
GAZİ ÜNİVERSİTESİ
Gazi
1926

ATÖLYE TİPİ ÇİZELGELEME PROBLEMİNİN GENETİK ALGORİTMA İLE ÇÖZÜMÜNDE MONOLOTİK VE MİKROSERVİS MİMARİLERİNİN KARŞILAŞTIRMALI ANALİZİ VE PERFORMANS OPTİMİZASYONU

Uğur KONAR

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEMMUZ 2023

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Uğur KONAR
20/07/2023

ATÖLYE TİPİ ÇİZELGELEME PROBLEMİNİN GENETİK ALGORİTMA İLE ÇÖZÜMÜNDE MONOLOTİK VE MİKROSERVİS MİMARİLERİNİN KARŞILAŞTIRMALI ANALİZİ VE PERFORMANS OPTİMİZASYONU

(Yüksek Lisans Tezi)

Uğur KONAR

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Temmuz 2023

ÖZET

Bu çalışmanın amacı, Atölye Tipi Çizelgeleme Problemi(ATÇP)' nin çözümü için genetik algoritma temelli monolotik ve mikro servis mimarilerinin performansını analiz etmek, zamankarmaşıklığını azaltma potansiyellerini değerlendirmek, avantajları ve sınırlılıkları ortaya koymaktır. Monolitik bir uygulama tasarlanarak zaman karmaşıklığının azaltılması için farklı teknikler denenmiş ve bu tekniklerin etkisi karşılaştırılmıştır. Ardından, genetik algoritma temelli çözüm mikro servis mimarisi kullanılarak ölçeklendirilmiş ve performans analizi yapılmıştır. Her iki mimarinin performansı, genetik algoritmanın zaman karmaşıklığını azaltma becerisine odaklanarak karşılaştırılmıştır. Sonuçlar, mikro servis mimarisinin genetik algoritmanın zaman karmaşıklığını azaltmada farklılıklar sağladığını ve bu yaklaşımların ölçeklenebilirlik ve modülerlik açısından avantajları ve sınırlılıklarını ortaya koymuştur. Ayrıca, genetik algoritmaların zaman karmaşıklığını azaltma konusunda gelecekte yapılacak araştırmalar için öneriler sunulmuştur.

Bilim Kodu	:	92408
Anahtar Kelimeler	:	Meta Sezgisel, Atölye Tipi Çizelgeleme Problemi, Genetik Algoritmalar, zaman karmaşıklığı, mikroservis mimari
Sayfa Adedi	:	52
Danışman	:	Prof. Dr. Aydın ÇETİN

COMPARATIVE ANALYSIS AND PERFORMANCE OPTIMIZATION OF SOLVING THE JOB SHOP SCHEDULING PROBLEM USING GENETIC ALGORITHM WITH MONOLITHIC AND MICROSERVICE ARCHITECTURES

(M. Sc. Thesis)

Uğur KONAR

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

July 2023

ABSTRACT

The aim of this study is to analyze the performance of genetic algorithm-based monolithic and microservice architectures for solving the Job-Shop Scheduling Problem, evaluate their potential for reduce time complexity, and highlight their advantages and limitations. Different techniques have been tried and the effects of these techniques have been compared in order to reduce time complexity by designing a monolithic application. Subsequently, the genetic algorithm-based solution was scaled using a microservice architecture, and a performance analysis was conducted. The performance of both architectures was compared focusing on the genetic algorithm's ability to reduce time complexity. The results revealed differences in reducing the time complexity of the Job-Shop Scheduling Problem using the microservice architecture compared to the monolithic approach, and highlighted the advantages and limitations of these approaches in terms of scalability and modularity. Additionally, suggestions were provided for future research on reducing the time complexity of genetic algorithms.

Science Code	:	92408
Key Words	:	Metaheuristic Algorithms, Job-Shop Scheduling Problem, Genetic Algorithms, time complexity, mikroservice architecture
Page Number	:	52
Supervisor	:	Prof. Dr. Aydın ÇETİN

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren kıymetli hocam Prof. Dr. Aydın ÇETİN'e, bilgi ve tecrübelerinden faydalandığım iş arkadaşlarım AhmetSamet HALICI ve Barış SATAR'a teşekkür ederim. Tüm eğitim hayatım boyunca maddi ve manevi desteklerini hiçbir zaman esirgemeyen aileme, hayatın her alanında olduğu gibi bu tez çalışmasında da desteğini hiçbir zaman esirgemeyen sevgili eşim İlknur KONAR'a çok teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklamalar**
gb	Gigabayt
ms	milisaniye

Kısaltmalar	**Açıklamalar**
AMQP	İleri Mesaj Kuyruğu Protokolü (Advanced Message Queuing Protocol)
API	Uygulama Programlama Arayüzü (Application Programming Interface)
HTTP	Hiper Metin Transfer Protokolü (Hyper-Text Transfer Protokol)
JDK	Java geliştirme kiti (Java Development Kit)
JSON	Javascript nesne notasyonu
KİÇ	Kalan işlerin çoğu (Most Work Remaining)
KİS	Kalan İşlemlerin Sayısı
NP	Non-Deterministic Polinomial (Belirleyici Olmayan Polinom
ÖKSDÇ	Öncelik Koruma Sırasına Dayalı Çaprazlama (Precendence Preserving Order-based Cross-over)
ÖKM	Öncelik Korumalı Mutasyon (Precendence Preserving Shift Mutation) (Most number of OperationsRemaining)
XML	Genişletilebilir İşaretleme Dili

1. GİRİŞ

Son yıllarda meta sezgisel algoritmalar birçok karmaşık optimizasyon probleminin çözümü için kullanılmaktadır. Meta sezgisel algoritmaların önemli problemlerinden biri de hesaplama maliyetidir. Çünkü meta sezgisel algoritmalar birçok farklı alanda karmaşık optimizasyon problemlerini çözüm bulurken hızlı sonuç üretememektedir. Bu soruna doğru bir yaklaşımla sistematik bir çözüm bulunduğu takdirde sistem tanılama, kontrol sistemleri, robot uygulamaları, görüntü ve ses tanıma, mühendislik tasarımları, planlama, yapay zekâ uygulamaları, uzman sistemler, fonksiyon ve kombinasyonel eniyileme problemleri ağ tasarım problemleri, yol bulma problemleri, çizelgeleme problemleri, sosyal ve ekonomik planlama problemleri gibi birçok farklı probleme daha hızlı çözümler bulunarak buradaki hesaplama maliyeti minimuma indirilecektir.

Meta sezgisel algoritmalar içerisinde Genetik algoritmalar (GA) çok önemli bir yer tutmaktadır. GA global optimizasyon problemlerini çözmek için doğal evrim süreçlerini modelleyen popülasyon temelli rastlantısal ve değişken arama algoritmasıdır. Bu tez çalışmasının amacı, ATÇP'nin GA temelli monolit ve mikro servis mimarilerin performansını analiz etmek, bu süreçte zaman karmaşıklığını azaltma potansiyellerini değerlendirmek, bu yaklaşımların avantajlarını ve sınırlılıklarını ortaya koymaktır. Böylece, daha etkili ve hızlı çözümlerin elde edilmesine ve optimizasyon problemlerinde daha bilinçli mimari seçimler yapılmasına katkıda bulunulması amaçlanmaktadır. Bu bağlamda, tez çalışması aşağıdaki üç temel araştırma sorusuna yanıt aramaktadır:

- ATÇP'nin genetik algoritma ile çözümünde algoritmanın çalışma performansının azaltılabilmesi için hangi teknikler kullanılabilir ve bu tekniklerin kullanımı algoritmanın çalışma performansında önemli farklılıklar oluşturabilir mi?
- ATÇP'nin çözümünde, genetik algoritmalar kullanarak monolit ve mikro servis mimarilerinin performansları arasında önemli farklar var mıdır ve bu farklar hangi faktörlere bağlıdır?
- Her iki mimari yaklaşımın ölçeklenebilirlik, modülerlik, hız, esneklik, dağıtım ve enerji verimliliği açısından avantajları ve sınırlılıkları nelerdir ve hangi durumlarda hangi mimari tercih edilmelidir?

Bu tez yukarıda sorulan sorulara yanıt vererek ATÇP'nin GA temelli çözümleriçin en uygun mimari seçimine yönelik bilgi birikimine katkıda bulunmayı ve ATÇP'nin çözümünde genetik algoritmalar (GA) kullanarak monolitik ve mikro servis mimarilerinin performanslarını kapsamlı ve detaylı bir şekilde karşılaştırarak analiz etmeyi amaçlamaktadır. Monolit yapıda GA'nın çalışma performansını optimize etmek için farklı teknikler denenmiştir. Monolit yapıda çalışma performansının azaltılması için elde edilen sonuçları daha da ileriye götürebilmek amacıyla aynı problem mikro servis mimarisi içinde tasarlanarak ölçeklendirilmiştir. Algoritmanın her bir aşaması ayrı bir mikro servis olarak değerlendirilerek bu servisler ölçeklendirilerek paralel bir şekilde çalıştırılmıştır. Böylece algoritmanın zaman karmaşıklığının minimize edilmesi amaçlanmıştır.

Ayrıca, bu çalışma GA'nın parametre optimizasyonu ve seçimi konusundaki mevcut anlayışı geliştirerek, optimizasyon problemlerinde daha etkili ve hızlı çözümlerin elde edilmesine ve genetik algoritmaların daha geniş kapsamlı uygulama alanlarında kullanılmasına katkı sağlamayı hedeflemektedir. Tez çalışması, ATÇP çözümünde monolit ve mikro servis mimarilerinin kapsamlı ve derinlemesine analizi ile alanda mevcut bilgi birikimine önemli ölçüde katkı sağlamayı hedeflemektedir. Bu bağlamda, çalışmada sunulan analizler ve bulgular, üretim süreçlerinin planlanması ve optimizasyonu gibi önemli uygulama alanlarında GA temelli çözümlerin etkinliğini ve verimliliğini artırmak için faydalı olacaktır.

Bu tez 6 bölüm halinde sunulmuştur. Bu bölümde, araştırmanın temel problem ve amacını tanımlamakta, araştırma sorularını belirtmekte ve tez çalışmasının genel yapısını sunmaktadır. 2. bölümde literatürde ATÇP'ninçözümü için kullanılan yaklaşımlar incelenmiş ve kullanılan genetik algoritma çözümlerinde performans iyileştirmesi için kullanılan metotlar analiz edilmiştir. 3. Bölümde GA, ATÇP ve mikro servis mimarisi ile ilgili genel bilgiler verilmiştir. Kullanılan metaryeller açıklanmış ve problemin çözüm metodundaki yeri açıklanmıştır. 4. bölümde ATÇP çözümü için GA temelli monolitik bir uygulama tasarlamayı ve uygulamanın performansını analiz etmeyi içermektedir. Algoritmanın parametreleri ve işlemleri detaylı olarak incelenmekte ve verilen senaryo üzerinde uygulamanın performansı değerlendirilmektedir. ATÇP çözümü için GA temelli mikro servis mimarisi kullanılarak ölçeklendirilmiş bir uygulama tasarlanmıştır. Tasarlanan bu yaklaşım detaylı bir şekilde anlatılmıştır. 5. bölümde ise monolit ve mikro servis

mimarilerinin performanslarını karşılaştırarak, her iki yaklaşım için deneysel sonuçlar ortaya konmuştur. 6. bölümde ise GA'nın zaman karmaşıklığını azaltma potansiyeli, parametre optimizasyonu ve seçimi konusundaki stratejilerin etkileri değerlendirilmektedir. Böylece, ATÇP için hangi mimarinin daha uygun olduğu ve hangi durumlarda hangi yaklaşımın tercih edileceği ile ilgili bilgiler sunulmaktadır. Tez çalışmasının bulguları ve sonuçları tartışılarak, mevcut literatüre katkılar ve çalışmanın sınırlılıkları belirtilmektedir. Ayrıca, gelecekte yapılacak araştırmalar ve geliştirilecek algoritma stratejileri için öneriler sunulmaktadır.

2. ATÖLYE TİPİ ÇİZELGELEME PROBLEMİ

ATÇP üretim süreçlerinin planlanması ve optimizasyonu ile ilgili karmaşık ve önemli bir optimizasyon problemidir. Problemin temel amacı, belirli işlerin ve bu işlere atanmış işlem sürelerinin olduğu bir iş atölyesinde, işlerin belirli makinelerde işlenmesi süreçlerini en iyi şekilde planlayarak, üretimin tamamlanma süresini minimize etmektir. ATÇP, NP-Zor sınıfında bir problem olarak kabul edilir ve bu nedenle en uygun çözümlerini elde etmek oldukça zor ve zaman alıcıdır.

2.1. Problem Tanımı

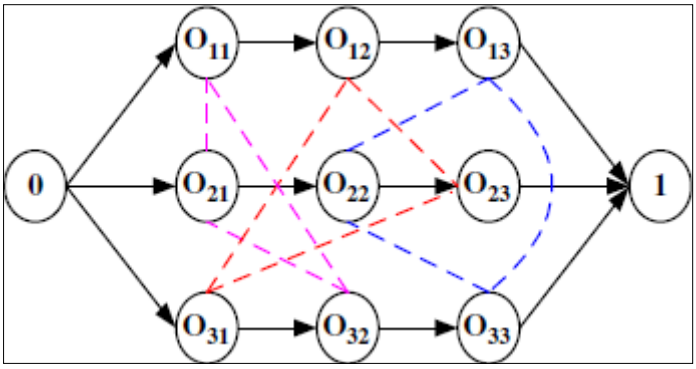
ATÇP'nin önemi, üretim süreçlerinin planlanması ve optimizasyonu için etkili yöntemler geliştirmek ve daha verimli üretim süreçleri elde etmekle ilişkilidir. ATÇP çözümleri, üretim planlaması, stok kontrolü, lojistik ve iş gücü yönetimi gibi alanlarda önemli roller oynamaktadır. Bu nedenle, ATÇP çözümlerinin geliştirilmesi, işletmelerin rekabet gücünü ve verimliliğini artırarak, kaynak kullanımını optimize etmeye ve maliyetleri azaltmaya yardımcı olmaktadır [1].

Atölye tipi çizelgeleme problemi temelde n sayıda farklı işin m tane makine üzerinde belirli kısıtları sağlayacak şekilde çalışmasını tanımlamaktadır. Her işin m tane operasyondan oluştuğu varsayılsın, bu durumda problemin çözümü aşağıdaki kısıtları karşılamak zorundadır. Bu kısıtlar aşağıdaki gibi tanımlanabilir:

i. Bir iş aynı makinede sadece bir kez çalıştırabilir.
ii. Farklı işlerin operasyonları arasında herhangi bir sıralama söz konusu değildir.
iii. Herhangi bir operasyonun çalışmamasına izin verilmez
iv. Her iş belirli bir anda yalnızca bir makinede çalışabilir
v. Her işin operasyonları kendi içerisinde verilen zaman sıralamasına göre çalışmak zorundadır.

Bu problemin çözümünde bütün makinelerdeki çalışma zamanının minimize edilmesi amaçlanmaktadır. Atölye tipi çizelgele problemini Şekil 2.1'de gösterildiği gibi bir graf modeli içerisinde tanımlanabilir. G=(N,A,E) şeklinde bir graf tanımlanırsa N problem içerisinde tanımlanan işlerin kapsadığı operasyonlar, A aynı işin bu operasyonlarının

birbirine bağlıolduğu komşu düğümler, E ise aynı makine içerisinde birbirine bağlı olmayan operasyon düğümlerini göstermektedir. Graf içerisindeki eşitlikler Eş. 1 – Eş. 3 ile gösterilir. Eş. 1'de i. İşin operasyonunu göstermektedir., Eş. 2'de her bir Ej tekrar etmeyen düğümlerin birleşiminden oluşmaktadır. Eş. 3'te birbirine bağımlı komşu düğümlerin birleşimi gösterilmektedir.



Şekil 2.1. Atölye tipi çizelgeleme probleminin tanımlanan graf modeli [2].

Matematiksel eşitlikler;

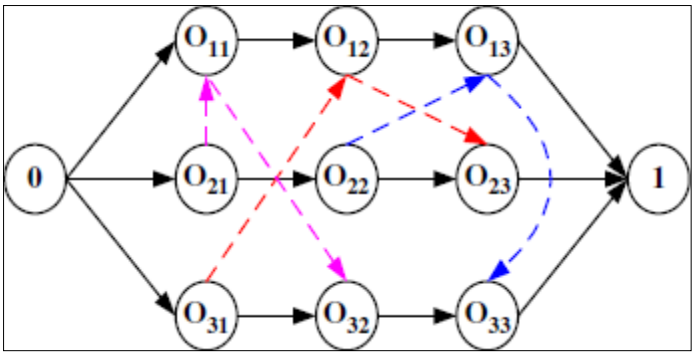
$$G=(N,A,E)\ N=\{0,1,\ O_{ij}\ ,\ İ=1,2,\ldots\ldots n;\ j=1,2,\ldots\ldots m|\} \quad (1)$$

$$E_j\ =\{j\ =\ 1,2,\ldots..m\} \quad (2)$$

$$A=\{(0,\ O_{i}1)|\ i=1,2,\ldots\ldots n\}\ U\{(O_{ij}\ ,\ O_{ij}\ +1)\ |\ i=1,2,\ldots..,m\text{-}1\}\ U\ \{(O_{im},1)i=1,2,\ldots..n\} \quad (3)$$

şeklinde ifade edilir.

Atölye tipi çizelgeleme probleminin tanımlanan graf modelinde belirtilen bütün kısıtların sağlandığı çözüm Şekil 2.2'de gösterilmiştir.



Şekil 2.2. Atölye tipi çizelgeleme probleminin tanımlanan graf modelinde belirtilen bütün kısıtların sağlandığı çözüm [2].

