TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GÜVENLİK FARKINDALIKLI VERİTABANI GÖÇÜ PLANLAMASI

YÜKSEK LİSANS TEZİ

Utku Umur AÇIKALIN

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Dr. Öğr. Üy. Buğra ÇAŞKURLU

AĞUSTOS 2022

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

Utku Umur AÇIKALIN

İMZA

ÖZET

Yüksek Lisans Tezi

GÜVENLİK FARKINDALIKLI VERİTABANI GÖÇÜ PLANLAMASI

Utku Umur AÇIKALIN

TOBB Ekonomi ve Teknoloji Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Dr. Öğr. Üy. Buğra ÇAŞKURLU

Tarih: AĞUSTOS 2022

İçinde bulunduğumuz Büyük Veri çağında birçok firma devasa boyutlarda verilerle çalışmaktadır. Veritabanı göçü, büyük veri üstüne çalışan firmalar için önemli bir problemdir. Veritabanı göçünün, servis kalitesi gereksinimleri nedeniyle tek seferde gerçekleştirilmesi çoğu zaman mümkün değildir. Bu yüzden, firmalar veritabanı göçünü parti adı verilen birden çok kısa zaman aralığında gerçekleştirmeyi tercih etmektedirler. Veritabanı göçü maliyetli bir işlem olmasının yanı sıra ciddi güvenlik riskleri de içermektedir. Bu nedenle, veri göçü işleminde, bazı firmalar uygulama testlerinden kaynaklanan maliyeti düşürmeyi amaçlarken, göç ettirilecek verinin gizli veya hassas bilgiler içerdiği durumlarda firmalar güvenlik risklerini azaltmayı amaçlamaktadırlar. Literatürde veritabanı göçü planlaması problemi uygulama test maliyetinin azaltılması yönünden ele alınmıştır. Bu çalışma ise uygulama test maliyetine ortogonal bir metrik olan güvenlik riskinin azaltılmasına odaklanmaktadır. Veri ne kadar uzun süre erişme açık kalırsa, o kadar büyük bir güvenlik ihlal riski oluşturmaktadır. Dolayısıyla bu tezde ele alınan veritabanı göçü planlaması probleminde güvenlik riski göçün tamamlanması gereken parti sayısıyla ilişkilendirilmektedir. Buna bağlı olarak, güvenlik riskinin minimize edilmesi için veri göçü en az sayıda parti kullanılarak tamamlanmalıdır. Bu çalışmada veritabanı göçü planlaması probleminin güvenlik riskinin minimize edilmesi yönünden ele alınması için teorik bir problem çatısı tanımlanmaktadır. Bu çatı firmaların farklı ihtiyaçları göz önünde bulundurularak tasarlanmıştır ve toplamda 24 farklı modelden oluşmaktadır. Tanımlanan çatıdaki 24 problem modelinden 23 tanesinin hesaplama karmaşıklığı tespit edilmiştir. Bu modellerin 16 tanesinin **NP-zor** olduğu, 7 tanesinin ise polinom

zamanda çözülebildiği tespit edilmiştir. Ayrıca **NP-zor** modellerden 2 tanesinin asimptotik tamamen polinom zamanlı yakınsama şemasına sahip olduğu, 12 tanesinin ise polinom zamanlı yakınsama şemasına sahip olmadığı ispatlanmıştır. **NP-zor** modeller için kurucu sezgisel algoritmalar, yerel arama algoritmaları ve bunları kullanan memetik algoritmalar geliştirilmektedir. Geliştirilen memetik algoritmalar problem spesifik 2 çaprazlama ve 2 mutasyon operatörü kullanmaktadır. Geliştirilen memetik algoritmalar 8 saniyeden az bir sürede, örneklerin %95'inde optimalden en fazla %10 uzaklıkta çözümler bulmakta ve örneklerin %72,1'ini optimal çözmektedir. Ayrıca, memetik algoritmalar 82 saniyeden kısa bir sürede, örneklerin %95'inde optimalden en fazla %7 uzaklıkta çözümler bulmakta ve örneklerin %74,5'ini optimal çözmektedir.

Anahtar Kelimeler: Veritabanı göçü, Çizelgeleme, Memetik algoritmalar, Sezgisel algoritmalar.

ABSTRACT

Master of Science

SECURITY AWARE DATABASE MIGRATION PLANNING

Utku Umur AÇIKALIN

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Dr. Öğr. Üy. Buğra ÇAŞKURLU

Date: August 2022

In the age of Big Data, many companies work with a huge amount of data. Database migration is an important problem faced by companies dealing with big data. Not only is migration a costly procedure, it involves serious security risks as well. Due to the quality-of-service requirements, it is often not possible to migrate all databases at once. Therefore, companies prefer to perform database migration in multiple short time intervals called shifts. For some institutions, the primary focus is on reducing the cost of the migration operation, which manifests itself in application testing. For other institutions, minimizing security risks is the most important goal, especially if the data involved is of a sensitive nature. In the literature, the database migration problem has been studied from a test cost minimization perspective. This thesis focuses on an orthogonal measure, i.e., security risk minimization. The security risk is associated with the number of shifts needed to complete the migration task. Ideally, the migration should be completed in as few shifts as possible, so that the risk of data exposure is minimized. In this thesis, a formal framework for studying the database migration problem from the perspective of security risk minimization is provided. The framework consists of 24 models and it is designed so that it can take the different needs of the companies into account. The computational complexity of 23 of the 24 models is established. 16 of the models in the framework are **NP-hard**, and 7 of the models can be solved in polynomial time. Moreover, it is proven that 2 of the **NP-hard** models admit an asymptotic fully polynomial time approximation scheme while 12 of the **NP-hard** models does not admit a polynomial time approximation scheme. For the **NP-hard** models, several constructive heuristic and local search algorithms are

designed, and memetic algorithms that employ these algorithms are developed. The memetic algorithms use 2 problem-specific crossover and 2 mutation operators. These memetic algorithms produce solutions that are within 10% in 95% of the instances and solve 72.1% of the instances optimally under 8 seconds. Furthermore, developed algorithms produce solutions that are within 7% in 95% of the instances and solve 74.5% of the instances optimally under 82 seconds.

Keywords: Database migration, Scheduling, Memetic algorithms, Heuristic algorithms.

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren, bilgilendiren ve bağımsız bir araştırmacı olmama çok büyük bir katkısı olan hocam Dr. Öğr. Üy. Buğra Çaşkurlu'ya, kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine, vermiş olduğu Sezgisel Arama Metotları dersi ile beni sezgisel algoritmarlarla tanıştırmış olan ikinci tez danışmanım Dr. Öğr. Üy. Eda YÜCEL hocama, sundukları üretken ve keyifli çalışma ortamı için Teorik Bilgisayar Bilimleri Laboratuvarı'ndaki çalışma arkadaşlarıma, destekleriyle her zaman yanımda olan annem Gülçin AYTAN AÇIKALIN'a, babam Osman AÇIKALIN'a ve nişanlım Damla HADDUR'a teşekkürlerimi sunarım. Ayrıca, yüksek lisans eğitimim boyunca çalışmalarımı destekleyen TÜBİTAK 2210 Burs Programına içten teşekkürlerimi sunarım.

İÇİNDEKİLER

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

KISALTMALAR

ATPZYŞ : Asimptotik Tamamen Polinom Zamanlı Yakınsama Şeması
GFVGP : Güvenlik Farkındalıklı Veritabanı Göçü Planlaması
İS : İlk Sığan
İSA : İlk Sığan Azalan
İSAPA : İlk Sığan Azalan Partiler Azalan
İSAPK : İlk Sığan Azalan Partiler Karışık
PZYŞ : Polinom Zamanlı Yakınsama Şeması
STZİ : Son Teslim Zamanı İlk
TİS : Topolojik İlk Sığan
TPZYŞ : Tamamen Polinom Zamanlı Yakınsama Şeması
VGP : Veritabanı Göçü Planlaması

SEMBOL LİSTESİ

Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	Açıklama
α	Veritabanlarının boyutlarını gösteren parametre
β	Partilerin kapasitelerini gösteren parametre
γ	Veritabanları arasındaki ilişkileri gösteren parametre
θ	Amaç fonksiyonunu gösteren parametre

1. GİRİŞ

Büyük Veri çağını yönlendiren birçok faktör bulunmaktadır. Nesnelerin İnterneti paradigması kapsamında etrafa toplu bir şekilde yayılan veya dağıtılan çok sayıda sensör muazzam boyutlarda veri üretmektedir [1]. Video kameralarının, gözlem aygıtlarının ve akıllı cihazların benimsenmesi devasa boyutlarda verilerin üretildiği akıllı şehirleri ortaya çıkardı [2]. Sağlık sektöründe, her yıl milyarlarca radyolojik görüntü üretilmekte, saklanmakta ve analiz edilmektedir [3]. Bu alanlara ek olarak, kullanıcılar tarafından sürekli oluşturulmaya devam edilen bir içerik dalgası bulunmaktadır. Gerçekten de bu yıkıcı teknolojiler veriyi nasıl sakladığımızı, işlediğimizi ve analiz ettiğimizi değiştirmeye başladılar. Ortaya çıkan bu bilgi tufanı, verilerin büyük oranda bulut bilişim altyapılarında saklanması ve işlenmesi ile kontrol altına alındı [4].

Veritabanı göçü, büyük veri üstüne çalışan firmalar için önemli bir problemdir. Veritabanı göçü maliyetli bir işlem olmakla birlikte ciddi güvenlik riskleri içermektedir. Bu nedenle, veri göçü işleminde, bazı firmalar uygulama testlerinden kaynaklanan maliyeti düşürmeyi amaçlarken, göç ettirilecek verinin gizli ve/veya hassas bilgiler içerdiği durumlarda güvenlik risklerinin minimize edilmesi amaçlanmaktadır.

Veri göçü, depolama türleri, biçimleri veya bilgisayar sistemleri arasındaki veri aktarma işlemi olarak tanımlanmaktadır [5, 6]. Kuruluşlar veya bireyler bilgisayar sistemlerini değiştirdiklerinde veya güncelliklerinde ya da iki firmanın birleşmesi sonucu bilgisayar sistemlerinin birleştirmesi gerektiğinde veri göçüne ihtiyaç duyulur.

Bu süreçte, yazılım çözümleri veri göçü işlemlerini otomatikleştirdiği [7] ve dolayısıyla da süreçteki insan gereksinimi azalttığı [8] için tercih edilmektedir [9]. Veri göçü işlemi [10, 11, 6] birden çok aşamadan oluşabilir fakat minimal olarak şu iki aşamayı içerir [12, 13] : *veri çıkarımı* [14] ve *veri yükleme* [15]. Veri göçünün efektif bir şekilde gerçekleştirilebilmesi için verinin eski sistemden yeni sisteme eşleştirilmesi gerekir [16]. Veri çıkarımı [17] ve veri yükleme [18] prosedürü, bu eşleşmeye uygun olacak şekilde belirlenir. Özellikle yeni sistemin, kullanılan sistemin güncellemesinden ziyade başka bir firmanın yazılımı olduğu durumlarda, bu prosedür karmaşık veri dönüşümleri içerir [19].

Depolama göçü [20, 21], en sık gerçekleştirilen veri göçü türüdür ve genellikle firmalar daha verimli depolama teknolojilerine geçiş yapmak istediklerinde gerçekleştirilir. Bunun sonucunda, veri bulunduğu diskten daha efektif bir depolama teknolojisine taşınır. Aynı veritabanı sistemi (ve dosya sistemi) kullanıldığı için genellikle verinin formatı ve içeriği değişmemektedir.

Veritabanı göçü operasyonu, firmaların farklı bir veritabanı sağlayıcısına geçiş yapmaları durumunda ya da kullandıkları veritabanı yazılımının güncellenmesi

sonucunda gerçekleştirilir [22, 23]. İlk durumda fiziksel veri göçü işlemine ihtiyaç duyulurken, ikinci durumda genellikle fiziksel veri göçü işlemine ihtiyaç duyulmaz. Veritabanı göçlerinde, göç ettirilen veritabanlarını kullanan uygulamaların nasıl etkileneceği de göz önüne alınmalıdır.

