
        }
    }
}

class Machine
{
    public string machine_id;
    //makinenin boşluklarını vermeli
    //boşluk boyutu, boşluk ne zaman başlıyor?
    //public List<Bosluk> makine_boslukları;
    public Bosluk makine_ilk_bosluk;
}
class Machine2
{
    public string machine_id;
    public List<int> bosluk_listesi = new List<int>();//2.boşluk tutma stili
}

class Bosluk
{
    public double bosluk_suresi;
    public double bosluk_baslangıc_zamanı;
```

```
public Bosluk önceki_bosluk;
public Bosluk sonraki_bosluk;

public bool atama(Operation atanacak_op, string atanacak_mak, bool conti, ref double atanabilecek_baslama_zamanı, ref double opt_baslama_zamani, ref string atanabilecek_makine)
{
  //Operasyonun ön şartıyla ilgili başlangıç zamanı ayarlamaları daha önceden yapıldığı varsayılır.
  bool result = false;
  //4 atama alternatifinden
  double geçici_baslangıç_zamanı;
  geçici_baslangıç_zamanı = Math.Max(atanacak_op.en_erken_baslama_zamanı, bosluk_baslangıc_zamanı);

  if (geçici_baslangıç_zamanı + atanacak_op.op_suresi <= bosluk_baslangıc_zamanı + bosluk_suresi)
  {
    result = true;
    if (conti == false)
    {
      //1-2 ilk if---başlangıçları=bosluk başlangıç zamanı
      //!-1den başlangıç zamanı olması sıfırdan küçük ya sorun olur mu acaba önşartlarla bak
      if (geçici_baslangıç_zamanı <= bosluk_baslangıc_zamanı)
      {
        if (atanacak_op.op_suresi == bosluk_suresi)
        {
          //SENARYO 1: op_süresi ile boşluk süresi aynıysa ona ata
          if (önceki_bosluk == null)
          {
            atanacak_op.baslangıc_zamanı = bosluk_baslangıc_zamanı;
```

```
                    atanacak_op.bitis_zamanı = atanacak_op.baslangıc_zamanı +
atanacak_op.op_suresi;
                    atanacak_op.atanan_makine = atanacak_mak;
                    sonraki_bosluk.önceki_bosluk = null;
                }
                else
                {
                    atanacak_op.baslangıc_zamanı = bosluk_baslangıc_zamanı;
                    atanacak_op.bitis_zamanı = atanacak_op.baslangıc_zamanı +
atanacak_op.op_suresi;
                    atanacak_op.atanan_makine = atanacak_mak;
                    önceki_bosluk.sonraki_bosluk = sonraki_bosluk;
                    sonraki_bosluk.önceki_bosluk = önceki_bosluk;
                }
            }
            else
            {
                //SENARYO 2: operasyonun en baştan atandığı sonda boşluk kaldığı
durum
                atanacak_op.baslangıc_zamanı = bosluk_baslangıc_zamanı;
                atanacak_op.bitis_zamanı = atanacak_op.baslangıc_zamanı +
atanacak_op.op_suresi;
                atanacak_op.atanan_makine = atanacak_mak;
                bosluk_baslangıc_zamanı = geçici_baslangıç_zamanı +
atanacak_op.op_suresi;
                bosluk_suresi = bosluk_suresi - atanacak_op.op_suresi;
            }
        }
        else
        {
            if ((geçici_baslangıç_zamanı + atanacak_op.op_suresi <
bosluk_baslangıc_zamanı + bosluk_suresi))
```

```
                {
                    ////SENARYO 3: boşluğun arasında bir yere işin atandığı(başa ve sona değmiyor)
                    atanacak_op.baslangıc_zamanı = geçici_baslangıç_zamanı;
                    atanacak_op.bitis_zamanı = geçici_baslangıç_zamanı + atanacak_op.op_suresi;
                    atanacak_op.atanan_makine = atanacak_mak;