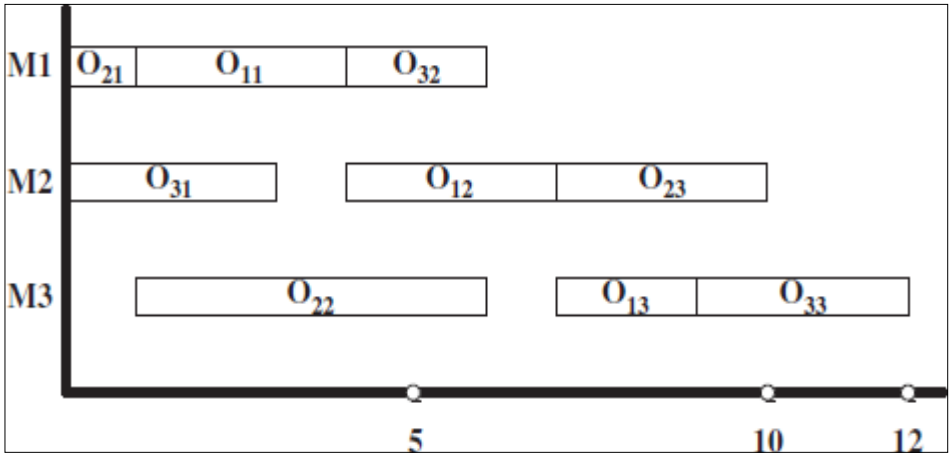
Makinelerde çalışacak işleri oluşturan operasyonların oluşturulduğu grafın düğümü olarak tanımlandığında ATÇP'nin bütün kısıtlarını karşılayan çözümü tek yönlü graf içerisinde ifade edilir. Şekil 2.2'de gösterildiği gibi her işin operasyonlarının numaralandırmalarına göre sıralı çalışması gerekmektedir. $O32$ operasyonu başladığında $O31$ operasyonunun tamamlanmış olması gerekmektedir. Farklı renklerle tanımlanan akışlar her makinedeki iş sıralamasını göstermektedir. Bir iş aynı makinede yalnızca bir kere çalıştırılabilir kısıttı da göz önüne alındığında bütün kısıtları karşılayan optimum sonuca ulaşılmıştır.

Atölye tipi çizelgeleme problemi matematiksel model ile açıklandığı gibi gerçek bir örneklem üzerinden de sonuca ulaşılması gerekir. Çizelge 2.1 'de belirlenen makinelerde verilen işlere bağlı operasyonların ne kadar süre çalışacağı tanımlanmıştır.

Çizelge 2.1. Minimum işlem zamanıyla belirtilen kısıtlara göre üretilen optimum sonuç örneği

	İşler	Operasyonlar		
		1	2	3
Çalışma Süresi	J1	3	3	2
	J2	1	5	3
	J3	3	2	3
Makine Sırası	J1	M1	M2	M3
	J2	M1	M3	M2
	J3	M2	M1	M3

Bu tanıma göre tasarlanan algoritma Şekil 2.3' te tanımlandığı gibi optimum sonucu üretmelidir. Burada bütün operasyonların kısıtları sağlamasının yanı sıra makinelerdeki toplam çalışma süresinin de minimum zaman alması gerekmektedir



Şekil 2.3. İşlerin zamanları, sıralamaları ve çalışmaları gereken makinelerin gösterim örneği

Bu tezde, ATÇP çözümünde genetik algoritmaların kullanılmasına odaklanılmıştır. Genetik algoritmalar, doğal seleksiyon ve genetik süreçlere dayanan bir meta-sezgisel algoritma türüdür. Bu algoritmalar, çözüm adaylarından oluşan bir popülasyon üzerinde çaprazlama, mutasyon ve seçilim gibi işlemler gerçekleştirerek, nesiller boyunca en iyi çözümleri evrimleştirmeye çalışır.

Genetik algoritmaların ATÇP çözümünde başarılı bir şekilde kullanılabilmesi için, uygun kodlama yöntemleri, seçilim işlemleri ve çaprazlama ve mutasyon stratejileri geliştirilmelidir.

2.2. Literatürdeki Yaklaşımlar

ATÇP'nin tanımı ve önemi, üretim süreçlerinin planlanması ve optimizasyonu için etkili yöntemler geliştirme ihtiyacının ve bu yöntemlerin işletmelerin rekabet gücünü ve verimliliğini artırma potansiyelinin altını çizmektedir. Problemin çözümü için birçok yöntem geliştirilmiştir. Bunlar arasında kesirli programlama, dinamik programlama, dal ve sınır, sezgisel algoritmalar ve meta-sezgisel algoritmalar (örneğin, genetik algoritmalar, tabu arama, benzetimli tavlama) gibi çeşitli yöntemler bulunmaktadır [3]. Bu çalışma,

ATÇP' nin çözümünde genetik algoritmaların kullanılmasına ve bu algoritmaların monolitik ve mikro servis mimarileri arasındaki performans karşılaştırmasına odaklanmaktadır.ATÇP çözümü için geliştirilmiş birçok farklı yaklaşım bulunmaktadır. ATÇP için farklı çözüm yaklaşımları mevcuttur ve her yaklaşımın avantajları ve sınırlılıkları bulunmaktadır Bu çalışmalar matematiksel programlama yaklaşımları, sezgisel algoritmalar, meta sezgisel algoritmalar, hibrit algoritmalar ve meta sezgisel algoritmaların zaman karmaşıklığının azaltılmasına yönelik çalışmalar olmak üzere 5 başlıkta incelenmiştir.

Matematiksel programlama yaklaşımları

ATÇP'yi çözmek için kullanılan ilk yöntemlerden biri, matematiksel programlama yöntemleridir. Bu yaklaşımlar arasında kesirli programlama, dinamik programlama ve dal ve sınır gibi yöntemler bulunmaktadır. Bu yöntemler, optimal çözümleri bulma konusunda güçlü olsalar da, büyük ölçekli problemler için zaman alıcı ve hesaplama açısından zorlu olabilirler [4-5]. İlk defa atölye tipi çizelgeleme problemini akademik anlamda NP-Complete birproblem olarak bu çalışmada tanımlanmıştır [4]. Thompson and Jensin ilk defa atölyetipi çizelgeleme problemini akademik anlamda tanımlamış ve bu algoritmanın çözümü için bir veri seti oluşturmuştur. Bu veri seti daha sonra literatürde yapılan çalışmalarda kullanılan yaklaşımların test edilmesi açısından önemli bir yapı taşı olmuştur [5].

Sezgisel algoritmalar

Sezgisel algoritmalar problem çözme sürecinde sezgi ve deneyime dayalı kural ve yöntemler kullanarak, karmaşık problemlere yaklaşık çözümler bulmayı amaçlar. Bu algoritmalar, özellikle büyük ölçekli ATÇP'ler için daha hızlı ve pratik çözümler sunabilir. Örnek olarak, yerel arama, değişken mahalle arama ve şifreleme algoritmaları gibi yöntemler bulunmaktadır. Sabuncuoğlu ve Bayiz ışın arama algoritması ile literatürde kullanılan veri setlerinde bilenen en iyi sonuçlara göre ortalama yüzde 4.26'lık bir hata payı ile önemli sonuçlar elde etmiştir [6]. Yang ve arkadaşları klon seçimi tabanlı memetik algoritma kullanarak aynı veri setlerinde yüzde0,4 gibi çok az bir hata payı ile atölye tipi çizelgeleme problemi çözümünde önemli sonuçlar elde etmiştir [7].

Meta-sezgisel algoritmalar

Meta-sezgisel algoritmalar, daha genel ve esnek çözüm yaklaşımları sunarak, farklı problemlere adapte edilebilir ve etkili çözümler üretebilir. Bu algoritmaların temel özellikleri arasında, çözüm arama sürecinde rastgelelik ve adaptasyon kullanarak, yerel optimumlardan kaçınma ve genel optimumayaklaşma eğilimi bulunmaktadır. Meta-sezgisel algoritma örnekleri arasında genetik algoritmalar, tabu arama, benzetimli tavlama, parçacık sürü optimizasyonu ve yapay arı kolonisi gibi yöntemler bulunmaktadır [8]. Pezella ve arkadaşları esnek atölye tipi çizelgeleme problemini çözümü için genetik algoritma tasarlamışlardır [9]. Bu algoritmatasarımı içerisinde popülasyonun oluşturulmasında KİÇ ve KİS yaklaşımları kullanılmıştır. Ayrıca çaprazlama işlemi için ÖKDÇ [9] ve mutasyon işlemi ÖKM [10] tekniği kullanılmıştır. Bu makalede de bu tekniklerin farklı oranlarda kullanımının geliştirilen genetik algoritmanın çalışma zamanına etkisi incelenmiştir.

Hibrit algoritmalar

Hibrit algoritmalar, yukarıda belirtilen yöntemlerin farklı kombinasyonlarını kullanarak, daha güçlü ve etkili çözüm stratejileri geliştirmeyi amaçlar. Örnek olarak, genetik algoritmaların tabu arama veya benzetimli tavlama ile birleştirilmesi gibi yöntemler bulunmaktadır. Hibrit yaklaşımlar, farklı yöntemlerin avantajlarını bir araya getirerek, daha başarılı ve hızlı çözümler sunabilir. Ren Qing-dao-er-ji ve Wang hibrit bir genetik algoritma yaklaşımı geliştirerek Yang'ın yapmış olduğuçalışmaya göre yüzde 0,02 gibi bir iyileştirmeli sonuç elde etmiştir [11].

Meta sezgisel algoritmaların zaman karmaşıklığının azaltılmasına yönelik çalışmalar

Bu tezde odaklanılan temel konu olan meta sezgisel algoritmaların hesaplama maliyetinin azaltılması problemine literatürdeki birçok çalışmada farklı yaklaşımlar ortaya konulmuştur. Meta sezgisel optimizasyon algoritmalarının ve çok amaçlı optimizasyon algoritmalarının paralelleştirilmesi ile ilgili farklı yaklaşımlar denenmiştir. Literatürde optimizasyon problemlerinin çözümünde çokça kullanılan genetik algoritmanın paralelleştirilmesi için map-reduce yaklaşımı ile çeşitli çalışmalar yapılmıştır [12-15]. Jin ve arkadaşları[13] nın çalıştığı bu çalışmada amaç 'Genetik

Algoritmaların Paralelleştirilmesi için Map-Reduce ‘(MRPGA:) yaklaşımı ile ada modeli baz alınarak genetik algoritmaların paralelleştirilmesidir. MPRGA yaklaşımında üç temel bölüm bulunmaktadır. İlk bölüm map işlemine ayrılmıştır. Diğer iki bölüm ise reduce işlemi için ayrılmıştır. Map işlemini yapan bölümpopülasyonun bir kısmını alır ve popülasyonun her bireyi için uygunluk fonksiyonunu hesaplar. Daha sonraki aşamada reducer lokal seçim işlemini gerçekleştirir. Üçüncü aşamada ikinci aşamanın çıktıları ile global seçim işlemi gerçekleştirilir. Verma ve arkadaşları [15] bir önceki çalışmayı [13] bir aşama azaltarak 2 aşamada map-reduce işlemini gerçekleştirmeyi amaçlamıştır. İlk aşamada uygunluk fonksiyonu her birey için hesaplanırken diğer aşamada genetik algoritmanın kalan tüm aşamaları gerçekleştirilmektedir. Salza ve arkadaşları [14] ‘elephant56’ isimli bir yapı geliştirmişleridir. Bu yaklaşım geliştirici ve araştırmacılara genetik algoritmayı HadoopMapReduce kümesi içerisinde paralel olarak geliştirmeyi ve çalıştırmayı sağlamaktadır.‘Core’ ve ‘user’ katmanı ‘elephant56’ yapısının iki temel soyut katmanını oluşturmaktadır. ‘Core’ katmanı Hadoop ile haberleşmeden sorumluyken ‘user2katmanı geliştirici ve platform arasındaki arayüzü oluşturmaktadır. ‘Elephant56’ genetik algoritmanın map- reduce yaklaşımıyla paralelleştirilmesinde üç temel modeli desteklemektedir. Birincisi, ada modelindeki her bir adanın yavru kabullerini kapsayan genetik operasyonlarla çalışmaktadır. İkincisi, spesifik bir fonksiyon bir adanın içindeki her bir bireyin parametrelerinin HDFS’e reduce katmanında taşımaktadır. Bu döngü koşul sağlanana kadar tekrar etmektedir.

Literatürde map-reduce yaklaşımına ek olarak havuz tabanlı mimari ile evrimsel algoritmaların dağıtık yapıda çalıştırılması üzerine çalışmalar bulunmaktadır [16-18]. Garcia- Valdez ve arkadaşları [16]‘evospace-js’ i event-driven mimarisi üzerinde çalıştırmıştır. Node.js, JavaScript, RESTful gibi web teknolojileri ile evrimsel algoritmaların paralelleştirilmesi üzerine çalışmıştır. Bu tez çalışmasının da temelini oluşturan mikro servismimarisi ve konteyner yapıları ile meta sezgisel algoritmaların ölçeklendirilmesi için istenilen meta sezgisel algoritmanın tak çıkar yaklaşımı ile çalıştırıldığı bir çatı kuran çalışma yapılmıştır. Bu çalışma [19] popülasyon tabanlı meta sezgisel algoritmalar için dağıtık bir altyapı sunmaktadır. Bu altyapı algoritmanın aşamalarını paralel olarak çalıştırabilmektedir ve sistem kaynaklarını çok daha optimum şekilde kullanarak yüksekhesaplama gücü sunmaktadır. Bu altyapı konteyner ve mikro servis teknolojilerini harmanlayarak ölçeklenebilir bir sistem ortaya koymuştur. Genetik algoritmanın aşamalarının ayrı mikro servisler olarak tasarlandığı bu makalede

[20] genetik algoritmanın paralelleştirilmesinde bu tezde kullanılan mikro servis ve konteyner yapılarının kullanılması açısından yol gösterici olmuştur, [21-22] çalışmalarında ise genetik algoritmanın map- reduce yaklaşımı ile paralelleştirilmesi bulutta çalışan bir yapı ile tasarlanmıştır.

3. MATERYAL VE METOD

Bu bölümde tezde ortaya konan tasarım için gerekli olan algoritmalar ve teknolojiler açıklanmıştır. GA'nın aşamaları ve çalışma mantığı detaylandırılmıştır. Mikroservis mimarisinde servisler arası iletişim için kullanılan Apache Kafka teknolojisi açıklanmıştır. Servisler arası sunucu-istemci modeline dayanan Rest API açıklanmıştır. Servislerin herbiri ayrı konteynerler içerisinde çalıştırıldığından konteyner teknolojisi açıklanmıştır. Son olarak GA'nın popülasyonlarının veri yönetiminin yüksek performansla yapılması için kullanılan Redis teknolojisi açıklanmıştır.

3.1. Genetik Algoritmalar

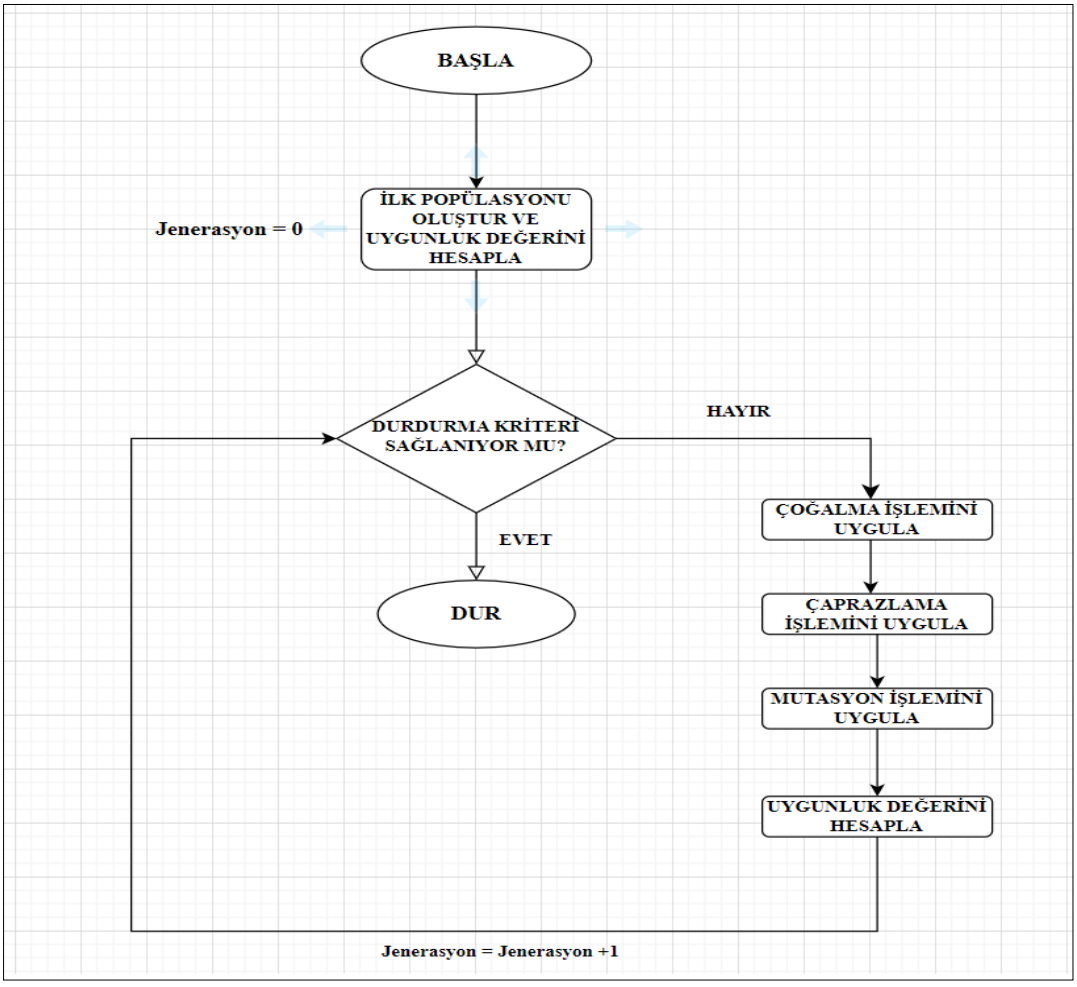
Genetik algoritmalar (GA), biyolojik evrimin dinamiklerini taklit eden ve doğal seçilim prensiplerini kullanan bir arama ve optimizasyon metodolojisidir. GA'lar genellikle geniş ve karmaşık arama alanına sahip problemlerde kullanılır ve çözüm uzayında en iyi çözümü bulmayı hedefler. GA'ların temel özelliği, bir popülasyon içinde genetik çeşitliliği korumak ve evrimsel süreci simule etmektir. Genetik algoritma, makine öğrenmesi üzerine yapılan çalışmalarda canlılardaki evrimden ve değişimden etkilenerek, bu genetik evrim sürecini bilgisayar ortamına aktarması ve böylece bir tek mekanik yapının öğrenme yeteneğini geliştirmek yerine, çok sayıdaki böyle yapıların tamamını "çiftleşme, çoğalma, değişim..." gibi genetik süreçler sonunda üstün yeni bireylerin elde edilebileceğini gösteren algoritmadır. Bir problem için olası pek çok çözümün içerisinde en uygununu bulmaya çalışan yöntemdir. Popülasyon nesilden nesile geliştikçe kötü çözümler yok olma, iyi çözümler ise daha iyi çözümler oluşturmak için kullanılma eğilimindedirler [23].