Kullanılan veri işlemi dili veya protokolü değiştiğinde, uygulamanın çalışması esnasında istenmeyen davranışlar veya uygulama performansında beklenmeyen değişimler gözlemlenebilir. Bu sebeple, göç ettirilen veritabanlarını kullanan tüm uygulamalar test edilmelidir [24]. Literatürde, veritabanı göçü planlaması problemi test maliyetlerinin minimize edilmesi açısından ele alınmıştır [24, 25]. Bu tezde ise test maliyetine ortogonal bir etkiye sahip olan güvenlik risklerinin minimize edilmesine odaklanılmaktadır [26]. Veritabanı göçü, hizmet kalitesi gereksinimleri nedeniyle çoğu zaman tek seferde gerçekleştirilmez. Bunun yerine, her birini parti olarak isimlendirdiğimiz, birden çok kısa zaman aralığında gerçekleştirilir. Veri ne kadar uzun süre erişme açık kalırsa, o kadar büyük bir güvenlik ihlal riski oluşturmaktadır. Dolayısıyla bu tezde ele alınan veritabanı göçü planlaması probleminde güvenlik riski göçün tamamlanması gereken parti sayısıyla ilişkilendirilmektedir. Bu ilişkiye bağlı olarak, güvenlik riskinin minimize edilmesi için veri göçü en az sayıda parti kullanılarak tamamlanmalıdır. Aşağıda, tez boyunca kullanılan notasyon verilecek ve Güvenlik Farkındalıklı Veritabanı Göçü Planlaması problemi matematiksek olarak tanımlanacaktır.

1.1 Notasyon ve Güvenlik Farkındalıklı Veritabanı Göçü Planlaması Probleminin Tanımı

Tezde ele alınan problem, Güvenlik Farkındalıklı Veritabanı Göçü Planlaması, kısaca GFVGP, problemi olarak tanımlanmıştır. GFVGP probleminde, B_i, i. veritabanını göstermek üzere, $\mathscr{B} = \{B_1, B_2, \ldots, B_n\}$ veritabanları kümesini, w_i, i. veritabanının boyutunu göstermek üzere, $w = [w_1, w_2, \ldots, w_n]^T$ ise veritabanları boyut vektörünü göstermektedir. Veritabanları partiler halinde göç ettirilir. Veritabanları partilerle eşleştirilerek, aynı partiye atanan bütün veritabanları aynı anda göç ettirilmektedir. Bir partide taşınabilecek toplam veritabanı boyutuna partinin kapasitesi denilmektedir. l_i, i. partinin kapasitesini göstermek üzere, $l = [l_1, l_2, \ldots, l_n]$ parti kapasite vektörüdür. En kötü durumda, tüm veritabanlarının farklı partilerde taşınması gerekebilir.

Her veritabanı B_i ile ilişkilendirilmiş bir planlama zamanı değişkeni s_i bulunmaktadır ve B_i'nin çözümde hangi partide taşınacağını belirtir. Veritabanları arasında atanabilecekleri partileri kısıtlayabilecek zamansal ilişkiler bulunabilmektedir. Tipik kısıtlara ilişkin şu örnekler verilebilir:

- Veritabanı B_5, ilk dört partiden birinde taşınmalıdır. Bu kısıt, $s_5 \leq 4$ şeklinde ifade edilebilir.
- Veritabanı B_2, ilk dört partiden birinde taşınmamalıdır. Bu kısıt, $s_2 \geq 5$ şeklinde ifade edilebilir.
- Veritabanı B_4, veritabanı B_3'ten önce taşınmalıdır. Bu kısıt, $s_4 < s_3$ şeklinde ifade edilebilir.

- Veritabanı B_4, veritabanı B_3'ten daha geç taşınmamalıdır. Bu kısıt, $s_4 \leq s_3$ şeklinde ifade edilebilir.
- Veritabanı B_4 ve B_5 aynı partide taşınmalıdır. Bu kısıt, $s_4 \geq s_5$ ve $s_5 \geq s_4$ şeklinde ifade edilebilir.

Zamansal ilişkiler de girdinin bir parçası olup, m elemanlı C listesiyle gösterilmektedir. İlk iki örnekteki gibi tek değişken içeren kısıtlara *kesin* kısıtlar, son üç örnekteki gibi iki değişken içeren kısıtlaraysa *göreceli* kısıtlar denilmektedir. Modellerde k bir tam sayı olmak üzere, $k \leq s_i$ ve $s_i \leq k$ şeklinde kesin kısıtlara izin verilmektedir. Göreceli kısıtlardaysa $s_i \geq s_j$ şeklinde gevşek eşitsizliklere ve $s_i > s_j$ şeklinde sıkı eşitsizliklere izin verilmektedir.

Yukarıda da anlatıldığı üzere GFVGP probleminin girdisi, $\langle \mathbf{w}, \mathbf{l}, \mathbf{C} \rangle$ üçlüsüdür.

Aşağıdaki Örnek 1, GFVGP problemi için bir girdi örneği içermektedir.

Örnek 1:

$$\langle \mathbf{w} = [\mathbf{1},\mathbf{2},\mathbf{2},\mathbf{3}]^{\mathbf{T}},\ \mathbf{l} = [\mathbf{5},\mathbf{7},\mathbf{10},\mathbf{12}]^{\mathbf{T}},\ \mathbf{C} = [\mathbf{s_1} < \mathbf{s_2}, \mathbf{s_2} \leq \mathbf{s_3}, \mathbf{s_4} \leq \mathbf{3}, \mathbf{s_4} \geq \mathbf{3}] \rangle.$$

Bu girdi örneğinde, $1,2,2$ ve 3 boyutlarında 4 tane veritabanı bulunmaktadır. l vektörü partilerin kapasitelerini göstermektedir. Dolayısıyla partilerde taşınabilecek toplam veritabanı boyutları sırasıyla $5,7,10$ ve 12'dir. C listesi 4 tane zamansal kısıt içermektedir. İlk kısıt sıkı göreceli zaman kısıtıyken, ikinci kısıt bir gevşek göreceli zaman kısıtıdır. Üçüncü ve dördüncü kısıtlarsa kesin zaman kısıtlarıdır ve beraber veritabanı B_4'ün 3. partide taşınması gerektiğini belirtirler. $S = [s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 3]$, Örnek 1'in olası bir çözümüdür. Bu çözümde B_1 veritabanı 1. partiye, B_2 ve B_3 veritabanları 2. partiye, B_4 veritabanı ise 3. partiye atanmıştır. Görülebileceği üzere S çözümünde, partilere atanan toplam veritabanı boyutları sırasıyla, 1, 4, 3 ve 0'dır ve dolayısıyla hiçbir parti kapasitesi ihlal edilmemiştir. Benzer bir şekilde hiçbir zaman kısıtı da ihlal edilmemiştir. Dolayısıyla S çözümü, Örnek 1 için olurlu bir çözümdür.

Bu problem girdilerin yapılarına bağlı olarak, gerçek hayatta farklı GFVGP problemi modelleri ile karşılaşılmaktadır. GFVGP probleminin temelde dört boyut üzerinde farklılaştığı görülmektedir: (i) veritabanlarının boyutları, (ii) parti kapasiteleri, (iii) veritabanları arasındaki ilişkiler, (iv) amaç fonksiyonu. Bu dört boyutu dikkate alacak şekilde GFVGP problem çatısı aşağıdaki gibi tanımlanmıştır.

1.2 Problem Modeli Çatısı

(i) Veritabanlarının Boyutları (α) - Bir veritabanının göç etme süresi boyutuyla pozitif korelasyona sahiptir. Çünkü veritabanı göçü işleminde veritabanı orijinal konumdan okunmalı, yeni konuma yazılmalı ve orijinal konumdan silinmelidir. Dolayısıyla, veritabanlarının boyutları, partilerde taşınabilecek veritabanı sayısını etkilemektedir. Fakat veritabanlarının boyutlarının birbirlerine çok yakın olduğu durumlar olabilir. Bu durumlarda veritabanları boyutlarının eşit kabul edilmesi problemi kolaylaştırılabilmektedir. Dolayısıyla, veritabanları boyutları için iki farklı modeli ele alınmaktadır.

(a) Sabit (*sbt*) - Bu modelde, bütün veritabanlarının boyutları eşittir. Bu model, aşağı yukarı aynı boyutta veritabanlarına sahip şirketlerin durumlarını modellemektedir.

(b) Keyfi (*kyf*) - Bu modelde farklı veritabanlarının boyutları arasında bir ilişki yoktur. Bu model farklı boyutlarda veritabanlarına sahip şirketlerin durumlarını modellemektedir.

(ii) Partilerin Kapasiteleri (β) - Veritabanı göçü işlemi esnasında fiziksel olarak taşınan veritabanları erişime kapatılmaktadır. Dolayısıyla bankalar gibi mesai saatlerinde yoğun hizmet veren şirketler, veritabanı göçü operasyonunu hafta sonları veya hafta içi akşam saatlerinde gerçekleştirmeyi tercih edeceklerdir. Facebook ve Youtube gibi dünyanın her tarafında ve dolayısıyla farklı zaman dilimlerinde yaşayan kullanıcılara sahip ve haftanın her günü yoğun olarak veritabanlarına erişilen şirketler için hiçbir zaman dilimi diğer bir zaman dilimine göre veritabanı göçü için daha uygun değildir. Dolayısıyla bu iki durumu modellemek için parti kapasitelerine ilişkin aşağıdaki iki model kullanılmaktadır.

(a) Sabit (*sbt*) - Bu modelde bütün parti kapasiteleri bir L tam sayısına eşittir. Bu model Facebook ve Youtube gibi veritabanı erişim oranının homojen dağıldığı şirketlere daha uygundur.

(b) Keyfi (*kyf*) - Bu modelde parti kapasiteleri arasında bir ilişki yoktur. Bu model veritabanı erişim oranının bankalar gibi heterojen dağıldığı şirketlere daha uygundur.

(iii) Veritabanları arasındaki ilişkiler (γ) - Farklı veritabanlarının sakladıkları veriler birbirlerini tamamlayıcı ya da ikame edici nitelikte olabilir. Kullanıcılar tamamlayıcı nitelikteki veritabanlarına eş zamanlı olarak erişim yaptıklarından bu nitelikteki veritabanlarının aynı partide göç etmesi hizmet kalitesi açısından daha uygun olacaktır. Youtube gibi kullanıcıların yüklediği videoları saklayan şirketlerde ise birbirini ikame edici nitelikte içeriklere sahip veritabanları bulunmaktadır. Tipik bir Youtube kullanıcısı, aradığı anahtar kelimelerle alakalı videolar izlemek isteyecektir. Hizmet kalitesi gereği benzer içeriklere sahip farklı veritabanlarının aynı zamanda erişilmez olmaması ve dolayısıyla farklı partilerde göç etmeleri gerekmektedir. Veritabanları arasındaki zamansal kısıtlara ilişkin aşağıdaki üç model kullanılmaktadır.

(a) Yok (ϕ) - Bu model hiçbir zamansal kısıtın olmadığı, yani, her veritabanının herhangi bir partide göç edebildiği durumları modellemektedir.

(b) Kesin (*ksn*) - Bu modelde, sadece $s_i \leq 4$ gibi tek değişkenden oluşan kesin zamansal kısıtlar yer alır.

(c) Görece (*grc*) - Bu modelde hem kesin hem de göreceli (iki değişkenden oluşan) zamansal kısıtlar bulunabilir ve bu üç modelin en genel halidir. Örnek olarak B_i ve B_j veritabanlarının içerikleri birbirini tamamlayıcı nitelikte ise $s_i \leq s_j$ ve $s_i \geq s_j$ kısıtları, birbirini ikame edecek nitelikteyse $s_i < s_j$ veya $s_j < s_j$ kısıtlarından biri tercih edilebilir.

(iv) Amaç Fonksiyonu (Θ) - Şirketlerin farklı önceliklerine ilişkin aşağıdaki iki amaç fonksiyonu modeli kullanılmaktadır.