                    Bosluk Yeni_Bosluk = new Bosluk();
                    Yeni_Bosluk.sonraki_bosluk = sonraki_bosluk;
                    Yeni_Bosluk.önceki_bosluk = this;//*****
                    Yeni_Bosluk.bosluk_baslangıc_zamanı = geçici_baslangıç_zamanı + atanacak_op.op_suresi;
                    Yeni_Bosluk.bosluk_suresi = bosluk_baslangıc_zamanı + bosluk_suresi - (geçici_baslangıç_zamanı + atanacak_op.op_suresi);

                    sonraki_bosluk = Yeni_Bosluk;
                    bosluk_suresi = geçici_baslangıç_zamanı - bosluk_baslangıc_zamanı;
                }
                else
                {
                    //bosluk_suresi = (geçici_baslangıç_zamanı - bosluk_baslangıc_zamanı);
                    ////SENARYO 4: operasyonun bitişinin boşluğun sonuna değdiği
                    atanacak_op.baslangıc_zamanı = geçici_baslangıç_zamanı;
                    //atanacak_op.baslangıc_zamanı = bosluk_baslangıc_zamanı + bosluk_suresi - atanacak_op.op_suresi;
                    atanacak_op.bitis_zamanı = bosluk_baslangıc_zamanı + bosluk_suresi;
                    atanacak_op.atanan_makine = atanacak_mak;
                    bosluk_suresi = bosluk_suresi - atanacak_op.op_suresi;
                }
            }
```

```
            }
            else
            {
                atanabilecek_baslama_zamanı = geçici_baslangıç_zamanı;
                if (atanabilecek_baslama_zamanı < opt_baslama_zamani)
                {
                    opt_baslama_zamani = atanabilecek_baslama_zamanı;
                    atanabilecek_makine = atanacak_mak;
                }
            }
        }
        return result;
    }
}
```

Program.cs

```
static int is_sayisi = 100;
static int max_operasyon_sayisi = 3;
static int makine_sayisi = 50;
static int makine_grup_sayısı = 1;
static Bosluk ilk_bosluk = new Bosluk();
static TimeSpan gecenSure;
int i = 1; // üretilen veri setinin benzer örnekteki kaçıncı problem olduğunu gösterir
        var e1_path = @"filepath\\datam1veri_j" + is_sayisi.ToString() + "o" + max_operasyon_sayisi.ToString() + "m" + makine_sayisi.ToString() + "g" + makine_grup_sayısı.ToString() + "i" + i.ToString() + ".json";
        var e2_path = @"filepath\\datam2veri_j" + is_sayisi.ToString() + "o" + max_operasyon_sayisi.ToString() + "m" + makine_sayisi.ToString() + "g" + makine_grup_sayısı.ToString() + "i" + i.ToString() + ".json";

        Data2 datam = JsonConvert.DeserializeObject<Data2>(File.ReadAllText(e1_path));
        Data.operations = datam.operations;
```

```
        Data.machines = datam.machines;
        Data.makine_grupları = datam.makine_grupları;

        Data2_2 datam2 = JsonConvert.DeserializeObject<Data2_2>(File.ReadAllText(e2_path));
        Data_2.operations = datam2.operations;
        Data_2.machines = datam2.machines;
        Data_2.makine_grupları = datam2.makine_grupları;

        Stopwatch stopWatch = new Stopwatch();
        if (!stopWatch.IsRunning)
           stopWatch.Restart();
        stopWatch.Start();
        Data.is_atama();
        stopWatch.Stop();

        gecenSure = stopWatch.Elapsed;
        Console.Out.WriteLine("Calisma Suresi=" + "\t" + gecenSure.Hours.ToString() + " saat " + gecenSure.Minutes.ToString() + " dakika " + gecenSure.Seconds.ToString() + " saniye " + gecenSure.Milliseconds.ToString() + " salise = " + gecenSure.TotalMilliseconds.ToString());
        Console.Out.WriteLine("Calisma Suresi=" + "\t" + Data.op_atama_suresi.Hours.ToString() + " saat " + Data.op_atama_suresi.Minutes.ToString() + " dakika " + Data.op_atama_suresi.Seconds.ToString() + " saniye " + Data.op_atama_suresi.Milliseconds.ToString() + " salise = " + Data.op_atama_suresi.TotalMilliseconds.ToString());

        stopWatch.Reset();
        stopWatch.Start();
        Data_2.is_atama2();
        stopWatch.Stop();
```