Uygulama imkânı geniş yelpazeye sahip olan genetik algoritmaların işleyiş serüveni şu şekilde açıklanabilir: Öncelikle, olası bütün mümkün çözümler bir dizi olarak kodlanır. Genellikle rastsal bir çözüm kümesi seçilir ve bu küme başlangıç popülasyonu olarak kabul edilir. Her bir dizi için bir uygunluk değeri hesaplanır. Bu uygunluk değeri istenilen çözüme ne derece yakın olunduğu ile ilgili bir ölçüttür. Bir grup dizi belirli bir olasılık değerine göre rastsal olarak seçilip yeniden çoğalma işlemi gerçekleştirilir. Yeni bireylerin uygunluk değerleri hesaplanarak, çaprazlama ve mutasyon işlemlerine tabi

tutulur. İstenilen sonuç kümeye ulaşılana dek işlemler tekrarlanır [24].

Şekil 3.1 ‘de bütün aşamalarının diyagram olarak gösterildiği GA Charles Darwin’in evrim teorisi yaklaşımından esinlenerek geliştirilmiştir. Dolayısıyla algoritma içerisinde yer alan kavramların biyolojik altyapıları mevcuttur. Bunlardan biri gendir. Gen, kalıtsal molekülde bulunan ve organizmanın karakterlerinin belirlenmesinde rol oynayan kalıtsal birimlerdir. Kromozom, birden fazla genin bir araya gelerek oluşturduğu dizidir. Popülasyon ise kromozomlardan oluşan topluluğa denir. Popülasyondaki kromozom sayısı arttıkça çözüme ulaşma süresi ve işlem adım sayısı azalır. Genetik algoritmada, var olan popülasyon üzerine bazı işlemler uygulanır.

Bu işlemlerin amacı daha iyi özelliğe sahip yeni nesiller üretmek ve arama algoritmasının alanını genişletmektir.

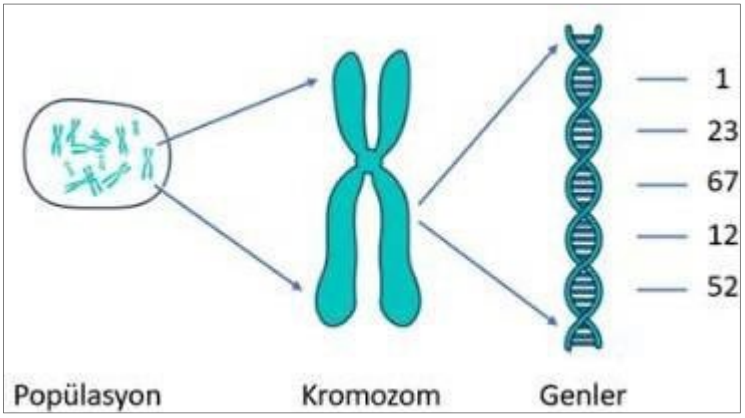


Şekil 3.1. GA Genel İşleyişi

GA aşağıdaki değinilen popülasyon, uygunluk fonksiyonu, seçilim, çaprazlama, mutasyon ve kodlama aşamalarından oluşmaktadır.

Popülasyon

GA'ların çalışma prensiplerinden biri, problemin çözüm uzayını oluşturanbir dizi aday çözüm (birey) oluşturulmasıdır. Bu bireylerin her biri genetik bir kromozom olarak ifade edilir ve problemin potansiyel bir çözümünü temsil eder. Şekil 3.2'de genlerin kromozom halinde popülasyonu oluşturma süreci ifade edilmiştir.



Şekil 3.2. Çözüm örneklerini ifade eden kromozom ve popülasyon gösterimi [25].

Uygunluk fonksiyonu

Uygunluk fonksiyonu, bir çözümün kalitesini değerlendirmek için kullanılır. Genetik algoritmanın amacı, uygunluk fonksiyonunu maksimize (veya minimize) etmektir. Bu fonksiyon, problem türüne ve hedeflere bağlı olarak belirlenir. Genetik algoritmaların uygulanmasındaki bu temel kavramları anlamak, bu yöntemin neden ve nasıl çalıştığını anlamak için hayati önem taşır. Bir genetik algoritmanın tasarımı ve uygulanması, tüm bu faktörlerin birbiriyle hassas bir denge içinde olmasını gerektirir. Bu dengenin sağlanması, genetik algoritmaların geniş ve karmaşık arama alanlarına sahip olan NP-Zor problemleri gibi zorlu problemlerin çözümünde etkinliğini belirler. Kodlama stratejisi, uygunluk fonksiyonu ve genetik operatörlerin seçimi ve uygulanması, GA'nın etkinliği ve başarısı için belirleyicidir. Örneğin, uygun bir kodlama stratejisi, algoritmanın problemi anlamasını ve çözüm uzayında etkili bir şekilde arama yapmasını sağlar. Benzer şekilde, uygun bir uygunluk fonksiyonu, algoritmanın potansiyel çözümlerin

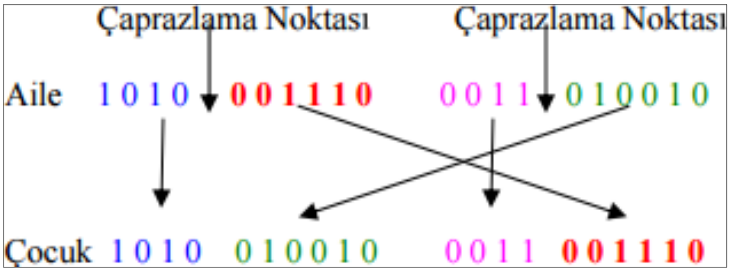
kalitesini doğru bir şekilde değerlendirmesini ve eniyi çözümleri seçmesini sağlar. Öte yandan, çaprazlama ve mutasyon operatörlerinin etkin bir şekilde uygulanması, genetik çeşitliliği korur ve yerel optimumlardan kaçmayısağlar. Bu, genetik algoritmanın hem geniş arama alanlarını kapsamasını hem de en iyi çözümlere hızlı bir şekilde yakınsamasını sağlar [23-24].

Seçilim

Seçilim süreci, bir bireyin sonraki nesilde var olma şansını belirler. Bu süreç, bireylerin uygunluk değerlerine dayalı olarak gerçekleşir. Yani, daha uygun olan bireylerin sonraki nesile aktarılma olasılıkları daha yüksektir, bu da doğal seçilim prensiplerini taklit eder.

Çaprazlama (crossover)

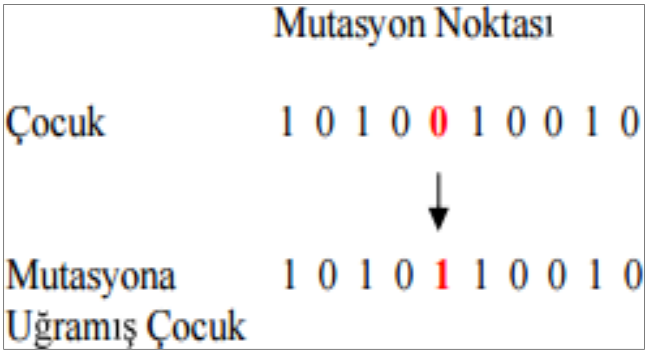
Çaprazlama, genetik algoritmaların ana işlemlerinden biridir ve iki bireyin genlerinin birleştirilmesiyle yeni bir çözüm oluşturur. Bu işlem, genetik çeşitliliği artırır ve yeni çözümler oluşturur



Şekil 3.3. Tek Noktalı Çaprazlama Örneği

Mutasyon

Popülasyonun genetik çeşitliliğini korumak ve yerel optimumlardan kaçmak içinkullanılır. Rastgele seçilen bir genin değeri değiştirilerek gerçekleştirilir.



Şekil 3.4. Mutasyon Örneği

Kodlama

GA'da her bir birey bir kromozom ile temsil edilir ve bu kromozomlar genellikle bit dizileri, sayılar listesi veya diğer veri yapıları şeklinde kodlanabilir. Problem türüne göre, uygun bir kodlama stratejisi seçmek algoritmanın etkinliği açısından önemlidir. Genetik algoritmalar, geniş ve karmaşık arama alanlarına sahip optimizasyon problemlerini çözmede etkin bir araçtır. Ancak, bu algoritmaların etkinliği, problem türüne, kodlama stratejisi, uygunluk fonksiyonu ve genetik operatörlerin uygulanma şekli gibi bir dizi faktöre bağlıdır. Bu faktörlerin her biri, genetik algoritmanın performansını ve sonuçları üzerinde belirleyici bir etkiye sahiptir. Bu nedenle, genetik algoritmaların uygulanmasında bu faktörlerin dikkatlice değerlendirilmesi ve uygun bir şekilde uygulanması gerekmektedir.

3.2. Mikroservis Mimarisi

Monolitik yapılardaki eksiklikler günümüz yazılım mühendisliği isteklerine cevap verememektedir. Monolitik yapılardaki eksiklikleri gidermek için mikro servisler giderek daha popüler hale gelmektedir. Monolitik terimi, uygulamanın diğer bölümlerini de yeniden kodlamadan bir işlevi değiştirmenin zor olduğu sıkı şekilde entegre edilmiş uygulamalar için kullanılır. Monolitik bir uygulamadaki bileşenler birçok makine arasında dağıtılabilir, ancak bunlar birbirlerine büyük ölçüde bağımlı kalırlar. Yeni bir özelliğin eklenmesi yalnızca kod boyunca dalgalanma etkilerine sahip olmakla kalmaz, değişikliğin uygulanması tüm uygulamanın yeniden test edilmesini ve yeniden ayaklandırılmasını gerektirir. Bu durum, özellikle bir uygulamada yüz binlerce hatta milyonlarca kullanıcıya sahip olduğunda, emek yoğun ve tehlikeli olabilir. Yazılım

mühendisleri altı ayda bir yazılım güncelleme lüksüne sahipken, bu tür bir yükseltme süreci tolere edilebilirdi. Günümüzdeki yazılım dünyası, yayınların haftalık, günlük veya daha sık gerçekleşmesini talep ediyor, bu nedenle monolitik uygulamalarda güncel sürüm almanın doğasında bulunan emek ve risk savunulamaz hale gelmektedir [26]. Sistemin mikro servis mimarisi ile oluşturulması için gerekli 5 şart bulunmaktadır.
Bunlar;

1. Her bir servisin tek bir amacı olmalıdır.
2. Her mikro servis ayrıktır
3. Her mikro servis taşınabilir.
4. Her mikro servis kendi verisini taşır
5. Her mikro servis sisteme etkisi olmadan oluşturulabilir ve yok edilebilir.

Meta-sezgisel algoritmaların ölçeklendirilmesinde mikro servis mimarisinin kullanılmasının avantajları:

i. Her mikro servis limitli sayıda fonksiyona sahiptir. Küçük kod tabanı barındırmaktadır ve hata çıkma olasılığı düşürülmektedir. Buna ek olarak herhangi bir hata tespit edildiği takdirde hatanın nedenin tespiti ve hatanın çözümü çok daha kolaydır.
ii. Mikro servisler sistemden bağımsız bir şekilde kolayca test edilebilirler. Genellikle bir mikro servis bir ekip tarafından yönetilir. Servislerin sorumlulukların ekiplere dağıtımı ve süreç yönetimi açısından çok daha avantajlıdır.
iii. Her bir mikro servis diğer servislerle yer değiştirebilir. Her servis kolaylıkla güncellenip konteyner yapısı içerisinde yeniden ayaklandırılabilir.
iv. Mikro servis mimarisi bütün modüllerin çiftelenmesi anlamına gelmemektedir. Yazılım geliştiricinin servisini monitör edebilmesi ve fazla işlem gücü gereken servislerini ölçeklendirmesini sağlamaktadır.

şeklinde ifade edilebilir.

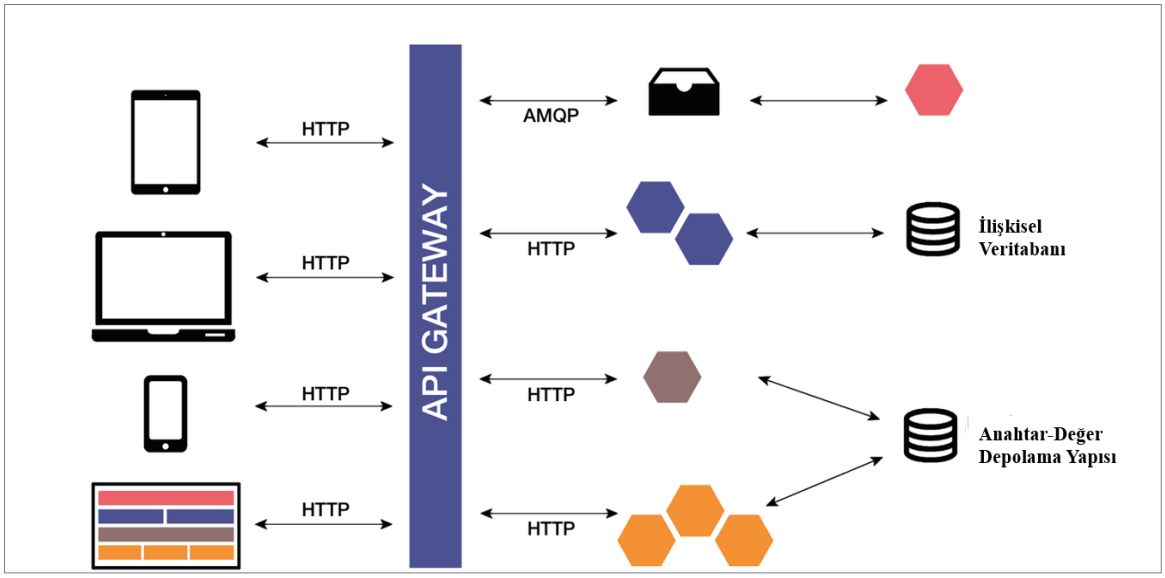
Mikroservis mimarisinin monolitik yapılara göre dezavantajlı tarafları da bulunmaktadır. Dezavantajları tarafları da:

i. Gönderilen mesajların ağdaki hızı servisin hafızasına yazma- okuma işlemlerinden daha yavaş kalabilir. Böyle bir durumda servislerin çalışması ağdaki gönderim hızı limitine takılabilir.
ii. Mikro servisler arası iletişim JSON veya XML formatındaki veriler ile sağlanmaktadır. Verilerin içeriğinin görüntülenme olasılığına karşı güvenlik önlemlerinin maksimize edilmesi gerekmektedir.

şeklinde sıralanabilir.

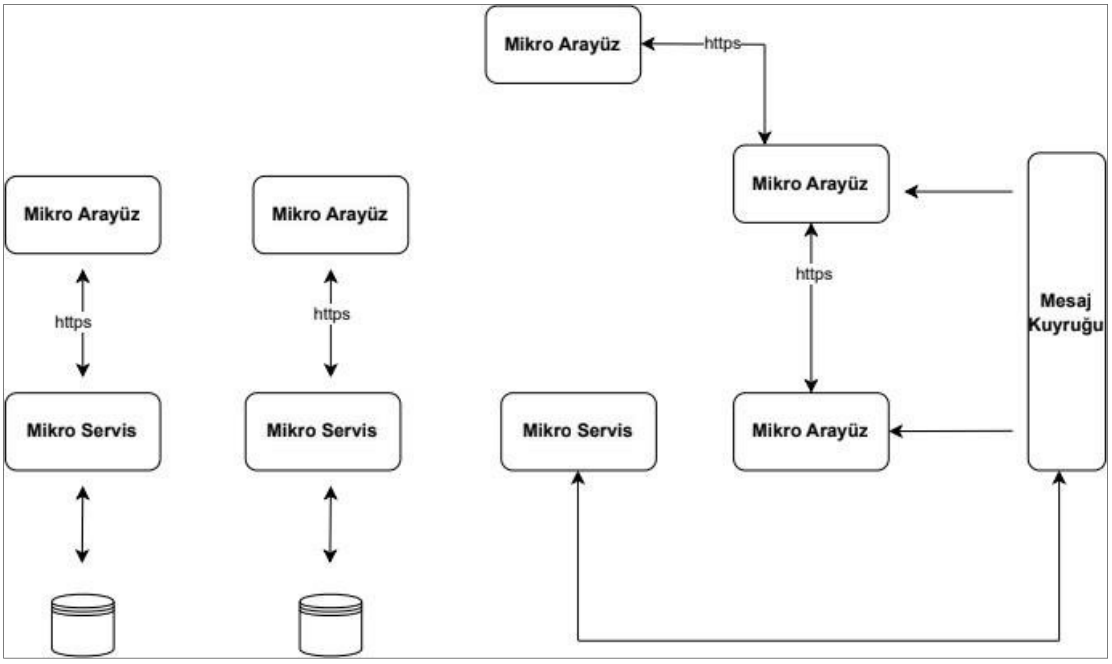
3.2.1. Mikroservis mimarisine genel bakış

Mikroservis mimarisi farklı iş yeteneklerinin birbirinden bağımsız bir şekilde yapılandırıldığı mimari bir modeldir. Uygulamayı tek bir monolitik uygulama halinde inşa etmek yerine, daha küçük, birbirine bağlı bir hizmet uygulamaları grubu halinde inşa etme fikrine dayanır. Bu mimaride büyük sistemler her biri farklı bir işten sorumlu ve birbirinden bağımsız daha küçük servislere ayrılır. Tüm bu servisler mikro servis olarak adlandırılır. Mikro servisler REST, MQ vb. mesajlaşma protokolleri bazlı programlama arabirimleri (API) sunar ve birbirleri ile iletişimde bu arabirimleri kullanırlar. Mikroservisler çevikliği destekler. Herhangi bir yeni özellik hızla geliştirilip sisteme adapte edilebilir. Her servisin kodu diğerinden bağımsız kurulur ve yönetilir. Her bir servis farklı bir dil ya da kütüphane kullanılarak üretilebilir. Yeni teknolojilerin kolayca denenmesini ve uygulanmasını sağlar. Şekil 3.5'te gösteriliği gibi her mikroservis farklı mesajlaşma protokolünü ve farklı veri tabanı teknolojilerini kullanabilmektedir. Geliştiriciler ve ekipler birbirlerinden bağımsız olarak çalışabilir, böylece hızlıca geliştirme ve test süreçleri yürütülebilir [26].