(a) Kullanılan parti sayısı (*kul*) - Kullanılan parti sayısını minimize etmeyi amaçlayan bu model, veri güvenliği savunmasızlığının azaltması için önerilmiştir. Veritabanı göçü esnasında, teknik personelin taşınacak tüm veritabanlarına erişimi olacaktır. Teknik personel veritabanı göçü operasyonunu mümkün olan en yüksek güvenlik önlemleriyle gerçekleştirmek için uygun eğitime sahip olsa bile, gerçekleşen operasyon sonuncunda işlem rutin hale gelecektir. Bu durum gerçekleştiğinde, personelin dikkatsizleşmesi ve veritabanı göçünün güvenlik önemlerini tam anlamıyla uygulamaması potansiyel bir risk haline gelecektir. Hatayla da olsa veri sızmalarının ciddi sonuçları olacağı için veritabanı göçü sık sık tekrarlanmamalıdır. Bu amaç fonksiyonu, potansiyel kötü sonuçları göç sayısını minimize ederek engellemeyi amaçlar. Bu modelde, eğer 1. ve 3. partilere veritabanı atanırsa fakat 2. partiye atanmazsa kullanılan parti sayısı 2 olduğu için amaç fonksiyonu değeri de 2 olacaktır.

(b) Son kullanılan parti numarası (*son*) - Bu amaç fonksiyonu donanım kaynaklı güvenlik zayıflıklarının olduğu durumlarda göç işlemini en kısa zamanda bitirmek isteyen firmalar göz önünde bulundurularak tasarlanmıştır. Bu amaç fonksiyonu, veritabanı göçü planlaması tamamlanıncaya kadar teknik personelin veriye erişiminin devam etmesi, farklı iki lokasyonda veya platformda verinin bulundurulması ve dolayısıyla iki tarafa da para ödendiği durumlarda da firmalar tarafından tercih edilebilmektedir. Böyle durumlarda firmalar partileri kullansalar da kullanmasalar da bu partiler için ücretlendirilmekte ya da güvenlik açığına maruz kalmaya devam etmektedirler. Bu doğrultuda amaç veritabanı göçünü mümkün olduğunca hızlı bir şekilde tamamlamaktır. Bu modelde, eğer 1. ve 3. partilere veritabanı atanırsa fakat 2. partiye atanmazsa kullanılan son partinin numarası 3 olduğu için amaç fonksiyon değeri 3 olacaktır.

Dolayısıyla, bir GFVGP probleminin ait olduğu problem sınıfı $\langle \alpha \mid \beta \mid \gamma \mid \Theta \rangle$ dörtlüsü ile gösterilir. Örneğin $\langle sbt \mid sbt \mid \phi \mid son \rangle$ GFVGP probleminin bütün veritabanı boyutlarının ve parti kapasitelerinin eşit olduğu, veritabanları arasında zamansal kısıtların bulunmadığı ve amaç fonksiyonunun kullanılan son parti numarasını minimize etmek olduğu problem sınıfını temsil etmektedir. Önerilen problem model çatısı, veritabanları boyutları için iki model, parti kapasiteleri için iki model, veritabanları arası ilişkiler için üç model ve amaç fonksiyonu için iki model olmak üzere veritabanı göçü planlaması için gerçek hayatta karşılaşılan toplam $2 \cdot 2 \cdot 3 \cdot 2 = 24$ farklı problem sınıfını modellemektedir.

1.3 Literatür Araştırması

Veritabanı Göçü Planlaması problemi ilk olarak Patil vd. tarafından test maliyetlerini azaltma açısından ele alınmıştır [27]. Bu problemi bundan sonra Test Maliyeti Odaklı Veritabanı Göçü Planlaması (TMODVGP) problemi olarak adlandıracağız. Patil vd. TMODVGP problemini 6 farklı problem modeli altında ele almış ve bunların hesaplama karmaşıklıkları tespit etmişler ve küçük boyutlu örneklerde kullanılabilen bir tam sayılı program vermişlerdir. Daha sonra, bu problem Subramani vd. tarafından daha da genellenerek 12 model altında incelenmiş ve bütün modellerin hesaplama

karmaşıklıkları tespit edilmesinin yanı sıra, birçok parametre için problemin sabit-parametre izlenemez olduğu gösterilmiş ve problemin özel bir durumu için 2-faktörlü bir yakınsama algoritması verilmiştir [24]. Açıkalın vd. başka bir çalışmada [25], problemin en basit versiyonunun bile $2^{o(n)}$'den daha efektif bir algoritma ile çözülemeyeceğini, Üssel Zaman Hipotezi [28] altında ispatlanmıştır. Bunlara ek olarak TMOVGP problemi modelleri ile hiperçizge bölümleme problemleri arasındaki ilişkiler kurulmuş, ve en basit TMOVGP probleminin **APX-zor** olması halinde, ünlü Ağırlıklı İkiye Bölümleme probleminin de **APX-zor** olacağı gösterilmiştir. Ayrıca Hiperçizge Bölümleme problemi için geliştirilen sezgisel algoritmaların, TMOVGP problem modelleri için nasıl uyarlanacağı anlatılmış ve en başarılı Hiperçizge Bölümleme sezgisel algoritmalarının uyarlamalarının performansları küçük boyutlu örneklerde ele alınmıştır.

Öte yandan, Veritabanı Göçü Planlaması problemi, güvenlik açısından ilk kez bu çalışmada ele alınmıştır. GFVGP problem modelleri tanımladığımız şekliyle literatürdeki Çizelgeleme ve Kutulama problemlerine benzemektedir. Özellikle veritabanı boyutlarının keyfi olduğu modeller altındaki GFVGP problemi kutulama problemi ve çeşitli varyantlarına benzemektedir.

Kutulama Problemi: Verilen 0 ile 1 aralığında ağırlıklara sahip eşyaları boyutu 1 olan kutulara en az kutu kullanılacak şekilde yerleştirin. Kutulama problemi gerçek hayatta birçok uygulamaya sahip olması ve ilginç kombinatoryal yapısı nedeniyle, literatürde en çok çalışmış optimizasyon problemlerinin başında gelmektedir. Yıllar içerisinde problemin birçok farklı varyantı ortaya konmuş, hem teorik açıdan hem de gerçekte hayattaki örnekleri çözen sezgisel algoritmaların geliştirilmesi açısından ele alınmıştır. Kutulama problemi için yapılan çalışmaların hepsinin burada verilmesi mümkün değildir. Dolayısıyla, okuyuculara, Kutulama problemine yönelik yakınsama algoritmalarına ilişkin çalışmalar için [29, 30], matematiksel modeller, alt sınırlar ve kesin algoritmalara yönelik çalışmalar için [31, 32, 33] yayınlarına ve oradaki referanslara bakmalarını öneriyoruz.

Fark edileceği üzere, $\langle kfy \mid sbt \mid \phi \mid * \rangle$ notasyonuyla ifade edilen 2 model için GFVGP problemi, Kutulama problemine denktir. Bu modellerde hiç zamansal kısıt bulunmamakta ve parti boyutları birbirlerine eşit oldukları için iki amaç fonksiyonu için de optimal çözümlerin amaç fonksiyon değerleri birbirine eşittir. Ayrıca, buradaki veritabanlarımız eşyalara, partiler ise kutulara denk gelecek şekilde Kutulama problemi olarak ele alınabilir. Kutulama problemi ve GFGVP problemi arasındaki ilişkiler bir sonraki bölümde indirgemeler ile matematiksel olarak gösterilecektir. Bu bölümün devamında ele aldığımız farklı modeller ile Kutulama probleminin farklı varyantları arasındaki ilişkiler resmi olmayan bir şekilde belirtilecektir.

$\langle kyf \mid kyf \mid \phi \mid kul \rangle$ notasyonu ile ifade edilen GFVGP problemi, $\langle kyf \mid sbt \mid \phi \mid kul \rangle$ notasyonun ifade edilen GFVGP problemini, partilerin farklı kapasitelere sahip olabilmesi yönünden genellemektedir. Bu GFVGP problemi, Friesen [34] tarafından tanımlanan Değişken-Boyutlu Kutulama (DBK) problemine benzemektedir. DBK probleminde farklı boyutlara sahip eşyalar, farklı kapasite ve maliyete sahip kutu türleri verilmekte ve eşyaların en az maliyete sahip olacak şekilde kutulara yerleştirilmesi istenmektedir. Bu GFVGP problemi ve DBK problemleri arasında üç büyük fark bulunmaktadır. Bunlardan ilki, DBK probleminde her kutu türünden sonsuz tane

olduğu var sayılmaktadır, fakat, GFVGP probleminde parti sayısı veritabanı sayına eşittir. İkinci fark, DBK probleminde kutuların boyutları arttıkça maliyetlerinin arttığı varsayılmaktadır. GFVGP probleminde tüm partilerin maliyetleri eşittir. Son fark, GFVGP probleminde bazı veritabanlarının boyutları, bazı partilerin kapasitelerinden büyük olabilirken, DBK probleminde, bütün kutu türlerinin kapasiteleri en büyük eşyanın boyutundan büyüktür.

Literatürde DBK problemi için yakınsama algoritması geliştiren çok sayıda çalışma mevcuttur. DBK probleminin, kutu maliyetlerinin kutu boyutlarına eşit olduğu özel durumu için, Friesen vd. Sonraki Sığan (SS) algoritmasının sadece en büyük türden kutuları kullanan versiyonunun DBK problemi için 2-faktör bir yakınsama algoritması olduğunu göstermişlerdir [34]. Daha sonra, İlk Sığan Azalan (İSA) algoritmasının sadece en büyük türden kutuları kullanan versiyonuyla elde edilen sonuçtaki her kutunun sığdığı en küçük boyutlu kutuyla değiştirilmesiyle elde edilen sonucun maliyetinin en kötü durumda optimal maliyetin bir buçuk katından bir fazla olduğu gösterilmiştir. Hatta, İSA algoritmasının büyük eşyalar içeren kutuları dinamik olarak tekrar kutulayan bir versiyonunun bulduğu sonuçların maliyetinin en kötü durumda, OPT optimal değer olmak üzere, $(4/3)OPT + 3$ olduğu gösterilmiştir. Murgolo [35] aynı problem için asimptotik tamamen polinom zamanlı yakınsama şeması (ATPZYŞ) geliştirmiştir. Epstein ve Levin, DBK probleminin kutu maliyetlerinin kutu boyutlarından bağımsız olduğu daha genel bir durumu için ATPZYŞ vermişlerdir [36].

Epstein vd. DBK problemin kutu maliyetlerinin eşit olduğu fakat farklı boyuttaki kutuların birer birer geldiği çevrimiçi bir versiyonunu ele almışlardır [37]. Bu problem için rekabet oranı, eğer kutuların boyutları artmayan sırada geliyorsa $\frac{5}{3}$ ve $\frac{9}{5}$ aralığında, azalmayan sırada geliyorsa en az $1,734$ olan açgözlü bir algoritma vermişlerdir. Fark edileceği üzere, bu probleminin çevrimdışı versiyonu $\langle kyf \mid kyf \mid \phi \mid kul \rangle$ notasyonu ile ifade edilen modeller için GFVGP problemine oldukça benzemektedir. Aradaki tek fark, bu problemde bütün kutuların kapasitelerinin bütün eşyaların boyutlarından büyük olmasıdır.

Kutulama probleminin başka bir varyantı ise Çelişkilerle Kutulama (ÇK) problemidir [38]. ÇK probleminde girdi Kutulama problemindekine ek olarak bir çelişki çizgesi $G = (V,E)$ içermektedir. Bu çizgenin düğümleri eşyalara karşılık gelmekte ve arasında bir kenar bulunan iki eşya aynı kutuya yerleştirilememektedir. ÇK problemi, Çizge Boyama [39] problemini kapsadığı için, $\mathbf{P} \neq \mathbf{NP}$ varsayımı altında, Çizge Boyama problemi (ve dolayısıyla da ÇK problemi) hiçbir $\varepsilon > 0$ değeri için $n^{1-\varepsilon}$'dan daha iyi yakınsanamamaktadır [40]. Bu yüzden, ÇK problemi için yürütülen çalışmaların büyük bir kısmı, ÇK probleminin verilen çelişki çizgesinin Çizge Boyama probleminin efektif bir şekilde çözülebildiği çizge sınıflarına ait olduğu özel durumları için yakınsama algoritmaları geliştirmeye odaklanmıştır.