```
        gecenSure = stopWatch.Elapsed;
        Console.Out.WriteLine("Calisma Suresi=" + "\t" + gecenSure.Hours.ToString()
+ " saat " + gecenSure.Minutes.ToString() + " dakika " + gecenSure.Seconds.ToString()
+ " saniye " + gecenSure.Milliseconds.ToString() + " salise = " +
gecenSure.TotalMilliseconds.ToString());
        Console.Out.WriteLine("Calisma Suresi=" + "\t" +
Data_2.op_atama_suresi.Hours.ToString() + " saat " +
Data_2.op_atama_suresi.Minutes.ToString() + " dakika " +
Data_2.op_atama_suresi.Seconds.ToString() + " saniye " +
Data_2.op_atama_suresi.Milliseconds.ToString() + " salise = " +
Data_2.op_atama_suresi.TotalMilliseconds.ToString());

        Console.Out.WriteLine("İş No" + "\t" + "Op No" + "\t" + "Makine" + "\t" +
"Basl." + "\t" + "Bit." + "\t" + "Operasyon Süresi");
        foreach (var op_item in Data.operations)
        {
          Console.Out.WriteLine(op_item.is_id + "\t" + op_item.op_id + "\t" +
op_item.atanan_makine + "\t" + op_item.baslangıc_zamanı + "\t" + op_item.bitis_zamanı
+ "\t" + op_item.op_suresi);
        }
        Console.Out.WriteLine(Environment.NewLine + "OK");
        Console.Out.WriteLine("İş No" + "\t" + "Op No" + "\t" + "Makine" + "\t" +
"Basl." + "\t" + "Bit." + "\t" + "Operasyon Süresi");
        foreach (var op_item in Data_2.operations)
        {
          Console.Out.WriteLine(op_item.is_id + "\t" + op_item.op_id + "\t" +
op_item.atanan_makine + "\t" + op_item.baslangıc_zamanı + "\t" + op_item.bitis_zamanı
+ "\t" + op_item.op_suresi);
        }

        //VERİ ÇIKTISI YAZDIRMA
```

```
var path1 = @" filepath \\senaryo2_datam1veri_j" + is_sayisi.ToString() + "o" + max_operasyon_sayisi.ToString() + "m" + makine_sayisi.ToString() + "g" + makine_grup_sayısı.ToString() + "i" + i.ToString() + ".json";
var path2 = @"filepath\\senaryo2_datam2veri_j" + is_sayisi.ToString() + "o" + max_operasyon_sayisi.ToString() + "m" + makine_sayisi.ToString() + "g" + makine_grup_sayısı.ToString() + "i" + i.ToString() + ".json";

Data2_2 w_datam2 = new Data2_2();
w_datam2.kopyala();
File.WriteAllText(path2, JsonConvert.SerializeObject(w_datam2));

foreach (var mak_item in Data.machines)
{
   mak_item.makine_ilk_bosluk = null;
}
Data2 w_datam = new Data2();
w_datam.kopyala();
File.WriteAllText(path1, JsonConvert.SerializeObject(w_datam));
Console.Out.WriteLine(i.ToString() + ". çıktılar yazdırıldı");

Console.ReadLine();
```

EK 11 Senaryo 3 için C# kodu

Data.cs

```
class Data
   {
      public static List<Operation> operations = new List<Operation>();
      public static List<Machine> machines = new List<Machine>();
      public static List<Makine_Grup> makine_grupları = new List<Makine_Grup>();
      public static void is_atama()
      {
         foreach (var op_item in operations)
         {
            op_item.op_atama();
         }
      }
   }
   class Data2
   {
      public List<Operation> operations = new List<Operation>();
      public List<Machine> machines = new List<Machine>();
      public List<Makine_Grup> makine_grupları = new List<Makine_Grup>();
      public void kopyala()
      {
         operations = Data.operations;
         machines = Data.machines;
         makine_grupları = Data.makine_grupları;
      }
   }

   class Data_2
   {
      public static List<Operation> operations = new List<Operation>();
```