Şekil 3.5. Mikroservis genel yapısı

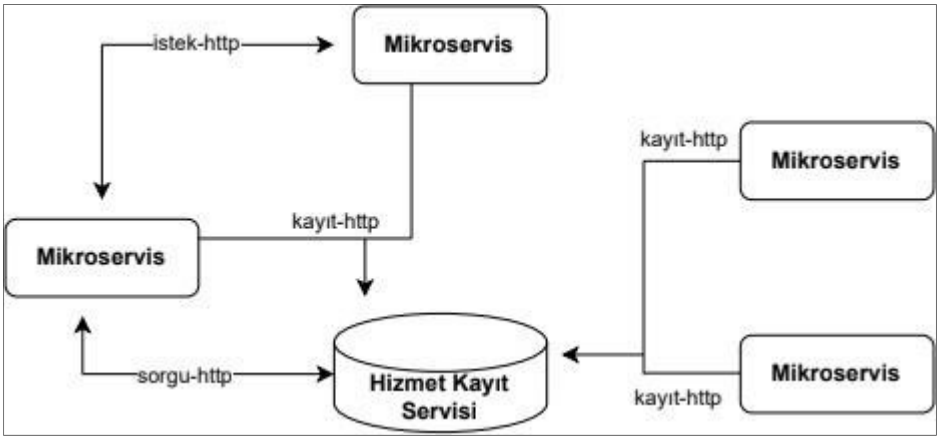
Her bir servisin verisi kendisine aittir. Ayrı tablolar, ayrı şemalar ya da ayrı veri tabanları şeklinde yapılandırılabilirler. Mikro servisler, kolayca değiştirilebilen ve versiyonları artırılabilen bağımsız bileşenlerdir. Yazılım geliştirme sürecinden testlerinin yapılması ve kurulumuna kadar istenilen ölçüde otomatikleştirmeye imkan sağlar. Servisler ayrı ayrı devreye alınacağı için, devreye alım süresi kısalır ve maliyet azalır. Servisler küçük olduğu için bakımı kolay yapılır. Yazılımda yeni özellikler getirme sıklığını artırır. Gereksinimlere çok hızlı tepki verilir. Hata izolasyonu sağlar. Eğer bir serviste bir sorun olursa, tüm sistemi etkilemeyecek, sadece o serviste kalacaktır. Geri kalan sistem sorunsuz çalışmaya devam edecektir. Şekil 3.6'da gösterildiği gibi mikroservisler mesaj kuyruğu ve HTTP üzerinden haberleşmektedirler.



Şekil 3.6. Mikroservisler arası iletişim yapısı

Mikroservis mimarisinde hizmet adreslenebilirliği

Farklı ip adreslerine sahip servislerin(ya da API'lerin) birbirleri arasındaki haberleşme için geliştiriciler tarafından adres bilgilerinin bilinmesi zorunluluğunu ortadan kaldırarak, bu servislerin ortak bir yerde kayıtlı olmasını sağlayan ve böylece erişim tarifini kolaylaştırarak iletişimi daha sade hale getiren bir modeldir. Bu model sayesinde ilgili proje ekosisteminde bulunan tüm servisler birbirleriyle servis adı ile rahatlıkla iletişim kurabilmekte ve geliştiriciler ip veya dns adresi, port vs. gibi konfigürasyon bilgilerine ihtiyaç duymamaktadır. Şekil 3.7'de gösterildiği gibi her bir servis isteklerini hizmet kayıt servisi üzerinden yönlendirmektedir. Tezde yapılan mikro servis çalışmasında genetik algoritma problemi ve kısıtları http isteği ile gitmektedir. Burada hizmet kayıt servisi ile web servis istekleri atılmıştır



Şekil 3.7. Mikroservislerde hizmet kayıt servisi

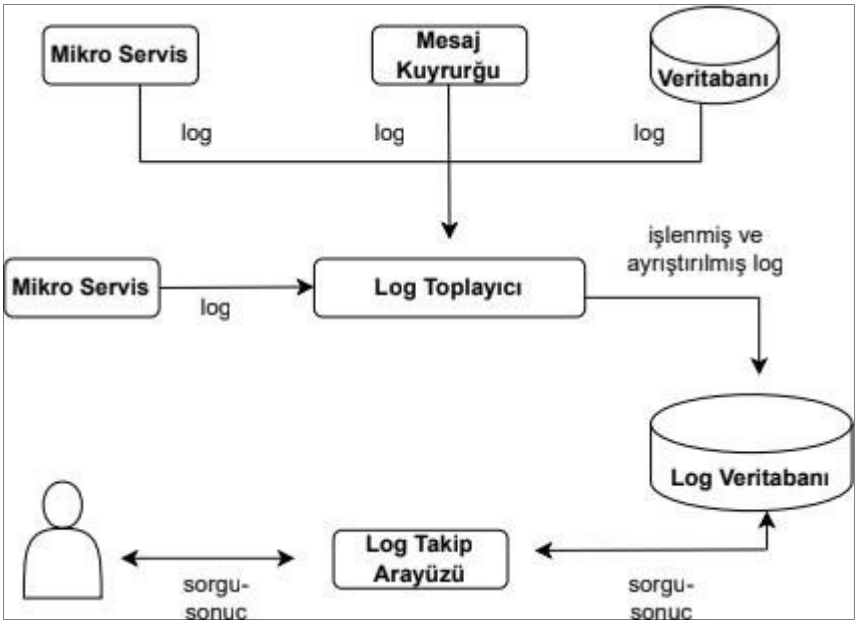
Servis sağlık durumu

Mikroservis mimarisi uygulandığında, bir servisin çalışır durumda olması ancak işlemleri gerçekleştirememesi olasılığı vardır. Her servisin, uygulamanın durumunu kontrol etmek için kullanılabilecek /health gibi bir uç noktası olması gerekir. Bu API, servisin durumunu, diğer hizmetlere, altyapıya bağlantıyı ve herhangi bir özel mantığı kontrol etmelidir. Spring boot actuator, uygulamaların ürün ortamına hazır özelliklerini (servis sağlık kontrolü, disk kullanımı vs.) otomatik aktifleştirir ve farklı HTTP erişim noktaları ile etkileşimde bulunmayı sağlayan bir yapı sunar.

Log toplama

Her bir mikro servis ayrı bir şekilde log üreteceğinden tüm sistemde çalışan yüzlerce servisin loglarını ayrı ayrı gözlemlemek ve yorumlamak çok zor olabilir. Çalışan her birservisten logları toplayan merkezi bir log kayıt hizmetine ihtiyaç duyulmaktadır. Şekil 3.8'de gösterildiği gibi log toplayıcı üzerinden veri tabanına kaydedilen loğlar log takip arayüzü ile görüntülenir.

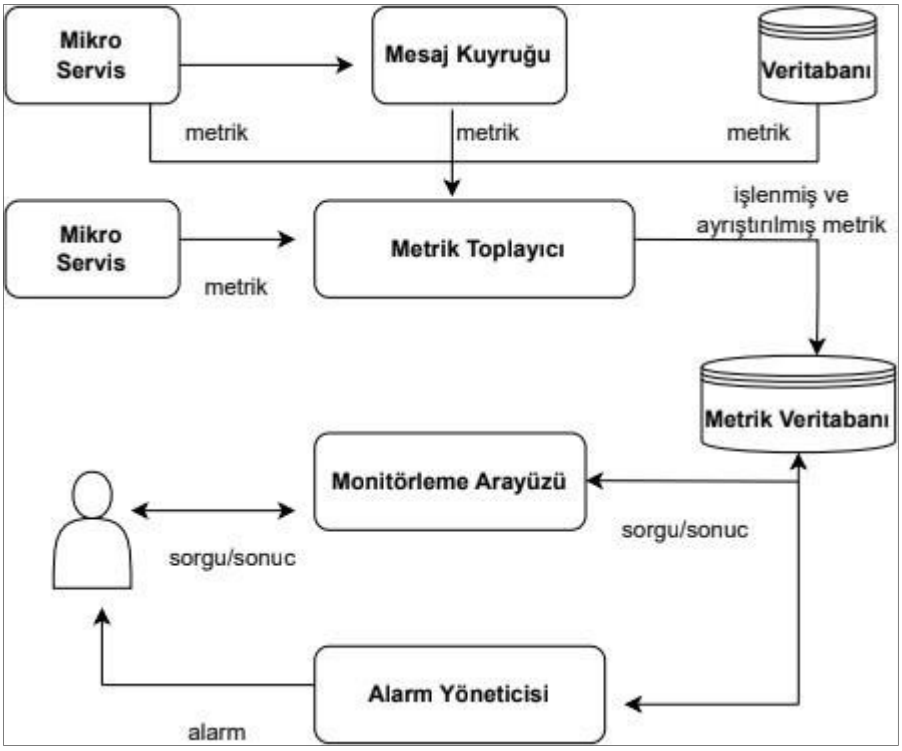
Kullanıcılar bu loglar içerisinde arama ve analiz yapabilir. Log tiplerine (error, warning, info vs) ve loglardaki belirli anahtar kelimelere göre tetiklenen uyarılar oluşturulabilir.



Şekil 3.8. Mikro servislerde log toplama yapısı

Performans metrikleri

Mikroservis mimarisi ile geliştirilen bir projede çok fazla sayıda aktif olarak çalışan örnek olması nedeniyle, bu örneklerin izlenebilmesi ve bir sorun olduğunda uyarıların gönderilebilmesi için işlemleri izlemek kritik hale gelir. Her bir servis örneğinin metriklerini toplamak için bir ölçüm hizmeti gereklidir. Raporlama ve uyarı sağlayan bir uygulama hizmetinin metriklerini toplamalıdır. Şekil 3.9'da gösterildiği gibi metrik veri tabanından alınan veriler monitörleme arayüzü ile kullanıcıya gösterilir.

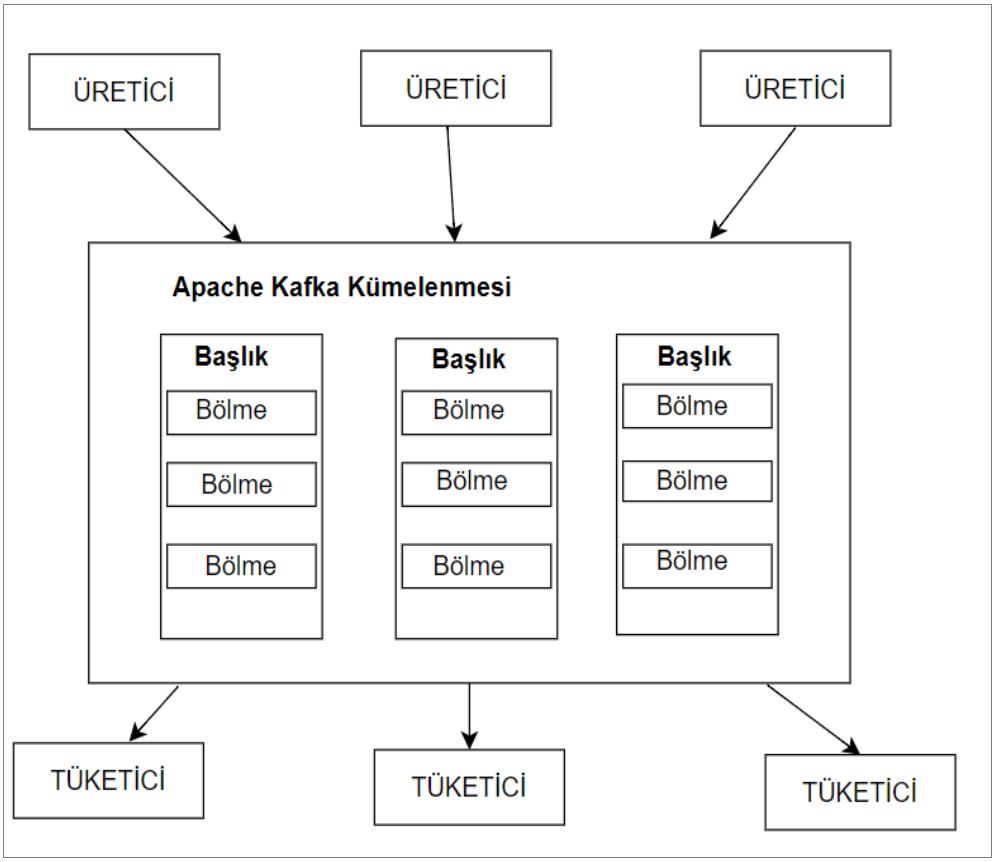


Şekil 3.9. Mikro servislerde performans metrikleri görüntüleme yapısı

3.2.2. Apache Kafka

Tasarlanan mikro servis yapısı içerisinde her bir servisin birbiriyle olan iletişimi çeşitli teknolojiler ile sağlanmaktadır. Veri iletimi servislerin ölçeklenebilir olması ilkesi doğrultusunda gerçekleştirilmelidir. Oluşturulan tasarımda servisler arası iletişim rest ve Apache Kafka teknolojileri ile gerçekleştirilmektedir

Apache Kafka, yüksek performanslı bir TCP ağ protokolü aracılığıyla iletişim kuran sunucular ve istemcilerden oluşan dağıtık bir haberleşme sistemidir. Şekil 3.10'da gösterildiği gibi veriyi gönderen servis 'başlık (topic) adı verilen kanallara veri kuyruğuna ekleyerek veriyi gönderir. Apache Kafka veriyi kuyruk yapısında depolar. Verilerin tutulduğu kanalı dinleyen servisler'topic' deki verileri tüketerek gerçek zamanlı veri iletişimini gerçekleştirirler. Apache Kafka, yüksek hacimli, hızlı ve güvenilir veri akışlarını işlemek için kullanılan bir yayın- abone mesajlaşma sistemi ve gerçek zamanlı veri akış platformudur. Kafka, mikroservisler arasındaki asenkron iletişim için genellikle tercih edilir. Kafka, büyük veri akışlarını hızlı ve hatasız bir şekilde işleyebilir ve bu özelliği, özellikle hızlı yanıt süreleri gerektiren ve/veya büyük ölçekte veri işleyen mikroservis tabanlı uygulamalar için faydalıdır [27].



Şekil 3.10. Apache- Kafka Yapısı [27].

Apache Kafka üzerinde veriler anahtar-değer(key-value) ikilisi ile gönderilir. Her bir başlık (topic) bölme (partition) lardan oluşur. Aynı anahtara sahip bilgiler aynı bölme (partition) ye yazılır. Apache Kafka yedekli olarak çalışabilmektedir. Veri iletimindeyken bir servisin çökmesi durumunda Apache Kafka yeninden dengeleme (rebalance) yaparak veri kaybı yaşanmadan veri iletiminin devam etmesini sağlamaktadır.

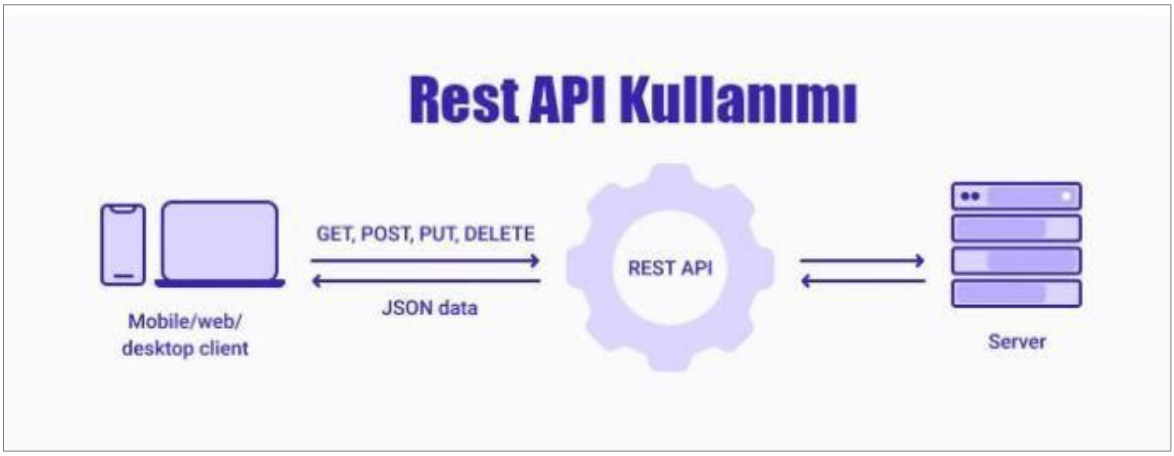
Tez çalışmasında, Apache Kafka tasarlanan mikroservis yapısındaki servisler arası veri iletimi için kullanılmıştır. Tasarlanan her bir servis çıktısını gerçek zamanlı olarak bir değer servise Apache Kafka ile iletmiştir.

3.2.3. Rest API

REST API'ler, mikro servislerin birbiriyle ve dış dünya ile iletişim kurmasını sağlar. Bu API'ler, HTTP protokolünün standart yöntemlerini (GET, POST, PUT, DELETE) kullanarak, servisler arasında bilgi aktarımını sağlar. REST API'ler, basitlik, ölçeklenebilirlik ve durum bilgisizlik (statelessness) gibi özellikleri sayesinde mikro

servis mimarisine uygundur.

Rest API, mikroservisler arası iletişimi sağlamak amacıyla HTTP protokolü üzerinde çalışan istemci-sunucu mimarisi ile kullanılan arabirimdir. Şekil 3.11'de gösterildiği gibi aynı HTTP protokolünde olduğu gibi kendisine istek atılmasını isteyen servis sunucu olarak servislerini tanımlar. Bu servisle iletişim kurmak istenildiği zaman tanımlanan servise istek atılır. GET, POST, PUT ve DELETE komutları üzerinden iletişim kurulur. Rest API ile senkron iletişimkurulmaktadır. Asenkron iletişim kurmak için kuyruk tabanlı çalışan Kafka teknolojisinin kullanılması daha uygun olacaktır [28].



Şekil 3.11. Sunucu istemci modelinde rest-api kullanımı [29].