Jansen [38] ÇK probleminin çelişki çizgesinin, d bir sabit olmak üzere, d-endükleyici çizge olduğu özel durumları için ATPZYŞ vermiştir. Epstein ve Levin [41] ÇK probleminin hem eşyaların teker teker verildiği çevrimiçi versiyonunu hem de orijinal çevrim dışı versiyonunu çalışmışlardır. Orijinal versiyonda çelişki çizgesinin mükemmel çizge ve iki parçalı çizge olduğu özel durumları için, sırasıyla, $\frac{5}{2}$ ve $\frac{7}{4}$ oranlarına sahip yakınsama algoritmaları geliştirmişlerdir. Bunlara ek olarak, ÇK probleminin çevrimiçi versiyonu için, hiçbir çevrimiçi algoritmanın $\frac{155}{36} \approx$

4.30556 değerinden daha iyi bir rekabet oranına sahip olamayacaklarını göstermiş ve 4.7-rekabet oranına sahip bir algoritma geliştirmişlerdir.

Yakın zamanda DBK probleminin başka bir genellemesi ise her i eşyasının atanabileceği bir zaman aralığı $t_i = [e_i, l_i]$'nın olduğu Zaman Aralıklarıyla Değişen-Boyutlu Kutulama (ZADBK) problemidir [42]. ZADBK probleminde, iki eşyanın aynı kutuya atanabilmesi için bu eşyaların en az bir ortak zaman aralığına sahip olması gerekmektedir. Fark edileceği üzere, ZADBK ve ÇK problemleri, eşyaların hangi eşyalarla atanabileceği/atanamayacağı açısından birbirlerine benzemektedir. Aynı zamanda bu problemler $\langle kyf \mid kyf \mid ksn \mid kul \rangle$ notasyonuyla ifade edilen model altındaki GFVGP problemine de benzemektedirler. Fakat, Kutulama problemleri ile GFVGP problemi varyantları arasındaki fark, Kutulama problemindeki kısıtlamalar hangi eşyaların beraber atanamayacakları üzerineyken, GFVP problemindeki kesin kısıtlar veritabanlarının (eşyaların) hangi partilere (kutulara) atanamayacakları üzerinedir.

Yine, GFVGP problemine benzeyen bir başka Kutulama problemi varyantı ise Önceliklerle Kutulama (ÖK) problemidir. ÖK probleminde, girdi eşya boyutları ve kutu kapasitesine ek olarak bir öncelik çizgesi $G = (V, E)$ içermektedir. Bu çizgenin düğümleri eşyalara karşılık gelmekte ve bir $(i, j) \in E$ kenarı, i eşyasının j eşyasından daha önceki bir kutuya yerleştirilmesi gerektiğini belirtmektedir. Basit montaj hattı dengeleme (BMHD) problemi, ÖK problemine çok benzemektedir ve girdi eşya boyutları ve kutu kapasitesine ek olarak bir öncelik çizgesi $G = (V, E)$ içermektedir. Bu çizgenin düğümleri eşyalara karşılık gelmekte ve bir $(i, j) \in E$ kenarı, i eşyasının j eşyasından daha sonraki bir kutuya yerleştirilemeyeceğini belirtmektedir [43]. Wee ve Magazine Kutulama problemi için geliştirilmiş olan İSA ve En İyi Sığan Azalan algoritmalarını da barındıran birçok yakınsama algoritmasının [43] BMHD problemi içinde 2-faktör yakınsama algoritması olduğunu gösterdi. ÖK problemindeki öncelik ilişkileri sıkıyken, BMHD problemindeki öncelik ilişkileri gevşektir. Bu iki problem, göreceli zamansal kısıtların bulunduğu GFVGP problemlerine benzemektedir. Fakat GFVGP probleminde, hem sıkı hem gevşek öncelik ilişkileri desteklenirken aynı zamanda kesin zamansal kısıtlar da desteklenmektedir.

1.4 Katkılarımız

Tezin literatüre katkıları aşağıdaki gibidir.

1. Veritabanı Göçü Planlaması probleminin güvenlik açısından ele alan bir problem çatısının tanımlanması. Bu problem çatısı, Veritabanı Göçü Planlaması problemini test maliyetlerinin azaltılması açısından modelleyen problem çatısına [24] benzemekle birlikte, farklı amaç fonksiyonlarını modellemesi ve zamansal kısıtları desteklemesi açısından farklılık göstermektedir.

2. Problem çatısı altında tanımlanan 24 GFVGP problem modelinden 23 tanesinin hesaplama karmaşıklığının belirlenmesidir.

3. **NP-zor** olduğu tespit edilen modeller için makul sürelerde optimale yakın çözümler üreten sezgisel ve metasezgisel algoritmaların geliştirilmesi ve başarımlarının ölçülmesidir.

Tezin geri kalanı şu şekilde organize edilmiştir. Bölüm 2'de GFVGP probleminin hesaplama karmaşıkları belirlenmektedir. Bölüm 3'te GFVGP modelleri için geliştirilen sezgisel yöntemler anlatılmaktadır. Geliştirilen sezgisel algoritmaların başarımları Bölüm 4'te ölçülmekte ve tartışılmaktadır. Bölüm 5'te sonuçlar tartışılmış ve diğer araştırmacılar için ilginç olabilecek çalışma konuları belirtilmiştir.

2. HESAPLAMA KARMAŞIKLIĞI SONUÇLARI

Tezin bu bölümü, farklı modeller için GFVGP probleminin hesaplama karmaşıklarının belirlenmesine adanmıştır. Elde eden sonuçların özeti Çizelge 2.1'de görülebilir. Çizelge, ilgili notasyonun karşılık geldiği model altındaki GFVGP probleminin hesaplama karmaşıklığını göstermektedir.

Çizelge 2.1: Farklı modeller altındaki GFVGP probleminin hesaplama karmaşıkları.

Problem	$\alpha = sbt$	$\alpha = kyf$
$\langle \alpha \mid sbt \mid yok \mid kul \rangle$	$O(n)$	NP-zor
$\langle \alpha \mid sbt \mid yok \mid son \rangle$	$O(n)$	NP-zor
$\langle \alpha \mid kyf \mid yok \mid kul \rangle$	$O(n)$	NP-zor
$\langle \alpha \mid kyf \mid yok \mid son \rangle$	$O(n)$	NP-zor
$\langle \alpha \mid sbt \mid ksn \mid kul \rangle$	$O(n+m)$	NP-zor
$\langle \alpha \mid sbt \mid ksn \mid son \rangle$	$O(m+nlgn)$	NP-zor
$\langle \alpha \mid kyf \mid ksn \mid kul \rangle$	?	NP-zor
$\langle \alpha \mid kyf \mid ksn \mid son \rangle$	$O(m+nlgn)$	NP-zor
$\langle \alpha \mid sbt \mid grc \mid kul \rangle$	NP-zor	NP-zor
$\langle \alpha \mid sbt \mid grc \mid son \rangle$	NP-zor	NP-zor
$\langle \alpha \mid kyf \mid grc \mid kul \rangle$	NP-zor	NP-zor
$\langle \alpha \mid kyf \mid grc \mid son \rangle$	NP-zor	NP-zor

İlk olarak, Teorem 1 ile keyfi veritabanı boyutlu 12 GFVGP probleminin **NP-zor** olduğunu ispatlıyoruz.

Teorem 1. $\langle kyf \mid * \mid * \mid * \rangle$ *notasyonu ile ifade edilen* 12 *model için GFVGP problemi* **NP-zor***'dur.*

İspat. Fark edileceği üzere $\langle kyf \mid sbt \mid yok \mid son \rangle$ ve $\langle kyf \mid sbt \mid yok \mid kul \rangle$ notasyonları ile ifade edilen modeller, sırasıyla $\langle kyf \mid * \mid * \mid son \rangle$ ve $\langle kyf \mid * \mid * \mid kul \rangle$ notasyonlarıyla ifade edilen modeller içinde en kısıtlı modellerdir. Ayrıca, zamansal kısıtların olmadığı ve parti kapasitelerinin eşit olduğu modellerde, *son* ve *kul* amaç fonksiyonları aynı değere sahip olacaktır. Bu yüzden $\langle kyf \mid sbt \mid yok \mid son \rangle$ ve $\langle kyf \mid sbt \mid yok \mid kul \rangle$ notasyonları ile ifade edilen iki problem birbirlerine denktir. Dolayısıyla, Kutulama probleminin optimizasyon versiyonundan $\langle kyf \mid sbt \mid yok \mid kul \rangle$ notasyonu ile ifade edilen model altındaki GFVGP problemine bir sıkı indirgeme [44] yaparsak ispatımız tamamlanacaktır.

Kutulama probleminin optimizasyon halinde, bize bir kutu kapasitesi V ve kutulanacak eşyaların boyutlarını içeren bir liste $A = [a_1, \ldots, a_n]$ verilir ve amaç bütün eşyaların, kutu kapasitelerini aşmadan, en az sayıda kutu kullanılacak şekilde kutulara yerleştirilmesidir.

Bir Kutulama problemi örneği I verildiğinde, I örneğine karşılık gelen bir $\langle kyf \mid sbt \mid yok \mid kul \rangle$ notasyonu ile ifade edilen model altındaki GFVGP problemi örneği F'yi şu şekilde oluşturuyoruz:

- F örneğindeki veritabanlarının boyutları, I örneğindeki eşyaların boyutlarına eşittir.
- F örneğindeki her bir partinin kapasitesi I örneğindeki kutu kapasitesine eşittir.

Yukarıda yapılan indirgeme, I'daki her olurlu çözüm aynı amaç fonksiyonu değeriyle F'deki bir olurlu çözüme eşleştirildiğinden, bir sıkı indirgemedir.

Sonuç 1 direkt olarak Teorem 1'in ispatında kullanılan indirgemenin bir yakınsama koruyan sıkı indirgeme olması ve Kutulama probleminin bir ATPZYS olması fakat PZYS olmadığından izlemektedir.

Sonuç 1. *Eğer* $\mathbf{P} \neq \mathbf{NP}$*,* $\langle kyf \mid * \mid * \mid * \rangle$ *notasyonu ile ifade edilen* 12 *hiçbir model altında GFVGP problemi için TPZYS yoktur. Dahası,* $\langle kyf \mid sbt \mid yok \mid * \rangle$ *notasyonuyla ifade edilen* 2 *model altında GFVGP problemi için ATPZYS vardır.*

Teorem 1'den dolayı, bölümün geri kalanında veritabanı boyutlarının birbirlerine eşit olduğu modellere odaklanıyoruz. Teorem 2 ile veritabanı boyutlarının sabit olduğu ve zamansal kısıtların bulunmadığı 4 model için GFVGP probleminin $O(n)$-zamanda optimal çözülebileceğini gösteriyoruz.

Teorem 2. $\langle sbt \mid * \mid yok \mid * \rangle$ *notasyonu ile ifade edilen* 4 *model altında GFVGP problemi* $O(n)$*-zamanda optimal çözülebilir.*

İspat. Bu teoremi, $O(n)$-zamanda çalışan Algoritma 1'i vererek ispatlıyoruz.

Bu notasyon ile gösterilen modellerde zamansal kısıtlar bulunmadığı için, kısıt listesi C'yi Algoritma 1'in girdilerinden çıkarıyoruz. Bu 4 problem modelinde bütün veritabanlarının boyutlarının aynı olduğunu ve bir w sayısına eşit olduğunu hatırlayın. Eğer amaç fonksiyonumuz *kul* ise, partileri kapasitelerine göre artmayan sırada olacak şekilde yeniden numaralandırıyoruz. Bu işlem Sayma Sıralaması algoritması ile $O(n)$-zamanda gerçekleştirilebilir [45]. Eğer amaç fonksiyonu *son* ise, partileri oldukları şekilde bırakıyoruz. Algoritma 1 her partiye, o partiye atanabilecek en fazla sayıda veritabanını atayan bir açgözlü algoritmadır. Yani, ilk partiye $\lfloor \frac{l_1}{W} \rfloor$ veritabanı, ikinci partiye $\lfloor \frac{l_2}{W} \rfloor$ veritabanı atayıp, bütün veritabanları bir partiye atanana kadar bu şekilde devam etmektedir. Algoritma 1'in $O(n)$-zamanda çalıştığı aşikardır.