```
    public static List<Machine2> machines = new List<Machine2>();
    public static List<Makine_Grup> makine_grupları = new List<Makine_Grup>();

    public static void is_atama2()
    {
        // Hiçbir operasyon önşartlarından önce atanmamalıdır.
        foreach (var op_item in operations)
        {
            op_item.op_atama2();
        }
    }
}
class Data2_2
{
    public List<Operation> operations = new List<Operation>();
    public List<Machine2> machines = new List<Machine2>();
    public List<Makine_Grup> makine_grupları = new List<Makine_Grup>();
    public void kopyala()
    {
        operations = Data_2.operations;
        machines = Data_2.machines;
        makine_grupları = Data_2.makine_grupları;
    }
}

class Makine_Grup
{
    public string mak_grup_id;
    public List<string> makine_ids;
    public Makine_Grup()
    {
        mak_grup_id = "-1";
```

```
            makine_ids = new List<string>();
        }
    }

    class Operation
    {
        public string id;
        public string op_id;
        public string is_id;
        public double op_suresi;
        //public string mak_grup_id;
        public List<string> mak_grup_ids;
        public List<int> makinekullanımoranları;
        //public List<string> makine_ids;
        public List<string> onsart_ids;
        public List<string> onsart_ids_temp; //count eşit olana kadar serbest bırakmayacağız
        public double baslangıc_zamanı;
        public double bitis_zamanı;
        public string atanan_makine;
        public List<string> atanan_makineler;
        public double en_erken_baslama_zamanı;
        //liste tipi olanları initialize ediyoruz hata almamak ve kontrol etmek için
        public Operation()
        {
            id = "-1";
            op_id = "-1";
            is_id = "-1";
            op_suresi = -1;
            //mak_grup_id="-1";
            mak_grup_ids = new List<string>();
            makinekullanımoranları = new List<int>();
```

```
            //makine_ids = new List<string>();
            onsart_ids = new List<string>();
            onsart_ids_temp = new List<string>();
            baslangıc_zamanı = -1;
            bitis_zamanı = -1;
            atanan_makine = "-1";
            atanan_makineler = new List<string>();
            en_erken_baslama_zamanı = -1;
        }
        public void op_atama()
        {
            string atanabilecek_makine = "-1";
            double atanabilecek_baslama_zamanı = 0;
            double opt_baslama_zamani = double.MaxValue;
            bool conti = true;

            if (onsart_ids.Count != 0) en_erken_baslama_zamanı = Data.operations.FindAll(x => onsart_ids.Contains(x.id)).Select(x => x.bitis_zamanı).ToList().Max();
            else en_erken_baslama_zamanı = 0;

            for (int i = 0; i < mak_grup_ids.Count; i++)
            {
                List<string> alt_mak = new List<string>();
                alt_mak = Data.makine_grupları.Find(x => x.mak_grup_id == mak_grup_ids[i]).makine_ids;
                atanabilecek_makine = "-1";
                opt_baslama_zamani = double.MaxValue;
                foreach (var mach_item in alt_mak)
                {
                    var atanacak_makine = Data.machines.Find(x => x.machine_id == mach_item);
```

```
            Bosluk mevcut_bosluk = atanacak_makine.makine_ilk_bosluk;
            conti = true;
            while (!mevcut_bosluk.atama(this, mach_item, conti, ref atanabilecek_baslama_zamanı, ref opt_baslama_zamani, ref atanabilecek_makine) && mevcut_bosluk.sonraki_bosluk != null)
            {
               mevcut_bosluk = mevcut_bosluk.sonraki_bosluk;
            }
         }
         if (atanabilecek_makine != "-1")
         {
            if (i == 0)
            {
               en_erken_baslama_zamanı = atanabilecek_baslama_zamanı;
               atanan_makineler.Add(atanabilecek_makine);
            }
            else
            {
               if (en_erken_baslama_zamanı == atanabilecek_baslama_zamanı)
               {
                  atanan_makineler.Add(atanabilecek_makine);
               }
               else
               {
                  atanan_makineler.Clear();
                  i = -1;
                  en_erken_baslama_zamanı = atanabilecek_baslama_zamanı;
               }
            }
         }
         else
         {
```