Tezde kısıtların ve problem tanımındaki ögelerin iletimi için Rest API kullanılmıştır. Popülasyon sayısı, problemin çalışacağı zaman periyodu, her bir operasyonun çalışması gereken makine, çalışma sırası ve çalışma zamanı gibi kısıtlar bu şekilde başlatma servisine iletilmiştir.

3.2.4. Konteyner teknolojisi

Konteyner sanal makinelere benzer bir rol oynayan bir uygulama dağıtım teknolojisidir. Tıpkı geleneksel sanallaştırma gibi, konteynerler de uygulamalarımız için yalıtılmış ortamlar sağlar. Sanal makinalar ana bilgisayar üzerinde kendi işletim sistemlerini çalıştırabilmek için hiper yönetici kullanırken, konteynerler ana bilgisayar işletim sisteminin çekirdeğini diğer konteynerler ile paylaşır. İşletim sisteminin tüm özelliklerini yüklemedikleri ve bunun yerine yalnızca uygulamanızın gerçekten ihtiyaç duyduğu

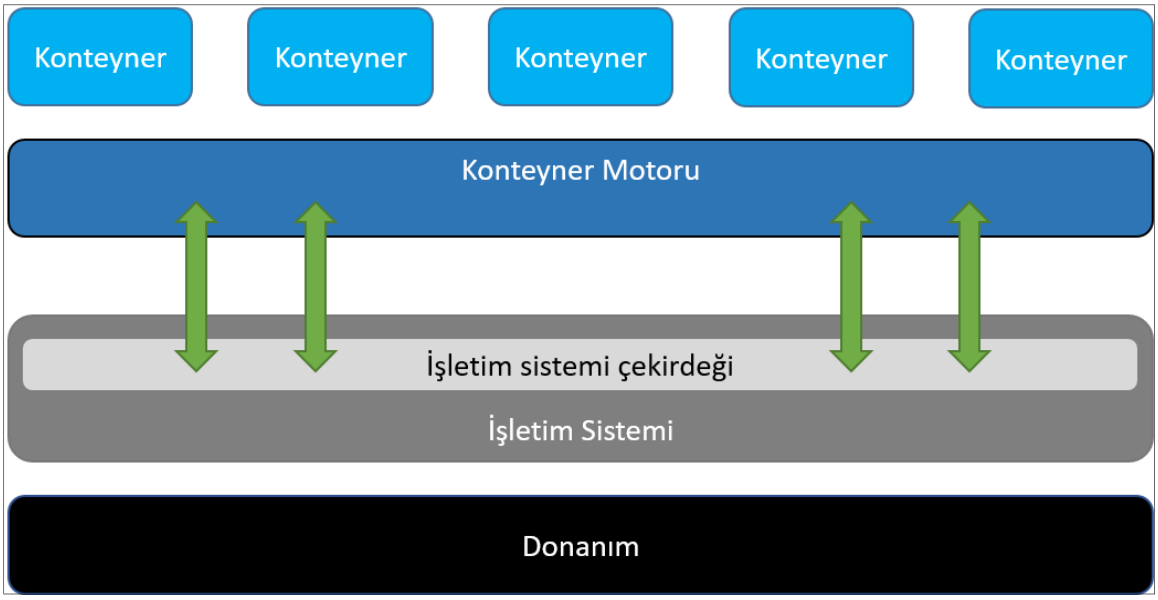
yüklemeler, bağımlılıklar ve kod gibi kaynakları sağladıkları için iş yüklerimiz için çok daha düzenli bir ortam sunarlar.

Konteynerler ayrıca uygulamalarımızın altyapı ayak izini önemli ölçüde azaltabilir, sanal makinalardan daha iyi performans sunar ve daha hızlı durup başlayabilirler. Bu da onları dalgalı ölçeklendirme gereksinimlerine daha duyarlı halegetirir. Konteynerler, Docker ve Kubernetes gibi konteynerleştirme teknolojisi ile birlikte bir çok geliştiricinin araç setlerinde giderek yaygınlaşan bileşenlerdir. Konteynerleştirmenin amacı, özünde öngörülebilir ve yönetimi kolay bir şekilde farklı ortamlarda yazılım oluşturmak, paketlemek ve dağıtmak için daha iyi bir yol sunmaktır. Konteynerler, konteyner çalışma özellikli herhangi bir ana bilgisayarda tutarlı bir şekilde çalışır, böylece geliştiriciler daha sonra tam üretim ortamlarına dağıtacakları aynı yazılımı yerel olarak test edebilir.

Konteynerler, işletim sistemi çekirdeğini paylaşarak, uygulama başına tam işletim sistemi özelliklerine olan gereksinimi ortadan kaldırır ve konteyner dosyalarının küçük olmasını ve kaynaklara yüklenmemesini sağlar. Özellikle sanal makinelere kıyasla daha küçük boyutları, hızlı bir şekilde oluşturma işlemi gerçekleştirebilecekleri ve yatay olarak ölçeklenen bulut tabanlı uygulamaları daha iyi destekleyebilecekleri anlamına gelir.

Şekil 3.12'de gösterildiği gibi konteynerler taşınabilir ve platformdan bağımsızdır. tüm bağımlılıklarını yanlarında taşır; bu, yazılımın bir kez yazılabileceği ve ardından dizüstü bilgisayarlar, bulut ve şirketiçi bilgi işlem ortamlarında yeniden yapılandırılmaya gerek kalmadan çalıştırılabileceğianlamına gelir. Modern geliştirmeyi ve mimariyi destekler. Platformlar arasında devreye alım taşınabilirliği/tutarlılığı ve küçük boyutlarının bir bileşimi nedeniyle, konteynerler, modern geliştirme ve sunucusuz ve mikro hizmetler gibi küçük artışlarla düzenli kod devreye alımlarına dayanan uygulama kalıpları için idealdir. Kullanımı iyileştirir ve kendilerinden önceki sanal makineler gibi, konteynerler de geliştiricilerin ve operatörlerin, fiziksel makinelerin CPU ve bellek kullanımını iyileştirmelerini sağlar. Konteynerlerin daha da ileri gittiği nokta, mikroservis mimarilerini de etkinleştirdikleri için uygulama bileşenlerinin daha alt seviyede ayrıntılandırılarak devreye alınabilmesi ve ölçeklenebilmesidir; bu, tek bir bileşenin yükle mücadele etmesi nedeniyle, monolitik uygulamanın tamamının ölçeğiniyükseltme

zorunluluğuna cazip bir alternatiftir.



Şekil 3.12. Konteyner yapısının işletim sistemi ile ilişkisi

Docker, uygulamaları ve onların bağımlılıklarını bir araya getirerek "konteyner" adı verilen izole edilmiş birimler halinde paketlemek için kullanılan bir açık kaynaklı platformdur. Docker, mikroservislerin dağıtımında ve yönetiminde önemli bir rol oynar. Konteyner teknolojisi, her bir mikroservisin ayrı bir ortamda çalıştırılmasını ve bu ortamların birbirinden izole edilerek çakışma ve uyumsuzluk problemlerinin önlenmesini sağlar. Docker, mikroservis mimarisi için hızlı, güvenli ve taşınabilir bir çözüm sunar [30].

Tez çalışmasında her bir servis konteyner halinde sanallaştırılarak çalıştırılmıştır. Genetik algoritma içerisinde mikro servis haline getirilen her bir öge Docker konteynerinde çalıştırılmıştır.

3.2.5. MongoDB

MongoDB, NoSQL tabanlı, belge odaklı bir veri tabanı sistemidir. Geleneksel tablo tabanlıveritabanlarından farklı olarak, MongoDB belge tabanlı bir yaklaşım benimser ve verileri JSON benzeri belgelerde saklar. Bu, verilerin daha doğal ve esnek bir şekilde saklanmasına ve sorgulanmasına olanak sağlar. Mikroservis mimarisi kapsamında,

MongoDB genellikle mikroservis için ayrı bir veri tabanı sağlar. Bu yaklaşım, veritabanı şeması değişikliklerinin diğer mikroservisleri etkilemesini önler ve mikroservislerin bağımsız olarak geliştirilmesini ve ölçeklendirilmesini kolaylaştırır [31].

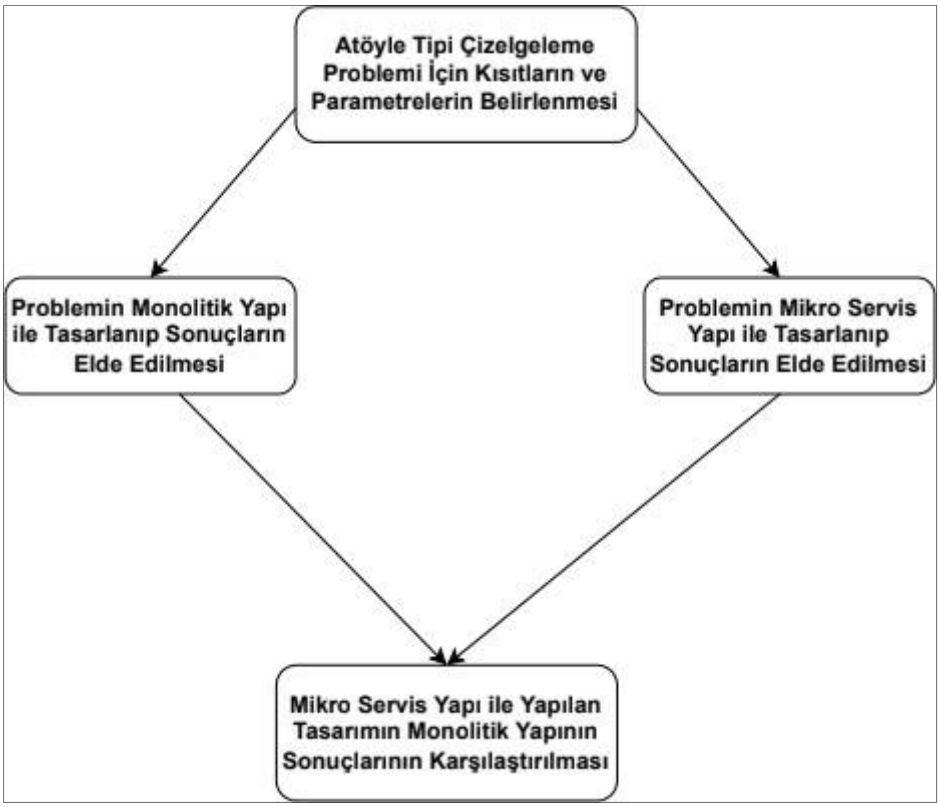
Tez çalışmasında, her bir kuşak diğer kuşak oluşturulmadan önce veri tabanına kaydedilmiştir. Ayrıca tanımlı olan problem ve kısıtlar başlangıçta veri tabanına kaydedilmiştir. Kullanıcı tarafından değiştirilmesi halinde de veri tabanında güncellenmiştir.

3.2.6. Redis

Redis, bellek üzerinden işlem yapan açık kaynaklı bir veri tabanıdır ve genellikle önbellekleme ve mesajlaşma sistemleri için kullanılır. Redis, süper hızlı yanıt süreleri sağlar ve bu özelliği, özellikle performansı önemli ölçüde etkileyen yüksek I/O işlemleri için kullanışlıdır. Mikroservis mimarisinde, Redis genellikle önbellek olarak kullanılır, bu da her mikroservisin veri tabanına yapılan sorgulamaların sayısını azaltabilir ve böylece genel uygulama performansını iyileştirebilir. Redis ayrıca, üretici-tüketici mesajlaşma özelliği sayesinde mikroservisler arasındaki asenkron iletişimi de destekler. Bu tez çalışmasında birden fazla konteyner aynı anda çalışmaktadır. Konteynerler arası veri bütünlüğü Redis önbellek yapısı ile sağlanmaktadır. Popülasyondaki bireyler ve lineer sıralamayöntemiyle oluşturulan çark yapısı gibi anlık gerçek zamanlı veri güncellemelerinin olduğuyapılar Redis önbellek yapısı içerisinde tutulmuştur[32].

4. MONOLİT VE MİKROSERVİS MİMARİ TASARIMLARI

Şekil 4.1'de de gösterildiği gibi araştırmada ATÇP'nin monolit ve mikroservis temelli iki farklı mimaride GA ile çözüm modeli tasarlanmıştır. Monolitik mimaride zaman karmaşıklığının azaltılması adına farklı teknikler denenerek zaman karmaşıklığına etkisi incelenmiştir. Daha sonra monolit yapıda alınan sonuçları daha da iyileştirmek adına GA'nın her bir aşaması mikro-servis mimarisinde ayrı birer servis olarak tasarlanmıştır. Her bir servis ölçeklendirilerek(aynı anda birden fazla sanal sunucu üzerinde çalıştırılarak) algoritmanın paralelleştirilmesi sağlanmıştır.

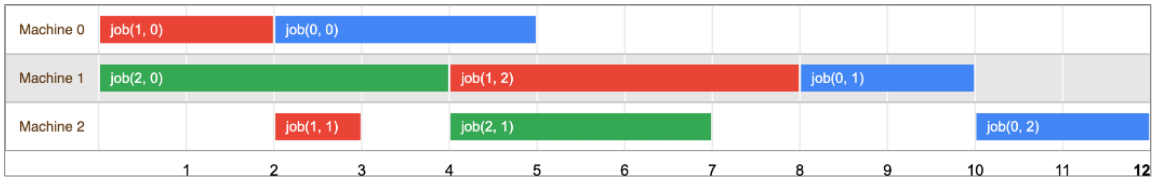


Şekil 4.1. Çalışmanın Genel İşleyişi

4.1. Genetik Algoritmanın Monolit Tasarımı

Genetik algoritma evrim fikrinden yola çıkarak oluşturulmuş optimum sonuca daha yakın olan çözümlerin sonraki jenerasyonlara aktarılması mantığına göre işleyen bir algoritmadır. Başlangıç popülasyonunun oluşmasıyla her seferinde iteratif bir şekilde yeni nesiller oluşturarak optimum sonucu bulmayı hedefler. Genetik algoritmada üretilen her yeni nesil bir önceki nesle göre optimum sonuca daha yakın kromozomlar

içermektedir. Bu çalışmada ATÇP için bir genetik algoritma yaklaşımı tasarlanmıştır. Tasarlanan bu algoritma birçok farklı ATÇP örneği için kullanılabilmektedir. Şekil 4.2'de tezde kullanılan ATÇP veri modelinin bütün kısıtları karşılayan çözümü. gösterilmiştir.



Şekil 4.2. Tezde ele alınan çizelgeleme problemi [33].

Örnekte, 0. işin üç görevi vardır. İlki (0, 3), 0, 3 birim olarak makine 0'da işlenmelidir. İkincisi (1, 2), makinenin 1. cihazında 2 zaman birimi şeklinde işlenmelidir. Toplamda sekiz görev vardır. Şekil 4.2'de verilen çözüm tüm kısıtları sağlayan optimum bir çözüm örneğidir. Tasarlanan algoritma kodlama, popülasyon oluşturma, uygunluk fonksiyonu, seçim, çaprazlama ve sonlandırma olmak üzere 6 aşamadan oluşmaktadır.

i. Kodlama**:** Algoritmanın içerisinde ifade edilen her bir kromozom bir çözümü ifade eder.
ii. Popülasyonu oluşturma: Algoritmanın ilk popülasyonunun oluşturulma aşaması çok önemlidir. Çünkü optimum sonuca gidecek olan üretilen nesiller bu üretilen popülasyonun üzerinden ilerleyecektir. Burada ilk popülasyon oluşturulurken Rastgele Seçim, KİÇ, KİS yöntemleri uygulanmıştır.
iii. Uygunluk Fonksiyonu: Oluşturulan popülasyondaki her bir kromozomun optimum sonuçtan ne kadar uzak olduğunu ölçmektedir. Her bir kromozomun ceza puanı oluşmaktadır.
iv. Seçim: Ceza puanları verilen kromozomlardan çaprazlama işlemi için seçim yapılması gerekmektedir. Bu işlem için, ikili turnuva, n- size turnuva [24] lineer sıralama [24] ve rulet tekerleği teknikleri kullanılmıştır.
v. Çaprazlama ve Mutasyon: Çaprazlama ve mutasyon işlemleri bir sonraki neslin oluşması için gerekli olan bir adımdır. Seçilen iki gen yeni nesildeki kromozom sayısı tamamlanıncaya kadar kendi arasında çaprazlanır ve mutasyon işlemi uygulanır. Bu işlem için atama ve sıralama olmak üzere iki aşama tasarlanmıştır. Atama işlemi için akıllı mutasyon, atamaya dayalı çaprazlama ve atamaya dayalı

mutasyon işlemleri uygulanır. Sıralama için ÖKSDÇ ve ÖKM teknikleri uygulanmıştır.

vi. Sonlandırma: Bu işlemler yapıldıktan sonra oluşturulan yeni nesil içerisindeki her bir kromozom uygunluk fonksiyonuna sokulur. Uygunluk fonksiyonunun sonucu 0 çıkan bir kromozom bulunduğu takdirde bütün kısıtları sağlayan çözüm bulunmuş demektir ve algoritma sonlandırılır. Bulunmazsa aynı işlemler tekrar edilerek yeni nesille oluşturulmaya devam edilir.

4.1.1. İlk popülasyonun oluşturulması

Algoritmanın genelinde kullanılacak olan popülasyondaki kromozom sayısı belirlendikten sonra bu sayıda kromozom belirtilen kurallara göre oluşturulmuştur. Kromozom sayısının çalışma zamanına etkisi de bu çalışmada incelenmiştir. N sayıda kromozom oluşturulurken belirli oranlarda aşağıda anlatılan tekniklerden yararlanırmıştır:

i. Rastgele iş seçimi: İşlerin makinelere ataması ve çalışacağı zaman aralığı rastgele seçilmiştir.
ii. Kalan İşlerin Çoğu (KİÇ): Burada kromozomlar oluşturulurken önce çalışma zamanı en fazla olan operasyonlara öncelik verilir. Yani operasyonlar çalışma zamanı en fazla olan operasyondan en az olana göre sıralandıktan sonra, önce çalışma zamanı en fazla olan operasyona makine ve çalışacağı zaman aralığının ataması yapılır [34].
iii. Kalan İşlerin Sayısı (KİS): Burada kromozomlar oluşturulurken en çok operasyona sahip olan işlere öncelik verilerek makine ve çalışacağı zaman aralığının ataması yapılır [34].