Doğruluk: İlk olarak amaç fonksiyonunun *kul* olduğunu varsayalım. s, Algoritma 1 tarafından bulunan planda kullanılan parti sayısını göstersin. Çelişki yakalamak için optimal planda $k < s$ parti kullanıldığını varsayalım. Ama bu sefer de, sadece

Algorithm 1 GFVGP probleminin $\langle sbt \mid * \mid yok \mid * \rangle$ notasyonu ile ifade edilen modeli için polinom zamanlı algoritma.

1: W veritabanlarının boyutunu göstersin.
2: U atanmamış veritabanlarının kümesini göstersin ve başta $\mathscr{B}$'ye eşit olsun
3: **if** Amaç fonksiyonu *kul* ise **then**
4: Partileri kapasiteleri artmayacak şekilde tekrar numaralandır.
5: **end if**
6: $i = 0$.
7: **while** $(U \neq \emptyset)$ **do**
8: $i = i + 1$.
9: D, U'nun $\lfloor \frac{l_i}{W} \rfloor$ elemanlı bir alt kümesi olsun .
10: D'de bulunan veritabanlarını i. partiye ata.
11: $U = U - D$.
12: **end while**

ilk k partiyi kullanan bir optimal plan olmalıdır çünkü partiler kapasitelerine göre azalmayan sıradadırlar. Fakat, hiçbir planda bütün veritabanları ilk k partiye atanamaz, aksi taktirde Algoritma 1 de bunu yapardı. Dolayısıyla, Algoritma 1 tarafından bulunan plan optimaldir.

Şimdi, amaç fonksiyonunun *son* olduğunu varsayalım. s, Algoritma 1 tarafından bulunan planda kullanılan en son partinin numarasını göstersin. Çelişki yakalamak için optimal planda kullanılan son partinin numarası k'nın s'den küçük olduğu, yani $k < s$ olduğunu varsayalım. Fakat, hiçbir planda bütün veritabanları ilk k partiye atanamaz, aksi taktirde Algoritma 1 de bunu yapardı. Dolayısıyla, Algoritma 1 tarafından bulunan plan optimaldir.

2.1 Zamansal Kısıtlı Modeller

Bu bölümde, sabit veritabanı boyutlu ve zamansal kısıtların olduğu 8 problemi ele alıyoruz ve $\langle sbt \mid kyf \mid ksn \mid kul \rangle$ notasyonuyla ifade edilen model dışındaki 7 modelin hesaplama karmaşıklarını tespit ediyoruz. İlk olarak kesin zamansal kısıtların olduğu modellerden amaç fonksiyonunun *son* olduğu 2 modelin, $O(m + n \lg n)$-zamanda çalışan bir algoritma ile optimal çözülebileceğini gösteriyoruz. Daha sonra, $\langle sbt \mid sbt \mid ksn \mid kul \rangle$ notasyonuyla ifade edilen modelin $O(n)$-zamanda optimal çözülebildiğini ispatlıyoruz. Son olarak göreceli zamansal kısıtların olduğu 4 GFVGP problem modelinin **NP-zor** olduğunu gösteriyoruz. Başlamadan önce aşağıdaki tanımları yapıyoruz.

Tanım 1. *r_i'yi, veritabanı B_i'nin* **sürülme zamanı** *olarak tanımlıyoruz. Eğer hiç $s_i \geq c$ şeklinde kesin kısıt bulunmuyorsa, r_i, 1 değerini alır. Aksi taktirde, r_i, k değerini alır, burada k, $s_i \geq c$ şeklindeki kısıtlar içerisinde en yüksek sabite sahip kısıtın sabitidir.*

Tanım 2. *d_i'yi, veritabanı B_i'nin* **son teslim zamanı** *olarak tanımlıyoruz. Eğer hiç $s_i \leq c$ şeklinde kesin kısıt bulunmuyorsa, d_i, n değerini alır. Aksi taktirde, d_i, k değerini alır, burada k, $s_i \leq c$ şeklindeki kısıtlar içerisinde en küçük sabite sahip kısıtın sabitidir.*

Fark edileceği üzere, bir veritabanının sürülme zamanı, o veritabanının kesin zamansal kısıtlar ihlal edilmeden atanabileceği ilk partinin numarasını göstermektedir.

Benzer bir şekilde, bir veritabanının son teslim zamanı, o veritabanının kesin zamansal kısıtlar ihlal edilmeden atanabileceği en son partinin numarasını göstermektedir.

Teorem 3, *son* amaç fonksiyonlu, veritabanı boyutlarının sabit olduğu ve sadece kesin zamansal kısıtların bulunduğu 2 model için GFVGP probleminin $O(m+n\cdot \lg n)$-zamanda çözülebildiğini göstermektedir. Burada, m zamansal kısıt sayısı ve n veritabanı sayısıdır.

Teorem 3. *$\langle sbt \mid * \mid ksn \mid son\rangle$ notasyonuyla ifade edilen 2 model için GFVGP problemi $O(m+n\cdot \lg n)$-zamanda optimal çözülebilir.*

İspat. Teoremi bir algoritma vererek ispatlıyoruz.

Algorithm 2 $\langle sbt \mid * \mid ksn \mid son\rangle$ modelleri için polinom zamanlı algoritma.

1: W veritabanlarının boyutunu göstersin.
2: Her veritabanı B_i için, r_i ve d_i değerlerini **C** üzerinden geçerek hesapla.
3: Veritabanlarını sürülme zamanlarına göre küçükten büyüğe sırala.
4: H anahtarları son teslim zamanları olan boş bir veritabanı yığın ağacı olarak tanımla.
5: **for** j 1'den n'ye kadar **do**
6: Eğer, $r_i = j$ ise, veritabanı B_i'yi H'ye ekle.
7: H'den $\min\{\lfloor \frac{l_i}{W} \rfloor, |H|\}$ tane veritabanını çıkart ve parti j'ye ata.
8: **end for**

Çalışma Zamanı: Algoritma ilk olarak C listesi üzerinden bir kez geçerek bütün veritabanlarının sürülme ve son teslim zamanlarını hesaplamaktadır. Bu işlem $O(n+m)$ zamanda gerçekleştirilir. Algoritmanın çalışması esnasında, her veritabanı yığın ağacına sadece bir kere eklenir ve bir kere çıkarılır. Bu işlemler toplamda $O(n\lg n)$ vakitte gerçekleştirilir. Dolayısıyla algoritmanın çalışma zamanı $O(m+n\lg n)$'dir.

Doğruluk: O, verilen bir GFVGP problemi örneğinin optimal çözümü olsun. A, aynı örnek için Algoritma 2'nin bulduğu çözüm olsun. $O_1, O_2, \ldots, O_n$ ve $A_1, A_2, \ldots, A_n$, sırasıyla O ve A çözümlerinde partilere atanan veritabanları kümelerini göstersin. Algoritmanın doğru olduğunu O çözümünü kötüleştirilmeden A çözümüne dönüştürerek ispatlayacağız.

k, A_k ve O_k kümelerinin eşit olmadığı en küçük tam sayı olsun. Genelliği kaybetmeden böyle bir k tamsayısının var olduğunu varsayalım. Aksi taktirde çözümler aynı olacak ve ispatımız tamamlanmış olacaktır.

Fark edileceği üzere, tüm $k' < k$ sayıları için $A_{k'} = O_{k'}$ eşitliği sağlandığından ve algoritmamız parti k'ye en fazla sayıda veritabanını atadığından $|A_k| \geq |O_k|$ eşitsizliği sağlanmaktadır. Eğer $|A_k| > |O_k|$ eşitsizliği sağlanıyorsa, $A_k \setminus O_k$ kümesinde bulunan $|A_k| - |O_k|$ tane veritabanını O çözümünde parti k'ye taşıyalım. Bu işlemden sonra bile A_k kümesi O_k kümesine eşit olmayabilir. Bu durumda, $A_k \setminus O_k \neq \emptyset$ ve $O_k \setminus A_k \neq \emptyset$ eşitsizliklerini sağlanacaktır.

B_i, $A_k \setminus O_k$ kümesinden bir veritabanı olsun. B_j ise $O_k \setminus A_k$ kümesinden bir veritabanı olsun. Fark edileceği üzere, B_i veritabanının son teslim zamanı, B_j veritabanının son teslim zamanından küçük ya da eşittir. Aksi taktirde algoritma, B_i yerine B_j'yi atardı. Yukarıdaki şartlar sağlandığı için, O çözümünde B_i ve B_j veritabanlarının yerleri, O çözümü kötüleştirilmeden karşılıklı değiştirilebilir. Bu argümanı tümevarımla tekrar ederek, O çözümünü kötüleştirmeden A çözümüne dönüştürebiliriz.

Teorem 4 $\langle sbt \mid sbt \mid ksn \mid kul \rangle$ notasyonuyla ifade edilen GFVGP probleminin $O(n+m)$-zamanda çözülebileceğini göstermektedir.

Teorem 4. *$\langle sbt \mid sbt \mid ksn \mid kul \rangle$ notasyonuyla ifade edilen model için GFVGP problemi $O(n+m)$-zamanda optimal çözülebilir.*

İspat. Teorem 4'ü, problemimizi *Minimum Aktif İşlemci Zamanı* probleminin $O(n)$-zamanda çözülebilen özel bir durumuna indirgeyerek ispatlıyoruz. Minimum Aktif İşlemci Zamanı (MAİZ) probleminde, bize n tane iş verilmekte, her i işi l_i uzunluktadır ve boş olmayan $T_i = \{I_k^i = [r_k^i, d_k^i]\}_k^{m_i}$ zaman aralıklarından birinde çizelgelenebilir. Bir zaman aralığı, eğer o zaman aralığında bir iş çizelgelendiyse aktif sayılır. Her zaman aralığında işlenebilecek en fazla iş sayısı parametresi B verildiğinde, amacımız bütün işleri en az sayıda aktif zaman aralığı yaratacak şekilde çizelgelemektir.

Chang vd. MAİZ probleminin bütün işlerin birim uzunlukta olduğu, her işin tek bir zaman aralığında çizelgelenebildiği ve işlerin çizelgelenebilecekleri son teslim zamanına göre küçükten büyüğe sırada verildiği özel durumunu $O(n)$-zamanda çözen bir algoritma verdiler [46]. Fark edileceği üzere, MAİZ problemi ile ele aldığımız GFVGP problemi aslında aynı problemdir fakat MAİZ probleminde veritabanlarına iş, partilere ise zaman aralığı denmektedir ve bizim problemimizin girdilerinde veritabanları son teslim zamanına göre sıralı olmak zorunda değildir. Dolayısıyla, ispatın kalanında bizim problemimizden MAİZ problemine bir sıkı indirgeme vereceğiz.

$\langle sbt \mid sbt \mid ksn \mid kul \rangle$ notasyonuyla ifade edilen GFVGP problemi örneği I verildiğinde, I örneğine karşılık gelen, yukarıda anlatılan şartları sağlayan MAİZ problemi örneği F'yi şu şekilde oluşturuyoruz:

İlk olarak bütün veritabanlarının sürülme ve son teslim zamanlarını, kısıt listesi C'nin üzerinden bir kez geçerek $O(m+n)$-zamanda belirliyoruz. Daha sonra, veritabanlarını,

son teslim zamanlarına göre artmayan sırada olacak şekilde sıralıyoruz. Bu işlem Sayma Sıralaması algoritması kullanılarak $O(n)$-zamanda gerçekleştirilebilir [45]. Bu adımlardan sonra, I'ya karşılık gelen F örneğini şu şekilde oluşturuyoruz:

- F örneğindeki işler, I örneğindeki veritabanlarına karşılık gelir.
- F örneğindeki iş i'nin atanma aralığı $T_i = \{[r_i, d_i]\}$'dır ve boyutu 1'dir.
- F örneğindeki her zaman aralığında işlenebilecek en fazla iş sayısı parametresi B, I örneğindeki parti kapasitesine eşittir.