```
                atanan_makineler.Clear();
                i = -1;
                en_erken_baslama_zamanı = atanabilecek_baslama_zamanı;
            }
        }
        var mak2 = Data.machines.FindAll(x => atanan_makineler.Contains(x.machine_id));
        foreach (var item in mak2)
        {
            conti = false;
            Bosluk mevcut_bosluk2 = Data.machines.Find(x => x.machine_id == item.machine_id).makine_ilk_bosluk;
            while (!mevcut_bosluk2.atama(this, item.machine_id, conti, ref atanabilecek_baslama_zamanı, ref opt_baslama_zamani, ref item.machine_id) && mevcut_bosluk2.sonraki_bosluk != null)
            {
                mevcut_bosluk2 = mevcut_bosluk2.sonraki_bosluk;
            }
        }
    }

    public void op_atama2()
    {
        string atanabilecek_makine = "-1";
        double atanabilecek_baslama_zamanı = 0;
        double opt_baslama_zamani = double.MaxValue;
        bool conti = false;

        if (onsart_ids.Count != 0) en_erken_baslama_zamanı = Data.operations.FindAll(x => onsart_ids.Contains(x.id)).Select(x => x.bitis_zamanı).ToList().Max();
        else en_erken_baslama_zamanı = 0;
```

```
var en_erken_baslama_zamanı2 = en_erken_baslama_zamanı;

for (int i = 0; i < mak_grup_ids.Count; i++)
{
   List<string> alt_mak = new List<string>();
   alt_mak = Data_2.makine_grupları.Find(x => x.mak_grup_id == mak_grup_ids[i]).makine_ids;
   atanabilecek_makine = "-1";
   atanabilecek_baslama_zamanı = 0;
   opt_baslama_zamani = double.MaxValue;
   foreach (var mach_item in alt_mak)
   {
      var mak = Data_2.machines.Find(x => x.machine_id == mach_item);
      conti = false;
      for (int j = 0; j < mak.bosluk_listesi.Count; j++)
      {
         if (j < en_erken_baslama_zamanı) continue;
         if (atanabilecek_baslama_zamanı <= j && atanabilecek_baslama_zamanı > 0) break;

         if (mak.bosluk_listesi[j] == 100)
         {
            for (int k = j; k < j + op_suresi; k++)
            {
               if (mak.bosluk_listesi[k] == 100)
               {
                  conti = true;
               }
               else
               {
                  conti = false;
                  break;
```

```
                }
            }
        }
        if (conti == true)
        {
            atanabilecek_baslama_zamanı = j;
            break;
        }
    }
    if (conti == true && atanabilecek_baslama_zamanı < opt_baslama_zamani)
    {
        opt_baslama_zamani = atanabilecek_baslama_zamanı;
        atanabilecek_makine = mach_item;
    }
}
if (atanabilecek_makine != "-1")
{
    if (i == 0)
    {
        en_erken_baslama_zamanı = atanabilecek_baslama_zamanı;
        atanan_makineler.Add(atanabilecek_makine);
    }
    else
    {
        if (en_erken_baslama_zamanı == atanabilecek_baslama_zamanı)
        {
            atanan_makineler.Add(atanabilecek_makine);
        }
        else
        {
            atanan_makineler.Clear();
            i = -1;
```

```
                    en_erken_baslama_zamanı = atanabilecek_baslama_zamanı;
                }
            }
        }
        else
        {
            atanan_makineler.Clear();
            i = -1;
            en_erken_baslama_zamanı = atanabilecek_baslama_zamanı;
        }
    }
    var mak2 = Data_2.machines.FindAll(x => atanan_makineler.Contains(x.machine_id));
    baslangıc_zamanı = opt_baslama_zamani;
    bitis_zamanı = opt_baslama_zamani + op_suresi;
    foreach (var item in mak2)
    {
        for (int k = (int)atanabilecek_baslama_zamanı; k < (int)atanabilecek_baslama_zamanı + op_suresi; k++)
        {
            item.bosluk_listesi[k] = 0;
        }
    }
  }
}

class Machine
{
  public string machine_id;
  public Bosluk makine_ilk_bosluk;
}
class Machine2
```