4.1.2. Kodlama

Algoritma içerisinde kromozom ve popülasyon kavramlarının ifade edilmesi için kullanılan bir veri modeli bulunmaktadır. Bu çalışmada nesne yönelimli programlama altyapısı ile Java programlama dili kullanılmıştır. Bu kapsamda her bir kromozom (bir çözüm örneği) Şekil 4.3'te java sınıfı ile temsil edilmiştir. Çalışan operasyonun çalışacağı makine, hangi sırada çalışacağı, hangi zaman dilimi içerisinde çalışacağı bilgileri tek bir sınıf

içerisinde tutulmuştur. Kromozomlar listesi ise popülasyonu oluşturmaktadır. Üretilen bir sonraki nesillerde daha iyi sonuca yakınsayan kromozomlar listesini ifade etmektedir.

```
class Task {
    1 usage
    final int machine;
    1 usage
    final int startTime;
    1 usage
    final int stopTime;
    1 usage
    final int duration;
    1 usage
    final int jobNum;
    1 usage
    final int precedenceJob;
    1 usage
    final int id;

    no usages
    public Task(int id, int startTime, int stopTime, int machine, int duration, int jobNum, int precedenceJob) {
        this.machine = machine;
        this.startTime = startTime;
        this.stopTime = stopTime;
        this.duration = duration;
        this.jobNum = jobNum;
        this.precedenceJob = precedenceJob;
        this.id = id;
    }

}
```

Şekil 4.3. Algoritma içerisindeki kromozomun Java sınıfı gösterimi

Popülasyon içerisindeki örnek bir kromozom aşağıda gösterilmiştir.

$$S = (O_{10},O_{M0}), (O_{00},O_{M0}), (O_{20},O_{M1}), (O_{12},O_{M1}), (O_{01},O_{M1}), (O_{11},O_{M2}), (O_{21},O_{M2}), (O_{02},O_{M2}) \quad (4)$$

4.1.3. Uygunluk fonksiyonu

Uygunluk fonksiyonu iki temel kısıt üzerinden değerlendirilmiştir. İşlerin doğru makineye atanıp atanmadığı kontrolü ve bir işi oluşturan operasyonların doğru sırada çalışıp çalışmadığı kontrolüdür. Bir kromozom içerisinde kısıtlara uymayan her bir operasyon için uygunluk fonksiyonu 0'dan başlatılarak 1 arttırılır. En uygun çözüm bulunduğunda uygunluk fonksiyonunun 0 olması beklenir.

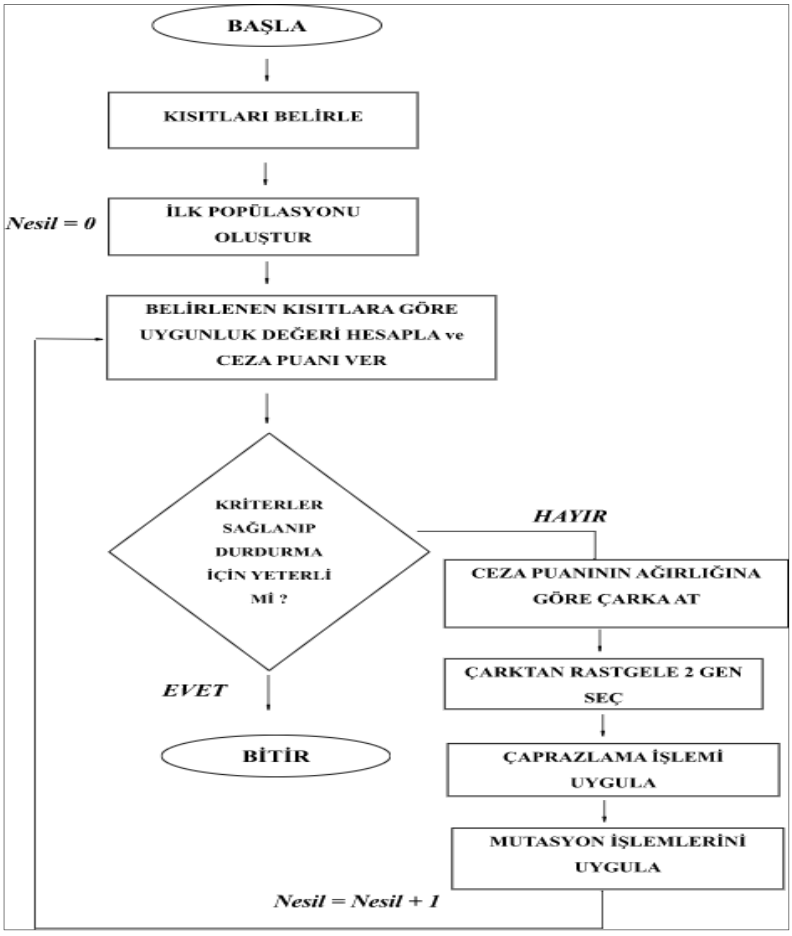
4.1.4. Seçilim

Uygunluk fonksiyonları hesaplanmış popülasyondan iki genin seçilmesi işlemi için farklı teknikler kullanılmıştır. Bu tekniklerin farklı oranlarda kullanılmasının sonuca etkisi de araştırılmıştır. Seçilim için kullanılan teknikler şunlardır;

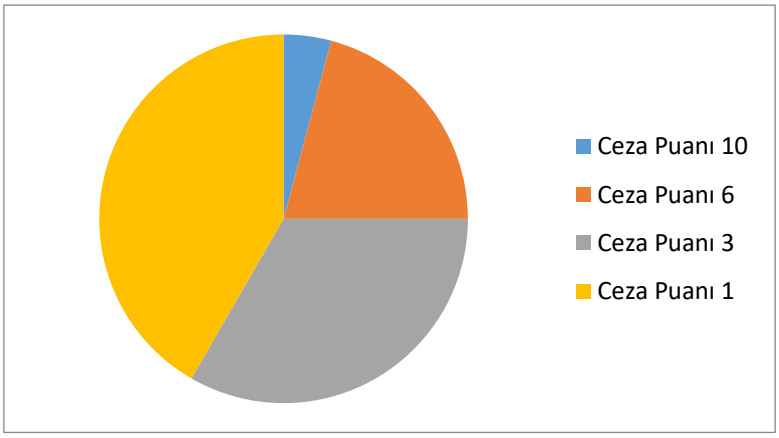
i. İkili Turnuva Seçilimi: Çaprazlanma için gereken iki gen rastgele seçilir.
ii. Turnuva Seçilimi: Çaprazlama için seçilen bireyler rastgele sayıda birey arasından seçilir.
iii. Lineer Sıralama Seçilimi: Popülasyon içerisindeki kromozomlar ceza puanı en azdan çoğadoğru sıralanır. Aşağıda verilen olasılığa göre ceza puanı düşük olanın seçilme yüzdesi daha fazla olacak şekilde iki genin seçilmesi sağlanır. Eş. 5'te i. Elemanın seçilme olasılığı gösterilmiştir.

$$pi = \frac{2ri}{N(N+1)}\ ,\ i = 1, \ldots\ldots\ldots, N \tag{5}$$

iv. Rulet Tekerleği Seçilimi: Şekil 4.4'te rulet tekerleği yönteminin algoritma içerisindeki kullanımı gösterilmiştir. Popülasyon içerisindeki kromozomlar ceza puanı ile ters orantılı olacak şekilde Şekil.4.5'te gösterildiği gibi bir çarka atılır ve kromozomların bu şekilde dağıtıldığı çarktan rastgele iki gen seçilir.



Şekil 4.4. Rulet tekerleği yöntemine göre tasarlanan genetik algoritma şeması



Şekil 4.5. Örnek rulet tekerleği

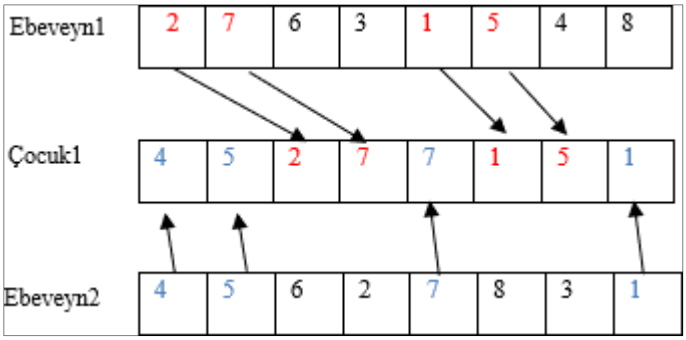
4.1.5. Çaprazlama ve mutasyon

Çaprazlama ve mutasyon işlemleri atama ve sıralama olmak üzere iki başlıkta değerlendirilmektedir. Atama için aşağıdaki metotlar kullanılmıştır.

- Atamaya Dayalı Çaprazlama: Seçilen iki gendeki operasyonlar ebeveynlerden rastgele seçilir.
- Atamaya Dayalı Mutasyon: Ebeveynlerden birinden rastgele seçilen bir operasyon değiştirilir.
- Akıllı Mutasyon: Ebeveynlerden birinden değiştirilecek olan operasyon uygunsa maksimum iş yükü olan makine ile minimum iş yükü olan makine arasında değiştirilir [10].

Operasyonların sıralaması için kullanılan çaprazlama ve mutasyon işlemleri için aşağıdaki metotlar kullanılmıştır.

- Öncelik Koruma Sırasına Dayalı Çaprazlama (ÖKSDÇ) : Bu çaprazlama tekniğinde iki ebeveynden iki çocuk oluşmaktadır. Bir operasyonun ait olduğu işe ait tüm operasyonlar zaman sıralaması korunarak 1. Çocuğu aktarılır. Belirlenen diğer işe ait olan bütün operasyonlar diğer ebeveynden sırası korunarak 1.çocuğu aktarılır. 2.çocuğun sıralaması yapılırken 1. Çocuk için yapılan işlemlerin aynısının tersi olacak şekilde işlemlerin sırası korunarak tekrarlanır. Şekil 4.5'de ÖKSD tekniğinde ebeveynlerdeki genlerin çocuğa sıralı aktarımı gösterilmiştir.



Şekil 4.6. ÖKSDÇ Yönteminde ilk çocuğun oluşumu [10].

Öncelik Korumalı Mutasyon (ÖKM) : Ebebeylerden birinden rastgele seçilen operasyondaki çalışma zamanı sıralama kısıtı dikkate alınarak mümkünse başka bir yerealınır. Buradaki mutasyon işlemi sadece çözüm kalitesini geliştiriyor ise uygulanır.

Monolit mimaride, GA temelli ATÇP çözümü, yazılımın tüm modüllerinin ve hizmetlerinin tek bir kod tabanında birleştirildiği bir yaklaşımdır. Bu bölüm, monolit mimaride GA

uygulamasının tasarım sürecini kapsamlı bir şekilde ele alır. Tasarım süreci, uygulamanın ana bileşenlerini belirlemeyi ve bu bileşenlerin birbirleriyle nasıl etkileşime gireceğini tanımlamayı içerir. GA'nın temel bileşenleri arasında popülasyon, uygunluk fonksiyonu, seçme operatörleri, çaprazlama ve mutasyon operatörleri, ve durma kriteri yer alır.

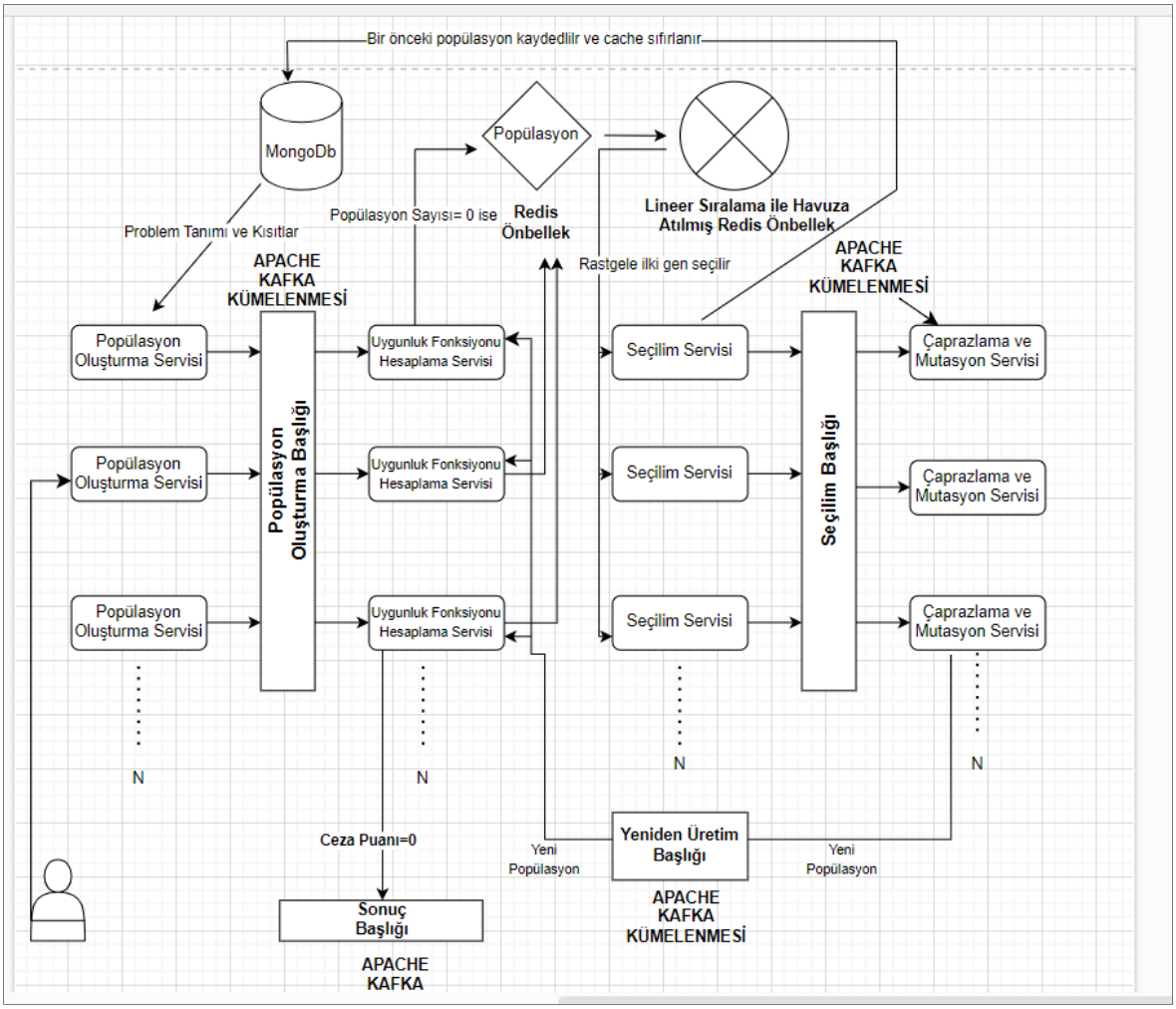
Bu bileşenlerin tasarımı ve uygulaması, ATÇP'nin karmaşıklığını ve GA'nın genel performansını doğrudan etkiler. Örneğin, uygun bir uygunluk fonksiyonu, GA'nın ATÇP çözümünün kalitesini değerlendirme ve en iyi çözümleri seçme yeteneğini belirler. Benzer şekilde, çaprazlama ve mutasyon operatörlerinin etkili tasarımı ve uygulaması, GA'nın çözüm alanında çeşitlilik oluşturma ve yeni çözümler keşfetme yeteneğini belirler.

4.2. Genetik Algoritmanın Mikroservis Mimarisi İle Tasarımı

GA'nın aşamalarının her biri ayrı bir mikroservis olarak tasarlanarak, bu aşamalar aynı anda birden fazla konteynerin çalışması ile paralelleştirilmiştir. Bu aşamaların veri bütünlüğü Redis önbellek altyapısı ile sağlanmıştır. Bu noktada bir servis işlemini bitirdiğinde oluşan sonuçları Apache Kafka altyapısı ile dağıtık bir şekilde algoritmanın diğer aşamasına göndermiştir. Her bir mikroservis Docker ile konteyner haline getirilmiştir.