Yukarıda yapılan indirgeme, I örneğindeki her olurlu çözümü aynı amaç fonksiyonu değeriyle F örneğindeki bir olurlu çözüme eşleştirdiğinden, bir sıkı indirgemedir. Ayrıca, bütün indirgeme $O(n+m)$ zaman almakta ve oluşturulan F örneğinde Chang vd. gösterdiği üzere $O(n)$-zamanda çözülebildiği için, $\langle sbt \mid sbt \mid ksn \mid kul \rangle$ notasyonuyla ifade edilen GFVGP problemi $O(n+m)$-zamanda çözülebilmektedir.

Son olarak, Teorem 5 ile göreceli zamansal kısıtlara sahip kalan 4 GFVGP probleminin **NP-zor** olduğunu, klasik 3-Bölümleme [47] probleminden bir indirgeme ile gösteriyoruz.

Teorem 5. $\langle sbt \mid * \mid grc \mid * \rangle$ *notasyonuyla ifade edilen* 4 *model için GFVGP problemi* **NP-zor***'dur.*

İspat. İlk olarak, $\langle sbt \mid sbt \mid grc \mid kul \rangle$ notasyonu ile ifade edilen model altında GFVGP probleminin **NP-zor** olduğunu 3-Bölümleme probleminden eşleme indirgemesiyle (mapping reduction) göstereceğiz.

3-Bölümleme problemi: Toplamları bir D tam sayısı için tD olan $\frac{D}{4} < x_i < \frac{D}{2}$ eşitsizliğini sağlayan $3t$ tane tamsayıdan oluşan bir çoklu küme $S = \{x_1, x_2, \ldots, x_{3t}\}$, her biri 3 elemanlı ve toplamları D olan t gruba bölünebilir mi?

Fark edileceği üzere, 3-Bölümleme problemi bir optimizasyon problemi değil, cevabı "EVET" ya da "HAYIR" olan bir karar problemidir.

Bize bir 3-Bölümleme örneği verildiğinde, buna karşılık gelen $\langle sbt \mid sbt \mid grc \mid kul \rangle$ notasyonu ile ifade edilen model altındaki GFVGP problemi örneğini şu şekilde oluşturuyoruz:

- GFVGP örneği tD tane birim boyutlu veritabanı içerir. Yani, 3-Bölümleme örneğinin her $x_i \in S$ tamsayısı için, GFVGP örneği x_i tane veritabanı içerir. Veritabanları $B_1, B_2, \cdots, B_{x_1}$, x_1 tamsayısına karşılık gelir, veritabanları $B_{x_1+1}, \cdots, B_{x_1+x_2}$, x_2 tamsayısına karşılık gelir, ve bu şekilde devam eder.

- Parti kapasiteleri sabit ve her partinin kapasitesi D'dir. Bütün veritabanlarının boyutu 1 olduğu için, her parti D veritabanını taşıyabilir.

- 3-Bölümleme örneğinin her $x_i \in S$ tamsayısı için, GFVGP örneği, x_i'ye karşılık gelen bütün veritabanlarının aynı partiye atanmasını garanti etmek için x_i tane gevşek göreceli kısıt içermektedir. Tam olarak GFVGP örneği, x_1 için $s_1 \leq s_2 \leq s_3 \leq \cdots \leq s_{x_1-1} \leq s_{x_1} \leq s_1$ kısıtlarını içerir ve x_i, $i > 1$ için $s_{\left(\sum_{j=1}^{i-1} x_j\right)+1} \leq s_{\left(\sum_{j=1}^{i-1} x_j\right)+2} \leq \cdots \leq s_{\sum_{j=1}^{i} x_j} \leq s_{\left(\sum_{j=1}^{i-1} x_j\right)+1}$ kısıtlarını içerir. Bunlardan başka zamansal kısıt içermez.

İspatı tamamlamamız için gereken son şey, eğer 3-Bölümleme örneğinin cevabı "EVET" ise, oluşturulan GFVGP örneğinin optimal değerinin t olduğunu göstermek, aksi taktirde ise daha yüksek olduğunu göstermek.

I, 3-Bölümleme probleminin bir örneği olsun. I'nın cevabının "EVET" olduğunu varsayalım. O halde, S'deki elemanlar 3 elemanlı öyle t gruba bölümlenebiliyorlar ki her grubun toplamı tam olarak D ediyor. $1 \leq i \leq t$ koşulunu sağlayan herhangi bir i için, i. gruptaki elemanlara karşılık gelen bütün veritabanları, i. partide göç ettirilebilir. Bu yüzden, oluşturulan GFVGP örneğinin optimal değeri en fazla t olacaktır. Fakat, optimal değer her partinin kapasitesinin D olduğu ve tD tane birim boyutlu veritabanı olduğundan genelleştirilmiş güvercin yuvası ilkesine göre t'den az olamaz. Dolayısıyla optimal değer (göstermemiz gerektiği gibi) t'dir.

Şimdiyse, I'nın cevabının "HAYIR" olduğunu varsayalım. Çelişki yakalamak için, GFVGP örneğinin optimal çözümünde en fazla t parti kullanıldığını varsayalım. GFVGP örneğinin optimal çözümünün genelleştirilmiş güvercin yuvası ilkesine göre t'den az olamayacağını da bildiğimize göre, optimal değer t olacaktır. O halde, her partiye tam olarak D veritabanı atanmış olmalı. $1 \leq i \leq t$ koşulunu sağlayan her i için, i. partiye atanan veritabanlarını ele alalım. Herhangi bir $x_j \in S$ tamsayısı için, yaratmış olduğumuz göreceli zamansal kısıtlar yüzünden ya x_j'ye karşılık gelen bütün veritabanları bu partiye atandı ya da hiçbiri bu partiye atanmadı. Bütün $x_j \in S$ tamsayıları $\frac{D}{4}$ ile $\frac{D}{2}$ aralığında olduğu için, her partide veritabanları toplamları D olan 3 tam sayıya karşılık gelmektedir. Bu her parti için geçerli olduğu için, örneğin cevabı "EVET" olmalıydı. Dolayısıyla bu bir çelişki yaratmaktadır.

Fark edileceği üzere, oluşturulan GFVGP örneğinde partilerin kapasiteleri sabit olduğu için ve göreceli zamansal kısıtlar sadece bazı veritabanı altkümelerinin aynı partide göç etmesini sağladığı için, çözümün olurluluğu bozulmadan herhangi iki partinin içerikleri birbirleriyle değiştirilebilir. Bu yüzden oluşturulan GFVGP örneğindeki herhangi bir t değeri için, t parti kullanan herhangi bir çözüm, ilk t partiyi kullanabilir. Bu sebeple, yapılan indirgeme, $\langle sbt \mid sbt \mid grc \mid son \rangle$ notasyonuyla ifade edilen problem için de geçerlidir. $\langle sbt \mid sbt \mid grc \mid kul \rangle$ ve $\langle sbt \mid sbt \mid grc \mid son \rangle$ notasyonları ile ifade

edilen modeller, $\langle sbt \mid * \mid grc \mid * \rangle$ notasyonuyla ifade edilen modeller içinde en kısıt modeller olduğu için, indirgeme bu notasyonla ifade edilen 4 problem için geçerlidir.

Eğer 3-Bölümleme problemi birlik-gösterimde verilirse, yukarıdaki indirgeme sonucunda GFVGP örneğinin boyutunun 3-Bölümleme örneğinin boyutu cinsinden polinom olduğuna dikkat edelim. Fakat 3-Bölümleme problemi **güçlü NP-zor** olduğu için bu ispat için sorun yaratmıyor.

3. SEZGİSEL ALGORİTMALAR

Bu bölümde, $\langle kyf \mid kyf \mid * \mid * \rangle$ notasyonu ile ifade edilen 6 **NP-zor** GFVGP problem modeli için küçük örneklerde optimal çözümü bulmakta kullanılan tamsayılı lineer program verilecek ve sonrasında geliştirilen sezgisel algoritmalar anlatılacaktır. İlk olarak Kutulama problemi için geliştirilmiş sezgisel algoritmaların, GFVGP problemine nasıl uyarlandıkları anlatılacak ve geliştirilen özgün kurucu sezgisel algoritmalar tanıtılacaktır. Daha sonra, Kutulama problemi için geliştirilen yerel arama algoritmalarındaki ana fikirler anlatılacak ve bu fikirlerden yararlanan ve genelleştirilen GFVGP için geliştirilen yerel arama algoritmaları anlatılacaktır. Son olarak bu algoritmaları kullanan Memetik Algoritmalar açıklanacaktır.

3.1 Tamsayılı Program

Geliştirdiğimiz sezgisel algoritmaların başarımlarını ölçmek için sezgisel algoritmalar tarafından bulunan çözümlerle aşağıda verilen tamsayılı lineer programın CPLEX ile çözdürülmesiyle elde edilen çözümleri karşılaştırıyoruz. Her örnek için CPLEX'e 8 saatlik bir zaman kısıtı ve 8GB bellek limiti verdiğimiz için her zaman optimal çözüm bulunamamaktadır. Bu durumlarda CPLEX'in elde ettiği alt sınır kullanılmaktadır.

Bu tamsayılı lineer programda, x_{ij} karar değişkeni, B_i veritabanı parti j'ye atandığı durumda 1 değerini almakta, aksi taktirde ise 0 değerini almaktadır. Karar değişkeni y_j, parti j kullanıldığı durumda 1 değerini, aksi taktirde ise 0 değerini almaktadır.

Kısıt 3.1, her veritabanının tek bir partiye atanmasını sağlamaktadır. Parti kapasiteleri Kısıt 3.2 sayesinde sağlanmaktadır. Kısıt 3.3, formülasyonu güçlendiren bir geçerli eşitsizliktir. Kısıtlar 3.4 ve 3.5 kesin zamansal kısıtlara uyulmasını sağlamaktadır. Kısıtlar 3.6 ve 3.7 ise göreceli zamansal kısıtlara uyulmasını sağlamaktadır. Kısıt 3.6'daki GE_i ifadesi, veritabanı B_i için $s_k \leq s_i$ şeklindeki gevşek zamansal kısıtlar kümesini göstermektedir. Kısıt 3.7'deki SE_i ifadesi, veritabanı B_i için $s_k < s_i$ şeklindeki sıkı zamansal kısıtlar kümesini göstermektedir. Amaç fonksiyonun *kul* olduğu modeller altındaki GFVGP problemi için, Kısıt 3.8 z karar değişkeninin kullanılan parti sayısına eşit olmasını sağlamaktadır. Amaç fonksiyonun *son* olduğu modeller altındaki GFVGP problemi için, Kısıt 3.9 z karar değişkeninin kullanılan en büyük parti numarasına eşit olmasını sağlamaktadır. Kısıtlar 3.10 ve 3.11 ise karar değişkenlerinin alabileceği değerleri kısıtlamaktadır.