```
{
    public string machine_id;
    public List<int> bosluk_listesi = new List<int>();//2.boşluk tutma stili
}

class Bosluk
{
    public double bosluk_suresi;
    public double bosluk_baslangıc_zamanı;
    public Bosluk önceki_bosluk;
    public Bosluk sonraki_bosluk;

    public bool atama(Operation atanacak_op, string atanacak_mak, bool conti, ref
double atanabilecek_baslama_zamanı, ref double opt_baslama_zamani, ref string
atanabilecek_makine)
    {
        //Operasyonun ön şartıyla ilgili başlangıç zamanı ayarlamaları daha önceden
yapıldığı varsayılır.
        bool result = false;
        //4 atama alternatifinden
        double geçici_baslangıç_zamanı;
        geçici_baslangıç_zamanı = Math.Max(atanacak_op.en_erken_baslama_zamanı,
bosluk_baslangıc_zamanı);

        if (geçici_baslangıç_zamanı + atanacak_op.op_suresi <=
bosluk_baslangıc_zamanı + bosluk_suresi)
        {
            result = true;
            if (conti == false)
            {
                //1-2 ilk if---başlangıçları=bosluk başlangıç zamanı
```

```
            //!-1den başlangıç zamanı olması sıfırdan küçük ya sorun olur mu acaba
önşartlarla bak
            if (geçici_baslangıç_zamanı <= bosluk_baslangıc_zamanı)
            {
               if (atanacak_op.op_suresi == bosluk_suresi)
               {
                  //SENARYO 1: op_süresi ile boşluk süresi aynıysa ona ata
                  if (önceki_bosluk == null)
                  {
                     atanacak_op.baslangıc_zamanı = bosluk_baslangıc_zamanı;
                     atanacak_op.bitis_zamanı = atanacak_op.baslangıc_zamanı +
atanacak_op.op_suresi;
                     atanacak_op.atanan_makine = atanacak_mak;
                     sonraki_bosluk.önceki_bosluk = null;
                  }
                  else
                  {
                     atanacak_op.baslangıc_zamanı = bosluk_baslangıc_zamanı;
                     atanacak_op.bitis_zamanı = atanacak_op.baslangıc_zamanı +
atanacak_op.op_suresi;
                     atanacak_op.atanan_makine = atanacak_mak;
                     önceki_bosluk.sonraki_bosluk = sonraki_bosluk;
                     sonraki_bosluk.önceki_bosluk = önceki_bosluk;
                  }
               }
               else
               {
                  //SENARYO 2: operasyonun en baştan atandığı sonda boşluk kaldığı
durum
                  atanacak_op.baslangıc_zamanı = bosluk_baslangıc_zamanı;
                  atanacak_op.bitis_zamanı = atanacak_op.baslangıc_zamanı +
atanacak_op.op_suresi;
```

```
                atanacak_op.atanan_makine = atanacak_mak;
                bosluk_baslangıc_zamanı = geçici_baslangıç_zamanı + atanacak_op.op_suresi;
                bosluk_suresi = bosluk_suresi - atanacak_op.op_suresi;
            }
        }
        else
        {
            if ((geçici_baslangıç_zamanı + atanacak_op.op_suresi < bosluk_baslangıc_zamanı + bosluk_suresi))
            {
                //değişkeni hafızadan düşürmeyi hoca anlatıcak**internetten arat kill a variable in c#, how to drop variable from memory
                //trash control //public silinmiyor
                ////SENARYO 3: boşluğun arasında bir yere işin atandığı(başa ve sona değmiyor)
                atanacak_op.baslangıc_zamanı = geçici_baslangıç_zamanı;
                atanacak_op.bitis_zamanı = geçici_baslangıç_zamanı + atanacak_op.op_suresi;
                atanacak_op.atanan_makine = atanacak_mak;