Burada problemin mikroservislere bölünme yöntemi önemli bir noktadır. Şekil 4.7'de tasarlanan sistemin bütün aşamaları gösterilmiştir. Algoritmanın içerisinde ayrı küçük problemler olarak değerlendirilen servisler şu şekildedir:

- Popülasyon oluşturma servisi
- Uygunluk fonksiyonu hesaplama servisi
- Seçilim Servisi
- Çaprazlama ve Mutasyon Servisi

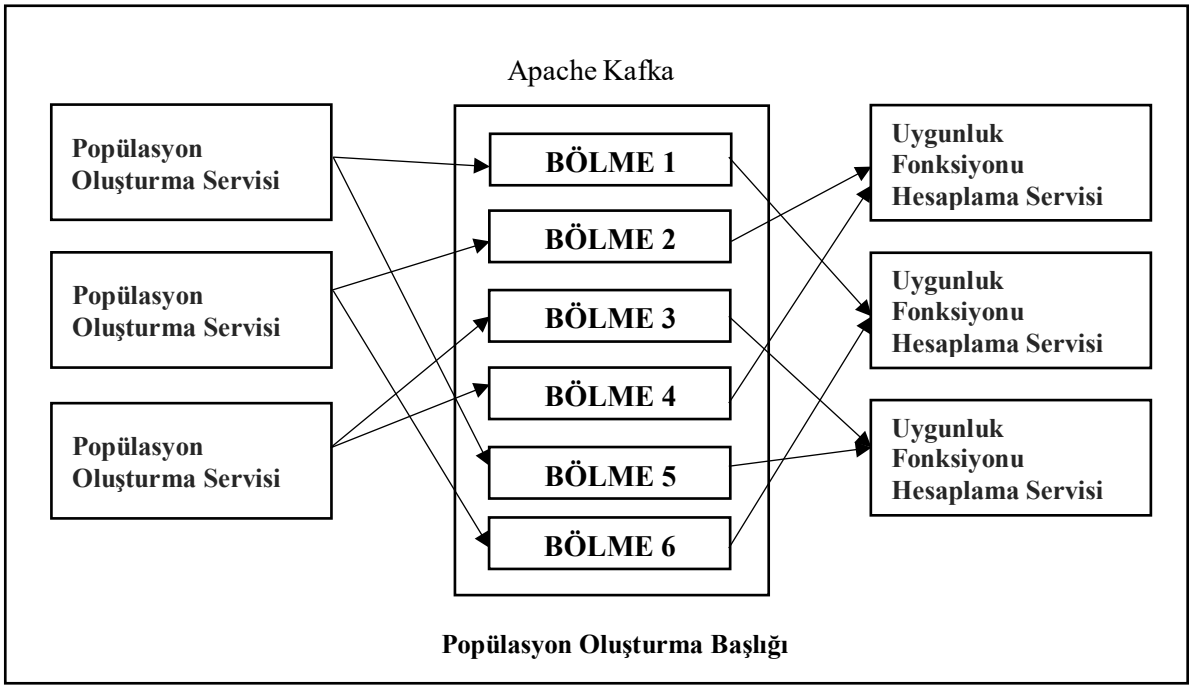


Şekil 4.7. Mikro servis mimarisi ile tasarlanan yapının genel gösterimi

4.2.1. Popülasyon oluşturma servisi

Kullanıcıdan belirli kısıtları RestAPI üzerinden alan bu servis ilk popülasyonun oluşmasını sağlar. Popülasyondaki kromozom sayısı, problemin kaç saatlik dönemde çalışacağı ve her bir operasyonun JSON formatında çalışacağı makine, operasyonun çalışma süresi, önceliği bilgileri kullanıcı tarafından bu servise gönderilir. Servis girilen bilgiler ışığında bireyleri oluşturmaya başlar ve oluşturdukça gerçek zamanlı bir şekilde Apache Kafka üzerinden uygunluk fonksiyonu hesaplama servisine gönderir. Buradaki servislerin hepsi ölçeklendirildiği için oluşturulan her bir birey Apache Kafka'nın yeniden dengeleme prensibine göre aynı anda birçok uygunluk fonksiyonu servisine eş zamanlı olarak gönderilebilir. Bu da aynı anda birçok servisin paralel bir şekilde bireylerin uygunluk fonksiyonunu hesaplamasını sağlamaktadır. Şekil 4.8'de gösterildiği gibi altı bölmeli bir "topic" tasarımı yapılmıştır. Her bir popülasyon

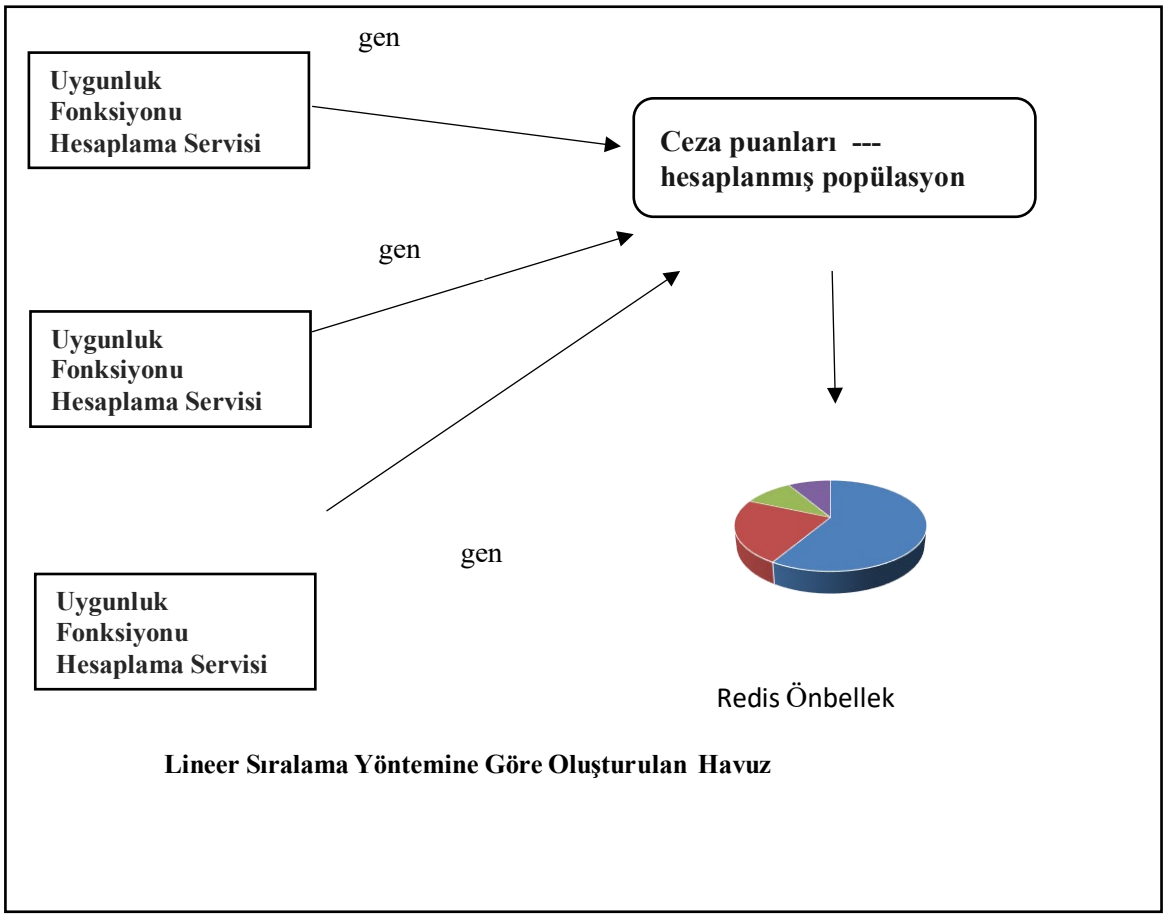
oluşturma servisi eş zamanlı bir şekilde Uygunluk Fonksiyonu Hesaplama servisine eş zamanlı bir şekilde gönderim yapılmıştır.



Şekil 4.8. Popülasyon Oluşturma Servisi ve Uygunluk Fonksiyonu Hesaplama Servisi İletişimi

4.2.2. Uygunluk fonksiyonu hesaplama servisi

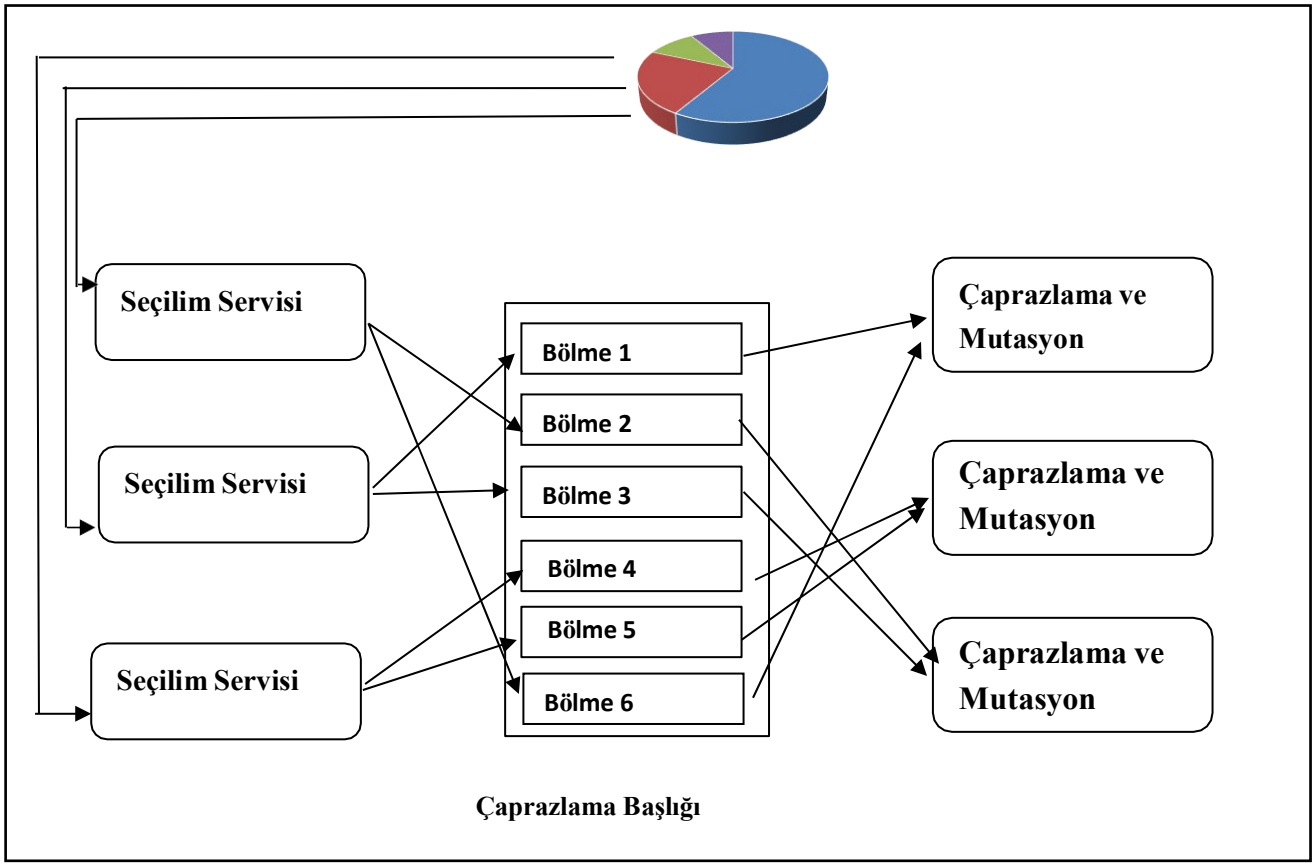
Uygunluk fonksiyonu eş zamanlı olarak gelen her bir bireyin uygunluk fonksiyonunu hesaplayarak Redis Önbellek üzerine yazılmıştır. Burada önbellek yapısının kullanılması diğer veri tabanı sistemlerine göre çok daha fazla hızlı olmasından kaynaklanmaktadır. Önbellek içerisinde bütün bireylerin uygunluk fonksiyonu hesaplandığında lineer sıralama yönteminegöre uygunluk fonksiyonuna göre ters orantılı olarak oluşturulan birey havuzu oluşturulur ve bu da önbelleğe kaydedilir. Şekil 4.9'da gösterildiği gibi uygunluk fonksiyonu hesaplanan her bir birey belirlenen popülasyon sayısına erişilinceye kadar Redis Önbelleğe kaydedilir. Popülasyon sayısına ulaşıldıktan sonra lineer sıralama yöntemine göre oluşturulan popülasyon havuzu ayrı bir veri yapısı olarak Redis Önbelleğe kaydedilir.



Şekil 4.9. Uygunluk Fonksiyonu Hesaplama Servisi Önbellek İlişkisi

4.2.3. Seçilim servisi

Lineer Sıralama yöntemine göre doldurulan rulet Redis önbelleğine atıldıktan sonra, seçilim servisi rastgele iki birey seçerek çaprazlama ve mutasyon servisine göndermektedir. Bu işlem birden fazla seçilim servisi örneğinin ölçeklendirilmiş bir şekilde aynı anda çalışmasıyla havuzdan seçilen rasgele iki gen ile popülasyondaki belirlenen birey sayısı oluşana kadar devam etmektedir. Böylece yeni bir popülasyonun oluşma süreci paralel çalışan servisler ile başlamış olmaktadır. Şekil 4.10'da verildiği gibi popülasyon havuzundan seçilen rastgele iki gen çaprazlama ve mutasyon servisine gönderilmiştir



Şekil 4.10. Mikro servis mimarisi ile tasarlanan yapının genel gösterimi

4.2.4. Çaprazlama ve mutasyon servisi

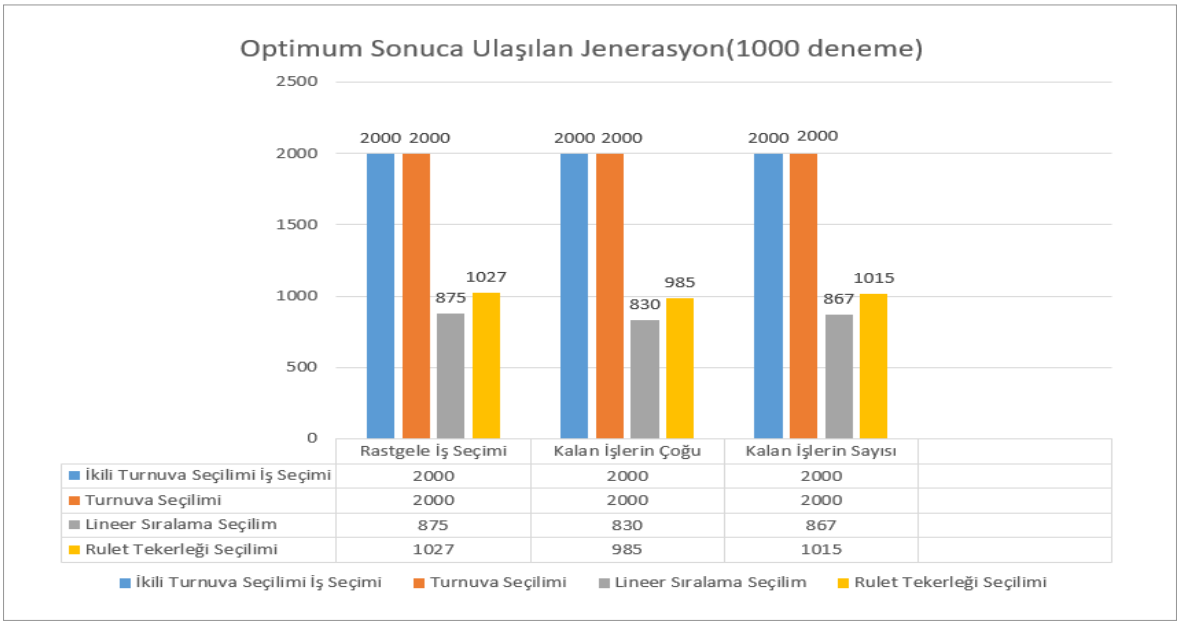
Çaprazlama ve mutasyon servisi ceza puanıyla ters orantılı şekilde lineer sıralama yöntemiyle gen havuzuna atılmış önbellek yapısından rastgele iki gen seçilir. Seçilen genler çaprazlama ve mutasyon işlemlerine sokulur. Oluşan yeni çocuklar yeni popülasyonun oluşturulması için uygunluk fonksiyonu hesaplama servisine gönderilir. Oluşturulan yeni popülasyon belirlenen sayıya ulaştığı zaman bir sonraki popülasyona hazırlanması açısından önbellekten silinir ve veri tabanına kaydedilir.

5. DENEYSEL SONUÇLAR

Bu bölümde tezdeki problemin çözüm zamanını iyileştirmek için kullanılan farklı teknikler analiz edilmiştir. Genetik algoritmalar sezgisel algoritmalar olduğu için algoritmanın tek bir çıktısı bulunmamaktadır. Yani algoritma her çalıştığında farklı çıktılar alınmaktadır. Bu noktada sonuçların daha sağlıklı analiz edilebilmesi için Monte Carlo Simülasyonu tekniğinden yararlanılmıştır. Bu teknik temelde yapılan birçok farklı denemede elde edilen çıktıların ortalamasını alarak özellikle sezgisel algoritmalar için daha doğru sonuçlar bulmayı amaçlamaktadır. Bu çalışma JDK 17 ile 4vCPU AMD A12-9800 Radeon R7 işlemci ve 16 GB RAM bir bilgisayarda yapılmıştır Yapılan çalışmada deneme sayısı 1000 ile sınırlandırılmıştır ve denemelerdeki sonuçların ortalaması alınmıştır. Algoritmanın çalışacağı maksimum jenerasyon sayısı 2000 ile sınırlandırılmıştır. Yani 2000 jenerasyonda da optimum sonucu bulamadığı takdirde algoritma sonlandırılmıştır.

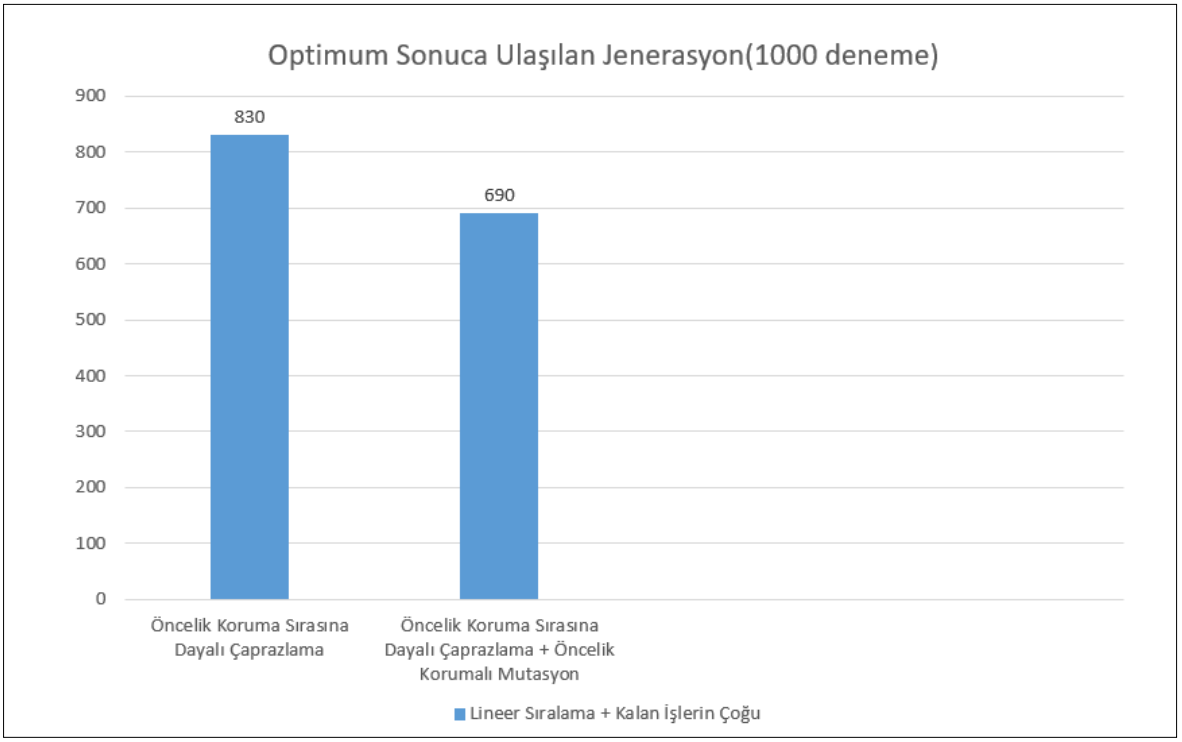
5.1. Monolit Yapı Sonuçları

Monolit yapıda farklı popülasyon oluşturma yöntemleri ile farklı seçilim yöntemlerinin algoritmanın optimumçözümünün bulunma zamanına etkisi Şekil 5.1'te verilmiştir.