Zamansal kısıtların olmadığı modeller altındaki GFVGP problemi için Kısıtlar 3.1, 3.2, 3.3, 3.10 ve 3.11 kısıtları ve amaç fonksiyonuna göre Kısıt 3.8'den veya 3.9'dan oluşan tamsayılı lineer matematiksel model kullanılmaktadır. Kesin kısıtların olduğu modeller içinse Kısıtlar 3.1, 3.2, 3.3, 3.4, 3.5, 3.10 ve 3.11 ve amaç fonksiyonuna göre Kısıt 3.8'den veya 3.9'dan oluşan tamsayılı lineer program kullanılmaktadır. Göreceli kısıtların olduğu modeller altındaki GFVGP problemi için ise 3.1, 3.2, 3.3, 3.4, 3.5,

3.6, 3.10 ve 3.11 kısıtları ve amaç fonksiyonuna göre Kısıt 3.8'den veya 3.9'dan oluşan tamsayılı lineer program kullanılmaktadır.

z'yi minimize et
öyle ki

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i = 1,...,n \tag{3.1}$$

$$\sum_{i=1}^{n} w_i \cdot x_{ij} \leq l_j \cdot y_j, \quad \forall j = 1,...,n \tag{3.2}$$

$$\sum_{i=1}^{n} x_{ij} \leq y_j, \quad \forall j = 1,...,n \tag{3.3}$$

$$\sum_{j=1}^{n} j \cdot x_{ij} \leq d_i, \quad \forall i = 1,...,n \tag{3.4}$$

$$\sum_{j=1}^{n} j \cdot x_{ij} \geq r_i, \quad \forall i = 1,...,n \tag{3.5}$$

$$\sum_{j=1}^{n} j \cdot x_{kj} \leq \sum_{j=1}^{n} j \cdot x_{ij}, \quad \forall i = 1,...,n, \quad \forall k \in GE_i \tag{3.6}$$

$$\sum_{j=1}^{n} j \cdot x_{kj} < \sum_{j=1}^{n} j \cdot x_{ij}, \quad \forall i = 1,...,n, \quad \forall k \in SE_i \tag{3.7}$$

$$\sum_{j=1}^{n} y_j = z, \tag{3.8}$$

$$j \cdot y_j \geq z, \quad \forall j = 1,...,n \tag{3.9}$$

$$x_{ij} \in \{0,1\}, \quad \forall i,j = 1,...,n \tag{3.10}$$

$$y_j \in \{0,1\}, \quad \forall j = 1,...,n \tag{3.11}$$

3.2 Kurucu Sezgisel Algoritmalar

Bu alt bölümde Kutulama problemi için geliştirilmiş İlk Sığan (İS) ve İlk Sığan Azalan (İSA) [29] algoritmalarının ve yaygın olarak çizelgeleme problemlerinde kullanılan Son Teslim Zamanı İlk (STZİ) algoritmasının [48] GFVGP problemine uyarlanmış versiyonlarının sözde kodları ile GFVGP problemi için geliştirilen İlk Sığan Partiler Karıştırılmış (İSPK) İlk Sığan Azalan Partiler Karıştırılmış (İSAPK), İlk Sığan Azalan Partiler Azalan (İSAPA), Topolojik İlk Sığan (TİS) algoritmalarının sözde kodları verilmiştir. Bütün algoritmalar aslında veritabanları ve/veya partilerin sıraladıktan sonra İS algoritmasını kullanmaktadır. Dolayısıyla yerden kazanmak adına Algoritma 3 parametrik bir şekilde bütün algoritmaları gerçekleyebilecek bir biçimde yazılmıştır. Bu kurucu algoritmalar, geliştirilen memetik algoritmalarda ve yerel arama algoritmalarında farklı şekillerde kullanılmaktadır.

Algoritma 3, girdi olarak GFVGP örneğine ek olarak U, *sirala*1 ve *sirala*2 adlı üç parametre daha almaktadır.

İlk parametre U, atanacak veritabanları kümesini göstermektedir. $U = \mathscr{B}$ olduğunda, Algoritma 3 sıfırdan bir çözüm üretmektedir. $U \subsetneq \mathscr{B}$ olduğunda ise, bir *kısmi çözüm* tamamlanır. Kısmi çözümler, bazı veritabanlarının atanmadığı çözümlerdir.

İkinci parametre *sirala*1, veritabanlarının sıralamasını kontrol etmek için kullanılır. Son parametre *sirala*2, partilerin sıralamasını kontrol etmek için kullanılır.

Algorithm 3 Kurucu sezgisel algoritmalar için sözde kod.

Girdi: $\langle w, l, C \rangle$, U, *sirala*1, *sirala*2

if *sirala*1 $==$ 4 **then**
    U'yu göreceli kısıtları ifade eden öncelik çizgesinin rastgele bir topolojik sıralamasına göre sırala.
else if *sirala*1 $==$ 3 **then**
    U'yu son teslim zamanlarına göre küçükten büyüğe sırala.
else if *sirala*1 $==$ 2 **then**
    U'yu veritabanı boyutlarına göre büyükten küçüğe sırala.
else if *sirala*1 $==$ 1 **then**
    U'yu rastgele karıştır.
end if
if *sirala*2 $==$ 2 **then**
    Partileri boyutlarına göre büyükten küçüğe sırala.
else if *sirala*2 $==$ 1 **then**
    Partileri rastgele karıştır.
end if
for all $B_i \in U$ **do**
    Zamansal kısıtlara göre, B_i'nin atanabileceği en erken parti e_i'yi belirle.
    Zamansal kısıtlara göre, B_i'nin atanabileceği en son parti f_i'yi belirle.
    if Eğer B_i e_i ve f_i arasındaki hiçbir partiye sığmıyorsa **then**
        B_i'yi r_i ve d_i arasındaki bir partiye rastgele ata.
    else
        B_i'yi sığdığı ilk sıradaki partiye ata.
    end if
end for

İS algoritması sadece veritabanlarının sırasını karıştırmaktadır. Fakat, GFVGP probleminde partiler özdeş olmadığı için, bizim partilerin sırasını karıştırmayı tercih ettiğimiz zamanlar olmaktadır. İS algoritmasının partilerin sırasını rastgele karıştıran halini İSPK kısaltmasıyla ile ifade ediyoruz.

İSA algoritması, veritabanlarının boyutlarına göre büyükten küçüğe sıralamaktadır. Biz de İSA algoritmasının iki farklı adaptasyonunu kullanıyoruz. İSAPA algoritmasında veritabanlarının yanı sıra, partiler de kapasitelerine göre büyükten küçüğe sıralanmaktadır. Ayrıca, İSA algoritmasının partileri sıralamak yerine karıştıran halini İSAPK olarak isimlendiriyoruz.

Kesin zamansal kısıtların olduğu modeller için, kesin zamansal kısıtlara öncelik vermek adına veritabanlarını son teslim zamanlarına göre sıralamayı tercih ettiğimiz zamanlar olmaktadır. Bu algoritmayı STZİ olarak isimlendiriyoruz.

Göreceli zamansal kısıtların olduğu modeller için, göreceli zamansal kısıtlardan bir öncelik çizgesi oluşturup çizgenin bir topolojik sıralamasını kullanarak veritabanlarını sıralıyoruz. Oluşturulan öncelik çizgesinin düğümleri veritabanlarını karşılık gelmektedir. Bu çizgedeki her B_i, B_j kenarı, B_i veritabanının B_j veritabanından

daha geç atanmaması gerektiğini ifade etmektedir. Bu algoritmayı TİS olarak isimlendiriyoruz.

STZİ ve TİS algoritmalarında amacımız zamansal kısıtlara öncelik vermek olduğu için partilerin sıralamalarını değiştirmiyoruz.

Yukarıda tanımlanan algoritmaların, Algoritma 3'ün hangi parametreleri ile gerçeklendiği Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1: Kurucu sezgisel algoritmaların Algoritma 3 ile gerçeklendiği parametreler.

İsim	*sirala*1 (veritabanları)	*sirala*2 (partiler)
İS	1	0
İSPK	1	1
İSA	2	2
İSAPA	2	2
İSAPK	2	1
STZİ	3	0
TİS	4	0

3.3 Yerel Arama Algoritmaları

İlk olarak Kutulama problemi için geliştirilen mutasyon operatörleri ve yerel arama algoritmalarında sıkça kullanılan [49, 50, 51, 52, 53, 54] iki önemli fikri açıklayacağız.

Kutu Eleme [49]: Bazı kutuları seç ve buradaki eşyaları kutulardan çıkar. Bu işlem kutu eleme olarak adlandırılmaktadır. Daha sonra çıkarılan eşyaları İS ve İSA gibi sezgisel algoritmalar kullanarak tekrar yerleştir. Burada amaç, elenen kutulardaki eşyaların elenmeyen kutulara aktarılması ile elenmeyen kutuların verimliliğini artırmaktır. Ayrıca, eğer çıkarılan hiçbir eşya elenmeyen kutulara aktarılamasa bile, eğer başlangıç durumunda kötü bir şekilde yerleştirildilerse, sezgisel algoritmalar ile daha iyi bir şekilde yerleştirilebilirler.

Değiştirme Prosedürü [50]: Bu fikir, Kutu Eleme fikrinin etkinliğini artırmayı amaçlamaktadır. Bazı kutular elendikten sonra, çıkarılan eşyalar tekrar yerleştirilmeden önce, çıkarılan eşyaların elenmeyen kutulardaki eşyalarla değiştirilmesi göz önüne alınır ve eğer elenmeyen kutunu verimi (doluluk miktarı) artıyorsa değişim gerçekleştirilir. Değiştirmeler için sadece bir eşya ile bir eşyanın değiştirilmesi değil, iki eşya altkümesinin değiştirilmesi göz önüne alınır. Bu değişimler sonucunda, hem yerleştirilmemiş eşyaların toplam boyutunun azaltılması hem de elenmeyen kutuların verimliliklerinin artırılması amaçlanmaktadır. Değiştirme prosedürü sadece yerel arama için değil genel olarak, bir kısmı çözümün bir sezgisel algoritma ile tamamlanacağı her durumda için kullanılabilir.

Kutu eleme fikrinde, eleme için kaç kutunun seçileceği ve nasıl seçilecekleri önemlidir. Falkenauer ve Delchambre çalışmasında elenecek kutular şu şekilde seçilmiştir: en az doluyu seç ve kalanları rastgele seç [49]. Alok ve Ashok her zaman 5 kutu elemişler ve bu kutuları Falkenauer ve Delchambre çalışmasındakine benzer bir şekilde

seçmişlerdir. Quiroz vd. yukarıdaki tekniklerin iyi yanları birleştirilerek, en az dolu olan k kutuyu elemişlerdir. Burada k sayısı çözümde kullanılan kutu sayısına göre dinamik olarak belirlenmektedir [54].

Bu bölümün geri kalanında, GFVGP için geliştirilen yerel arama algoritmalarını anlatıyoruz. Bölüm 3.3.1'deki yerel arama algoritmaları kutu eleme ve değişim prosedürü fikirlerinden faydalanmaktadır. Dolayısıyla bunları uyarlanan yerel arama algoritmaları olarak adlandırıyoruz.

3.3.1 Uyarlanan Yerel Arama Algoritmaları

Uyarlanan yerel arama algoritmalarında, her zaman 3 parti elenmek için şu şekilde seçilmektedir. Kapasitesi aşılmış ya da zamansal kısıtı ihlal edilen bir veritabanı içeren partilere öncelik verilmektedir. Uyarlanan yerel arama algoritmaları, önceliklendirilen parti sayısı üçten az olduğu durumlarda kalan partilerin nasıl seçildiği yönünden farklılık göstermektedir.

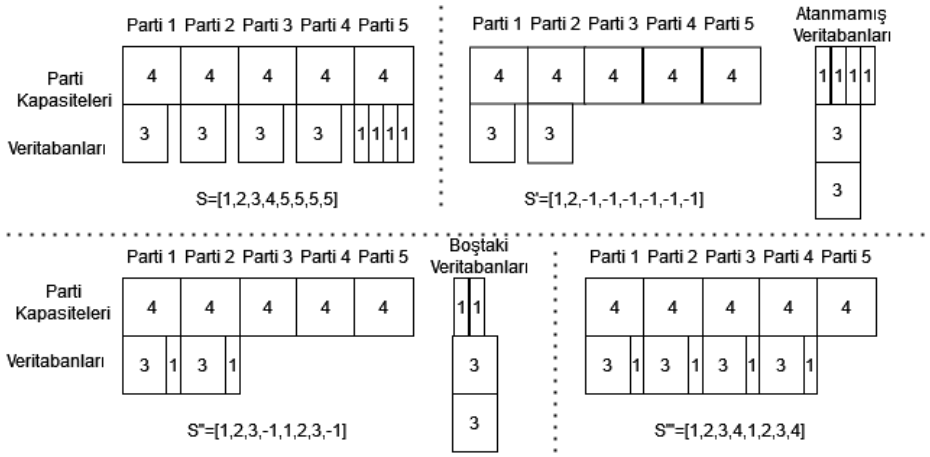
Açgözlü Son Parti (ASP) algoritması, kullanılan partiler içerisinden en büyük numaraya sahip olan partileri seçmektedir. Rastgele Son Parti (RSP) algoritması ASP algoritmasına benzemektedir fakat sadece bir partiyi açgözlü bir şekilde seçmekte ve kalan partileri rastgele seçmektedir. Fark edileceği üzere, bu algoritmalar amaç fonksiyonu *son* olduğu durumlarda efektiftir.