                Bosluk Yeni_Bosluk = new Bosluk();
                Yeni_Bosluk.sonraki_bosluk = sonraki_bosluk;
                Yeni_Bosluk.önceki_bosluk = this;//*****
                Yeni_Bosluk.bosluk_baslangıc_zamanı = geçici_baslangıç_zamanı + atanacak_op.op_suresi;
                Yeni_Bosluk.bosluk_suresi = bosluk_baslangıc_zamanı + bosluk_suresi - (geçici_baslangıç_zamanı + atanacak_op.op_suresi);

                sonraki_bosluk = Yeni_Bosluk;
                bosluk_suresi = geçici_baslangıç_zamanı - bosluk_baslangıc_zamanı;
            }
```

```
                else
                {
                    //bosluk_suresi = (geçici_baslangıç_zamanı - bosluk_baslangıc_zamanı);
                    ////SENARYO 4: operasyonun bitişinin boşluğun sonuna değdiği
                    atanacak_op.baslangıc_zamanı = geçici_baslangıç_zamanı;
                    //atanacak_op.baslangıc_zamanı = bosluk_baslangıc_zamanı + bosluk_suresi - atanacak_op.op_suresi;
                    atanacak_op.bitis_zamanı = bosluk_baslangıc_zamanı + bosluk_suresi;
                    atanacak_op.atanan_makine = atanacak_mak;
                    bosluk_suresi = bosluk_suresi - atanacak_op.op_suresi;
                }
            }
        }
        else
        {
            atanabilecek_baslama_zamanı = geçici_baslangıç_zamanı;
            if (atanabilecek_baslama_zamanı < opt_baslama_zamani)
            {
                opt_baslama_zamani = atanabilecek_baslama_zamanı;
                atanabilecek_makine = atanacak_mak;
            }

        }
    }
    return result;
  }
}
```

Program.cs

```
static int is_sayisi = 100;
static int max_operasyon_sayisi = 3;
```

```
static int makine_sayisi = 50;
static int makine_grup_sayısı = 1;
static Bosluk ilk_bosluk = new Bosluk();
static TimeSpan gecenSure;
int i = 1; // üretilen veri setinin benzer örnekteki kaçıncı problem olduğunu gösterir
        var e1_path = @"filepath\\datam1veri_j" + is_sayisi.ToString() + "o" + max_operasyon_sayisi.ToString() + "m" + makine_sayisi.ToString() + "g" + makine_grup_sayısı.ToString() + "i" + i.ToString() + ".json";
        var e2_path = @"filepath\\datam2veri_j" + is_sayisi.ToString() + "o" + max_operasyon_sayisi.ToString() + "m" + makine_sayisi.ToString() + "g" + makine_grup_sayısı.ToString() + "i" + i.ToString() + ".json";

        Data2 datam = JsonConvert.DeserializeObject<Data2>(File.ReadAllText(e1_path));
        Data.operations = datam.operations;
        Data.machines = datam.machines;
        Data.makine_grupları = datam.makine_grupları;

        Data2_2 datam2 = JsonConvert.DeserializeObject<Data2_2>(File.ReadAllText(e2_path));
        Data_2.operations = datam2.operations;
        Data_2.machines = datam2.machines;
        Data_2.makine_grupları = datam2.makine_grupları;

        Stopwatch stopWatch = new Stopwatch();
        if (!stopWatch.IsRunning)
           stopWatch.Restart();
        stopWatch.Start();
        Data.is_atama();
        stopWatch.Stop();

        gecenSure = stopWatch.Elapsed;
```

```
        Console.Out.WriteLine("Calisma Suresi=" + "\t" + gecenSure.Hours.ToString()
+ " saat " + gecenSure.Minutes.ToString() + " dakika " + gecenSure.Seconds.ToString()
+ " saniye " + gecenSure.Milliseconds.ToString() + " salise = " +
gecenSure.TotalMilliseconds.ToString());
        Console.Out.WriteLine("Calisma Suresi=" + "\t" +
Data.op_atama_suresi.Hours.ToString() + " saat " +
Data.op_atama_suresi.Minutes.ToString() + " dakika " +
Data.op_atama_suresi.Seconds.ToString() + " saniye " +
Data.op_atama_suresi.Milliseconds.ToString() + " salise = " +
Data.op_atama_suresi.TotalMilliseconds.ToString());

        stopWatch.Reset();
        stopWatch.Start();
        Data_2.is_atama2();
        stopWatch.Stop();