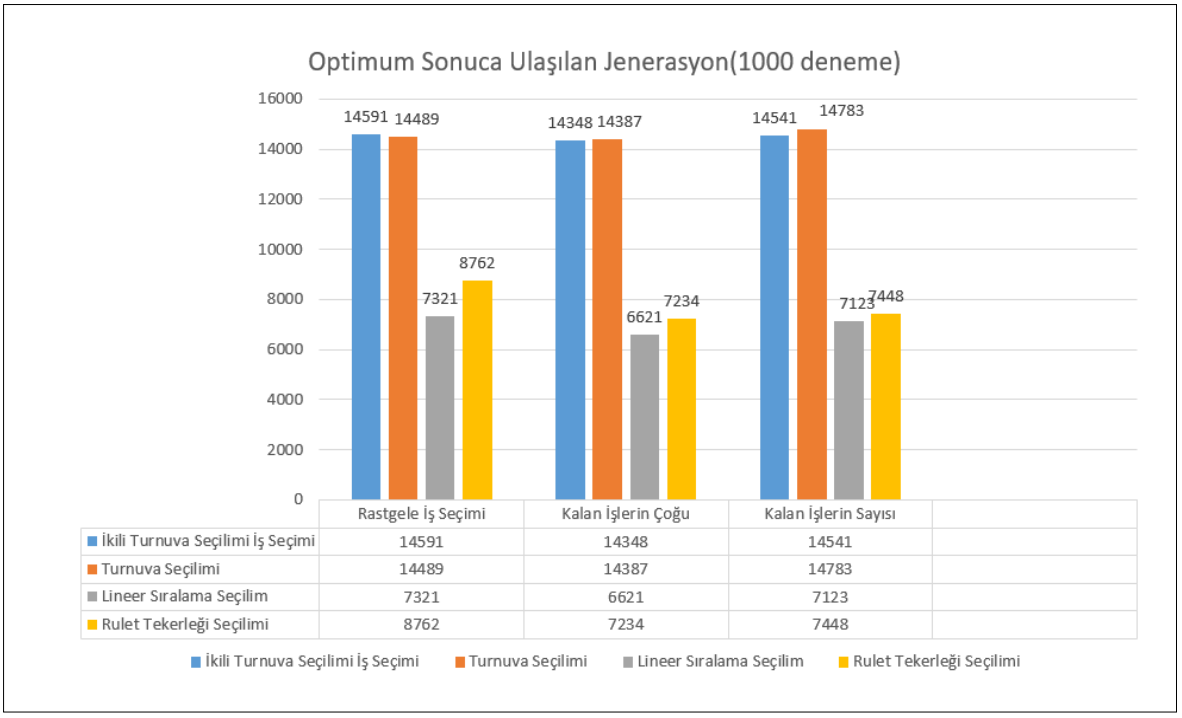
Şekil 5.1. Popülasyon oluşturma ve seçilim yöntemlerinin optimum sonucun bulunduğu ortalama jenerasyon sayısı

Buna ek olarak monolit yapıda ÖKM yönteminin kullanımı sonuca olumlu katkıda bulunmuştur. Şekil 5.2'da gösterildiği gibi en iyi sonucu veren tekniklere ek olarak kullanılan ÖKM çalışma zamanına olumlu etkide bulunmuştur.



Şekil 5.1. ÖKSDÇ ve ÖKM popülasyon oluşturma yöntemlerinin optimum sonucun bulunduğu ortalama jenerasyon sayısı

Algoritmada optimum sonuca ulaşılan jenerasyon sayısına ek olarak optimum sonuca ulaşılma süreside önem arz etmektedir. Şekil 5.3'te kullanılan yöntemlere göre monolit yapıda elde edilen optimum sonuçların çalışma süresi ms cinsinden gösterilmiştir.



Şekil 5.2. Monolitik yapıda popülasyon oluşturma ve seçilim yöntemlerinin çalışma zamanına etkisi

Bu çalışmanın sonuçları, ATÇP için genetik algoritmaların etkili bir çözüm yöntemi olduğunu göstermiştir. Ayrıca, farklı popülasyon oluşturma, çaprazlama ve mutasyon tekniklerinin etkisi karşılaştırılarak, en iyi sonucu veren yöntemler belirlenmiştir. Bu sonuçlar, benzer problemler için çözüm arayışında olan araştırmacıların işlerini kolaylaştıracaktır. Bu çalışmanın kapsamı sınırlıdır ve daha büyük ölçekli ve karmaşık atölye tipi çizelgeleme problemlerinin çözümü için daha farklı yöntemlerin araştırılması gerekmektedir. Özellikle paralel genetik algoritma ve kümeleme algoritmalarının performansı da değerlendirilebilir.

5.2. Mikro Servis Yapı Sonuçları

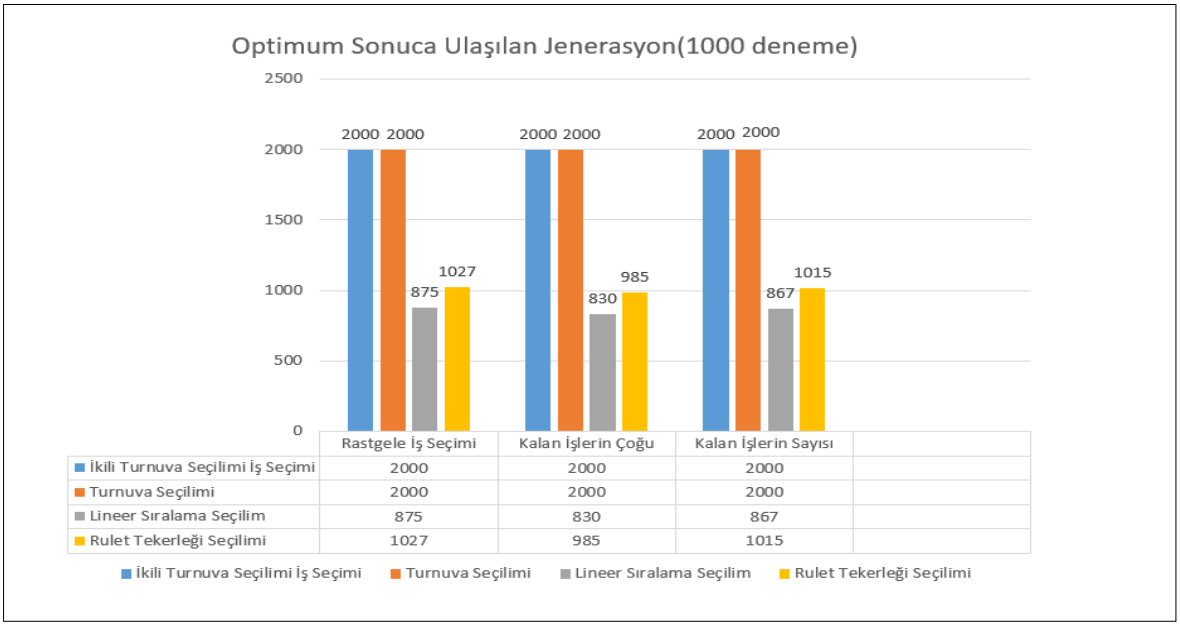
Monolitik yapıda optimum sonuçların bulunduğu yöntemler mikro servis mimarisi ile tasarlanan sistemde de kullanılmıştır. Popülasyon oluşturma yöntemi olarak ÖKSDÇ ve ÖKM seçilim yöntemi olarak lineer sıralama ve çaprazlama yöntemi olarak KİÇ yöntemleri kullanılmıştır. Çünkü önceki bölümde anlatıldığı gibi en iyi sonuçların bu yöntemler kullanılarak alındığı belirlenmiştir. Bu yöntemlerle birlikte algoritmanın her bir aşaması mikro servis mimari üzerinde çalıştırılmıştır. Sistem başlatıldığından itibaren optimum sonuç bulunana kadar her bir mikro servis üzerine düşen görevi yerine

getirme süreleri 1000 denemede ortalama olarak Çizelge 5.1'de gösterilmiştir.

Çizelge 5.1. Her bir mikro servisin ortalama çalışma süresi

Servisler	ÖKSDÇ, ÖKM, KİÇ ve Lineer Sıralama Yöntemleri ile Ortalama Çalışma Süreleri (ms)
Popülasyon Oluşturma Servisi	982ms
Uygunluk Fonksiyonu Hesaplama Servisi	1831ms
Seçilim Servisi	621ms
Çaprazlama ve Mutasyon Servisi	811ms

Monolit yapıda kullanılan teknikler ile optimum sonucun bulunduğu jenerasyon sayısı Şekil 5.4'te gösterilmiştir. Burada mikro servis mimarisinin çalışma süresine ek olarak optimum sonucun bulunduğu jenerasyon sayısında da iyileştirme sağladığı gözlemlenmiştir.



Şekil 5.3. Mikro servis mimarisinde popülasyon oluşturma ve seçilim yöntemlerinin optimum sonucun bulunduğu ortalama jenerasyon sayısına etkisi

6. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında ATÇP için monolitik yapıda genetik algoritma çözümü çalıştırılmıştır. Kullanılan farklı tekniklerin algoritmanın çalışma performansına etkileri incelenmiştir. Sonuç olarak lineer sıralama, KİÇ, ÖKSDÇ ve ÖKM tekniklerinin kullanımı ileen iyi sonucun alındığı tespit edilmiştir. Monte Carlo simülasyon tekniği ile birçok farklı deneme yapıp o denemelerin ortalaması alınarak sonuçların daha güvenilir olması istenmiştir. En iyi sonuçta ortalama 690. jenerasyonda sonucun bulunduğu gözlemlenmiştir. Burada sistemin 2000. jenerasyona kadar doğru sonucu bulamadığı durumlarda ortalamaya dâhil edilmiştir. Yani bazı denemelerde 2000 jenerasyon boyunca optimum sonuç bulunamadığından algoritma o noktada durdurulmuştur. Çalışma zamanı olarak en iyi sonuçta ortalama 6622 saniyede doğru sonuç elde edilmiştir.

Monolitik yapıdaki sonuçlara ek olarak sistem mikroservis mimarisi ile çalıştırıldığında ortalama 564. Jenerasyonda optimum sonucun bulunduğu tespit edilmiştir. Ayrıca çalışma zamanı 4241 saniyede gerçekleşmiştir. Bu sonuçlara göre algoritmanın çalışma hızının arttırılmasının yanı sıra bulunan jenerasyon sayısında da azalma tespit edilmiştir. Algoritmanın çalışma hızında %36 lık bir iyileşme elde edilmiştir.

Algoritmanın aşamalarının ayrı servisler olarak değerlendirilmesi çalışma zamanını çok kısaltsa da servisler arasındaki iletişim, veri tabanı ve önbellek ekleme güncelleme maliyeti algoritmanın çok daha kısa sürede çalışmasının önündeki engellerdir.

Bu tez çalışması meta sezgisel algoritmaların paralelleştirilmesi ve çalışma süresinin azaltılmasında mikro servis mimarisi kullanımın olumlu ve olumsuz yanlarını ortaya koymuştur. Çalışma bu yönüyle değerlendirildiğinde, mikroservis mimarileri kullanılarak meta sezgisel algoritmaların çalışma zamanının hızlandırılması konusunda ileride yapılacak çalışmalar için yol gösterici olabilecektir.

KAYNAKLAR

1. Gerşil, M. , Palamutçuoğlu, T. (2013). Ders çizelgeleme probleminin melez genetik algoritmalar ile performans analizi. *Niğde Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi,* 6(1), 242-262.

2. Wang, Y. (2012). A new hybrid genetic algorithm for job shop scheduling problem. *Computers & Operations Research*, *39*(10), 2291-2299.

3. Xiong, H., Shi, S., Ren, D., Hu, J. (2022). A survey of job shop scheduling problem. *The types and models. Computers & Operations Research*, 142, 105731.

4. Lenstra, J. K., Kan, A. R., Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics* , 1, 343-362.

5. Crowston, W. B., Glover, F., Thompson, G. L.,Trawick, J. D. (1963). Probabilistic and parametric learning combinations of local job shop scheduling rules, *ONR Research Memorandum* , 117, 924-939.

6. Sabuncuoglu, I., Bayiz, M. (1999). Job shop scheduling with beam search, *European Journal of Operational Research*, 118(2), 390-412.

7. Yang, J. H., Sun, L., Lee, H. P., Qian, Y., & Liang, Y. C. (2008). Clonal selection based memetic algorithm for job shop scheduling problems. *Journal of Bionic Engineering*, 5(2), 111-119.

8. Kaya, S. (2014). *Çok amaçlı esnek atölye tipi çizelgeleme problemlerinin geliştirilmiş parçacık sürü optimizasyonu ile çözümüne yönelik model önerileri,* Doktora Tezi, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Kocaeli.

9. Pezzella, F., Morganti, G., Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations research*, *35*(10), 3202-3212.

10. Lee, K. M., Yamakawa, T., Lee, K. M. (1998, 21-23 Nisan). *A genetic algorithm for general machine scheduling problems*. Second International Conference. Knowledge-Based Intelligent Electronic Systems. Adelaide, 60-65.

11. Wang, Y. (2012). A new hybrid genetic algorithm for job shop scheduling problem. *Computers & Operations Research*, *39*(10), 2291-2299

12. Di Martino, S., Ferrucci, F., Maggio, V., Sarro, F. (2013). Towards migrating genetic algorithms for test data generation to the cloud. *Software Testing in the Cloud: Perspectives on an Emerging Discipline* 6, 113-135.

13. Jin, C., Vecchiola, C., Buyya, R. (2008, 7-12 Aralık). *MRPGA: an extension of MapReduce for parallelizing genetic algorithms.* 4. International Conference on eScience, Indiana , 214-221.

14. Salza, P. , Ferrucci, F., Sarro F. (2016). *elephant56: Design and implementation of a parallel genetic algorithms framework on hadoop MapReduce*., Genetic and Evolutionary Computation Conference Companion, New York , 1315-1322.

15. Verma, A., Llora, X., Goldberg, D. E., Campbell, R. H. (2009, 30 Kasım – 2 Aralık). *Scaling genetic algorithms using mapreduce*, *9.* International Conference on Intelligent Systems Design and Applications, Pisa, 13-18.

16. García-Valdez, M., Merelo, J. J. (2017,15-17 Temmuz). *Evospace-js: Asynchronous pool-based execution of heterogeneous metaheuristics*. Genetic and Evolutionary Computation Conference Companion*,* Berlin, 1202-1208.

17. Roy, G., Lee, H., Welch, J. L., Zhao, Y., Pandey, V., Thurston, D. (2009, 18-21 Mayıs). *A distributed pool architecture for genetic algorithms*, IEEE Congress on Evolutionary Computation , Trondheim, 1177-1184

18. Talukdar, S., Baerentzen, L., Gove, A., De Souza, P. (1998). Asynchronous teams: Cooperation schemes for autonomous agents. *Journal of Heuristics*, 4, 295-321.

19. Khalloof, H., Jakob, W., Liu, J., Braun, E., Shahoud, S., Duepmeier, C., Hagenmeyer, V. (2018, 15-19 Temmuz). *A generic distributed microservices and container based framework for metaheuristic optimization*, Proceedings of the genetic and evolutionarycomputation conference companion*,* Kyoto, 1363-1370.

20. Stevanoska, E., Spirovski, K., Petkovski, G., Jakimovski, B.,Velinov, G. ,(2018, 20-22 Nisan) *Microservice based architecture for the genetic algorithm*, 15th International Conference for Informatics and Information Technology*,* Mavrovo, 43-48.

21. Klock, S., Van Der Werf, J. M. E., Guelen, J. P., Jansen, S. (2017, 3-7 Nisan). *Workload-based clustering of coherent feature sets in microservice architectures*, 2017 IEEE International Conference on Software Architecture*,* Gothenburg, 11-20.

22. İnternet: Kumar, M., Husain, D. M., Upreti, N., Gupta, D. (December 2010). Geneticalgorithm: Review and application, URL: https://ssrn.com/abstract=3529843 / Son Erişim Tarihi: 04.08.2023

23. Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2), 88-105.

24. Emel, G. G., Taşkın, Ç. (2002). Genetik algoritmalar ve uygulama alanlari. *Uludağ Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, 21(1), 129-152.

25. İnternet: Genetik algoritmalar: Değişime uyum sağlamanın önemi, URL: https://www.atp.com.tr/genetik-algoritmalar-degisime-uyum-saglamanin-onemi-ahmet- tugrul-bayrak /Son Erişim Tarihi: 02.06.2023

26. Larrucea, X., Santamaria, I., Colomo-Palacios, R., Ebert, C. (2018). Microservices. *IEEE Software*, *35*(3), 96-100.

27. Cerny, T., Donahoo, M. J., & Trnka, M. (2018). Contextual understanding of microservice architecture: current and future directions. *ACM SIGAPP Applied Computing Review,* 17(4), 29-45.

28. İnternet: Apache Kafka Tutorial for Beginners, URL: https://howtodoinjava.com/kafka/tutorial-introduction/ Son Erişim Tarihi: 04.06.2023

29. İnternet: REST API (RESTful API) nedir? , URL: https://medium.com/mobillium/rest- api /Son Erişim Tarihi: 04.06.2023

30. Potdar, A. M., Narayan, D. G., Kengond, S., & Mulla, M. M. (2020). Performance evaluation of docker container and virtual machine. *Procedia Computer Science*, 171, 1419-1428.

31. İnternet: Where can a developer data platform take you?, , URL: https://www.mongodb.com/ , Son Erişim Tarihi: 04.06.2023

32. İnternet: Get started with Redis, URL: https://redis.io/docs/getting-started/ Son Erişim Tarihi: 04.07.2023

33. İnternet: The Job Shop Problem , URL: https://developers.google.com/optimization/scheduling/job_shop?hl=tr /Son Erişim Tarihi: 04.06.2023

34. Kacem, I., Hammadi, S., 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 (Applications and Reviews)*, 32(1), 1-13.

Gazili olmak ayrıcalıktır