        gecenSure = stopWatch.Elapsed;
        Console.Out.WriteLine("Calisma Suresi=" + "\t" + gecenSure.Hours.ToString()
+ " saat " + gecenSure.Minutes.ToString() + " dakika " + gecenSure.Seconds.ToString()
+ " saniye " + gecenSure.Milliseconds.ToString() + " salise = " +
gecenSure.TotalMilliseconds.ToString());
        Console.Out.WriteLine("Calisma Suresi=" + "\t" +
Data_2.op_atama_suresi.Hours.ToString() + " saat " +
Data_2.op_atama_suresi.Minutes.ToString() + " dakika " +
Data_2.op_atama_suresi.Seconds.ToString() + " saniye " +
Data_2.op_atama_suresi.Milliseconds.ToString() + " salise = " +
Data_2.op_atama_suresi.TotalMilliseconds.ToString());

        Console.Out.WriteLine("İş No" + "\t" + "Op No" + "\t" + "Makine" + "\t" +
"Basl." + "\t" + "Bit." + "\t" + "Operasyon Süresi");
        foreach (var op_item in Data.operations)
        {
```

```
            Console.Out.WriteLine(op_item.is_id + "\t" + op_item.op_id + "\t" +
op_item.atanan_makine + "\t" + op_item.baslangıc_zamanı + "\t" + op_item.bitis_zamanı
+ "\t" + op_item.op_suresi);
        }
        Console.Out.WriteLine(Environment.NewLine + "OK");
        Console.Out.WriteLine("İş No" + "\t" + "Op No" + "\t" + "Makine" + "\t" +
"Basl." + "\t" + "Bit." + "\t" + "Operasyon Süresi");
        foreach (var op_item in Data_2.operations)
        {
            Console.Out.WriteLine(op_item.is_id + "\t" + op_item.op_id + "\t" +
op_item.atanan_makine + "\t" + op_item.baslangıc_zamanı + "\t" + op_item.bitis_zamanı
+ "\t" + op_item.op_suresi);
        }

        //VERİ ÇIKTISI YAZDIRMA
        var path1 = @" filepath \\senaryo2_datam1veri_j" + is_sayisi.ToString() + "o" +
max_operasyon_sayisi.ToString() + "m" + makine_sayisi.ToString() + "g" +
makine_grup_sayısı.ToString() + "i" + i.ToString() + ".json";
        var path2 = @"filepath\\senaryo2_datam2veri_j" + is_sayisi.ToString() + "o" +
max_operasyon_sayisi.ToString() + "m" + makine_sayisi.ToString() + "g" +
makine_grup_sayısı.ToString() + "i" + i.ToString() + ".json";

        Data2_2 w_datam2 = new Data2_2();
        w_datam2.kopyala();
        File.WriteAllText(path2, JsonConvert.SerializeObject(w_datam2));

        foreach (var mak_item in Data.machines)
        {
            mak_item.makine_ilk_bosluk = null;
        }
        Data2 w_datam = new Data2();
        w_datam.kopyala();
```

```
File.WriteAllText(path1, JsonConvert.SerializeObject(w_datam));
Console.Out.WriteLine(i.ToString() + ". çıktılar yazdırıldı");

Console.ReadLine();
```

ÖZGEÇMİŞ

Adı Soyadı : Beray BAYAZIT
Doğum Yeri ve Tarihi : Bursa, 24.07.1993
Yabancı Dil : İngilizce

Eğitim Durumu
Lise : Turhan Tayan Anadolu Lisesi 2011
Lisans : Uludağ Üniversitesi 2017

Çalıştığı Kurum/Kurumlar : Uludağ Üniversitesi Müh. Fak. 2018-2019
Biteg Yazılım Ltd.Şti. 2021-…

İletişim (e-posta) : beraybayazit@hotmail.com

Yayınları : Gençosman, B., Bayazıt B., Beğen, M., Uçarkuş G. 2019. Raf Alanı Tahsisi ve Sergileme Problemi çözümünde Genetik Algoritma. Yöneylem Araştırması ve Endüstri Mühendisliği 39. Ulusal Kongresi (YAEM 2019), Ankara.