Açgözlü Kullanılan Parti (AKP) algoritması, kullanılan partiler içerisinden en az dolu olan partileri seçmektedir. Rastgele Kullanılan Parti (RKP) algoritması AKP algoritmasına benzemektedir fakat sadece bir partiyi açgözlü bir şekilde seçmekte ve kalan partileri rastgele seçmektedir. Fark edileceği üzere, bu algoritmalar iki amaç fonksiyonu için de efektiftir.

Uyarlanan yerel arama algoritmaları, partiler elendikten sonra aynı şekilde çalışmaktadır. Partiler elendikten sonra, değiştirme prosedürünün zamansal kısıtları göz önüne alacak şekilde uyarlanmış hali uygulanmaktadır. Bu prosedürde boyutu en fazla 2 olan veritabanı altkümelerini göz önüne alınmaktadır. Atanamamış veritabanlarını amaç fonksiyonu *kul* ise İSAPA algoritmasını, amaç fonksiyonu *son* ise İSA algoritmasını kullanarak atıyoruz. Yerel arama algoritması daha iyi bir sonuç bulunamayana kadar tekrarlanır. Diğer çalışmaların aksine, işlem tamamlandıktan sonra çözümde *iyileşme* olmaması durumunda, yeni çözüm değil orijinal çözüm sonuç olarak verilir. Bunun asıl sebebi, bizim rastgele yerel arama yöntemlerini de kullanmayı tercih etmemiz ve bu durumlarda çözümün çok kötüleşebilmesi hatta ve hatta olursuz hale gelebilmesidir.

Şekil 3.1'de AKP algoritmasının iyileştirme yapamadığı fakat RKP algoritmasının iyileştirme yapabildiği bir duruma örnek verilmiştir.

Bu GFVGP problemi örneğinde, sırasıyla boyutları $3, 3, 3, 3, 1, 1, 1, 1$ olan 8 veritabanı ve kapasiteleri 4 olan 8 parti mevcuttur. Hiç zamansal kısıt olmadığını varsayalım. Şekil 3.1'de sol üstte olası bir çözüm gösterilmektedir. Yerden kazanmak için kullanılmamış olan kalan 3 partiye şekilde yer verilmemiştir. Fark edileceği üzere, bu çözüm iki amaç fonksiyonu altında da 5 değerine sahiptir. Örnek için amaç



Şekil 3.1: Rastgele yerel arama algoritması için motivasyonel örnek.

fonksiyonunun *kul* olduğu varsayalım. Bu çözümde yerel arama ile iyileştirme sağlanabilmesi için en dolu parti olan parti 5 elenmelidir. Fakat AKP yerel arama algoritmasında bu parti en son elenecek parti olduğundan, iyileştirme bulunması imkansızdır. Ama RKP yerel arama algoritmasında 3 parti elenecekken, parti 5'in de elenmesi ihtimaller dahilindedir.

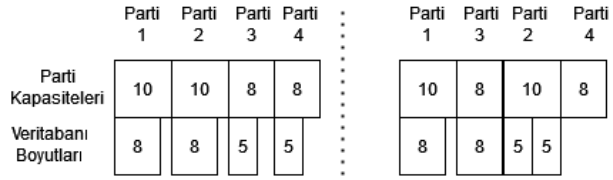
Şekil 3.1'de sağ tarafta parti 3, parti 4 ve parti 5'in elendiği kısmı çözüm görülmektedir. Bu kısmı çözümde 2 tane 3 boyutunda, 4 tane 1 boyutunda veritabanı atanmamıştır. Daha sonra, bu veritabanlarının ikili ve tekli halde dolu olan partilerdeki boş, tekli ve ikili veritabanları ile değişimleri göz önüne alınmaktadır. Bu süreç sonucunda, 2 tane 1 boyutlu veritabanı ilk iki partiye atanmakta ve Şekil 3.1'de sol altta belirtilen 4 veritabanının atanmadığı çözüm elde edilmektedir. Daha sonra bu 4 veritabanı İSAPA algoritması kullanılarak atanmakta ve Şekil 3.1'de sağ alt tarafta görülen çözüme ulaşılmaktadır. Bu çözümde dört parti kullanılmaktadır.

3.3.2 Parti Karıştırma Algoritması

Bu yerel arama algoritmalarına ek olarak, çözümü iyileştirmek için kullandığımız bir başka yöntem ise kullanılan partileri ve/veya partilerin içeriklerini değiştirmektir.

Parti Karıştırma Algoritması: *o* bir çözümün amaç fonksiyonu değerini göstersin. Parti Karıştırma (PK) algoritması, çözümde kullanılan her partideki veritabanlarının tamamı büyük tek bir veritabanı gibi ele alınır. Yani, bu işlem sonunda, en fazla *o* veritabanımız olur. Eğer amaç fonksiyonu *kul* ise, en büyük kapasiteye sahip *o* parti seçilir, amaç fonksiyonu *son* ise ilk *o* parti seçilir. Daha sonra, bu büyük veritabanları bu partilere İSAPK algoritması ile yerleştirilir. Elde edilen çözüm olurlu ise amaç fonksiyonu değeri kötüleşmemiş olacaktır, fakat, veritabanları büyük ihtimalle farklı partilere yerleştirilmiş olacaktır. Dolayısıyla elde edilen çözüm olurlu ise, bu çözüm dönülür. Aksi taktirde orijinal çözüm dönülür.

PK algoritmasının motivasyonu, büyük partilerin içeriklerini sığdıkları daha küçük bir partiye aktararak büyük partiyi daha efektif bir şekilde kullanmaktadır. Şekil 3.2'te PK algoritmasının yarar sağladığı bir duruma örnek verilmiştir.



Şekil 3.2: Parti Karıştırma algoritması için motivasyonel örnek.

Şekil 3.2'deki örnekte, boyutları sırasıyla $8,8,5,5$ olan 4 veritabanı ve kapasiteleri $10,10,8,8$ olan 4 parti bulunmaktadır. Hiçbir zamansal kısıt olmadığını varsayalım. Başlangıçta Şekil 3.2'de sağ tarafta görülen $S = [1,2,3,4]$ çözümünde, veritabanları partilere sırayla yerleştirilmiştir. S verilen örnek için İSAPA, İSA ve İS algoritmaları sonucunda elde edilebilen karşılaşılması oldukça olası bir çözümdür. S, önceki bölümde tanımlanan yerel arama algoritmaları ile iyileştirilemeyecektir. Fakat PK algoritması içerisinde kullanılan İSAPK algoritması esnasında parti sıralamaları Şekil 3.2'nin sağ tarafta görüldüğü gibi sıralanırsa, $S' = [1,3,2,2]$ çözümü elde edilir.

3.3.3 Ardışık Yerel Arama Algoritması

PK algoritması göreceli zamansal kısıtların olmadığı örneklerde yararlı olsa da, göreceli zamansal kısıtların olduğu örneklerde partiler karıştırıldığı için genellikle elde edilen çözümler olurlu olmamaktadır. Dolayısıyla, göreceli zamansal kısıtların olduğu örnekler için karıştırma algoritması yerine aşağıda tanımlanan Ardışık Yerel Arama (AYA) algoritmasından yararlanıyoruz. AYA algoritması, dağıtılacak partileri ardışık olarak seçmektedir. Bunun ana sebebi, göreceli zamansal kısıtların olduğu örneklerde, elde edilen çözümlerde birçok veritabanının, bir önceki veya sonraki partideki veritabanlarının yeri değiştirilmediği sürece yerinden oynatılamamasıdır.

Ardışık Yerel Arama Algoritması: o, bir çözümün amaç fonksiyonu değeri olmak üzere, 1 ile $o-2$ arasında bir i tamsayısını rastgele seçilir. Çözümdeki i, $i+1$ ve $i+2$ partileri elenir. Atanmamış veritabanlarını, amaç fonksiyonu *kul* ise İSAPA ve TİS algoritmalarıyla, amaç fonksiyonu *son* ise İSA ve TİS algoritmalarıyla atanarak iki farklı çözüm elde edilir. Orijinal çözüm ve elde edilen iki çözüm içerisinden en iyi olan dönülür. AYA algoritmasını sadece göreceli zamansal kısıtlar varken kullanıyoruz.

3.3.4 Tamir Algoritmaları

Zamansal kısıtların olduğu örneklerde İS ve İSA algoritmaları ve bunların farklı versiyonları zamansal kısıtları önceliklendirmedikleri için olurlu çözüm bulmakta zorlanabilmektedir. Dolayısıyla zamansal kısıtların olduğu örneklerde, bütün yerel arama algoritmalarından önce eğer çözüm olurlu değil ise aşağıda tanımlanan tamir algoritmaları kullanılarak çözümlerin olurlu hale getirilmesi amaçlanmaktadır.

Tamir Algoritmaları: Kapasitesi ihlal edilen veya zamansal kısıtı ihlal edilmiş bir veritabanı içeren bütün partileri ele. Elenen partilerdeki veritabanlarını, sadece kesin zamansal kısıtlar varsa, STZİ algoritmasıyla, göreceli zamansal kısıtlar varsa TİS algoritmasını kullanarak yerleştir.

3.4 Memetik Algoritmalar

Bu alt bölümde genetik algoritmaların nasıl çalıştığı açıklanacak ve Kutulama problemi için geliştirilen genetik algoritmalar ele alınacaktır. Daha sonra, GFVGP probleminin farklı modelleri için geliştirmiş olduğumuz memetik algoritmalar anlatılacaktır.

Genetik algoritmalar, doğadaki biyolojik evrim sürecini bireylerin oluşturduğu bir popülasyonu seçim, çaprazlama ve mutasyon operasyonları kullanarak nesiller boyu evrimleştirerek taklit eder [55]. Genetik algoritma, rastgele çözümlerin bireylere karşılık geldiği ilk nesille başlar [56]. Bireyler genlerden meydana gelir ve aslında bu çözümün nasıl temsil edileceğiyle alakalıdır [55]. Genellikle bireyler sabit sayıda genden oluşmakta ve bireyler bir bit dizi olarak temsil edilmektedir [55]. Fakat değişken sayıda genden oluşan daha karmaşık çözüm temsilleri de kullanılmaktadır. Fitlik fonksiyonu adı verilen bir fonksiyon kullanılarak bireylerin ne kadar başarılı (fit) oldukları ölçülür [57]. Her nesilde, seçim kuralları kullanılarak popülasyondan bazı bireyler ebeveyn olarak seçilir [58]. Seçim kuralları genellikle fit bireyleri kayırarak, zayıf bireylerin genlerini aktarmadan ölüp gitmelerini ve fit bireylerin genlerinin yeni nesillere aktırılmasını sağlar [59]. Ebeveynlerin genleri çaprazlama operatörleri ile çaprazlanarak (karıştırılarak) bir ya da birden fazla yavru oluşturulur [55]. Oluşturulan her yavru, popülasyondaki bir bireyle değiştirilir. Popülasyonun nasıl evrileceğini kontrol etmek için literatürde kullanılan birçok evrim şeması bulunmaktadır. Eksiksiz değişim semasında her yeni nesil tamamen yeni yavrulardan oluşurken, durağan-durum evrim şemasında her nesilde popülasyondan sadece bir birey yeni bir yavru ile değiştirilir [57]. Bunların yanı sıra yaygın olarak kullanılan bir başka evrim şeması ise seçkinlikli değişim şemasıdır. Bu evrim şemasında, popülasyonun seçkin ufak bir kısmı yeni nesile aktarılırken geri kalan bireyler yeni yavrularla değiştirilir [57]. Genetik algoritmalar popülasyonda çeşitlilik sağlanamazsa/korunamazsa zamanından önce bir yerel optimal çözüme yakınsayabilirler [60]. Dolayısıyla, popülasyondaki çeşitliliğin artırılması için bireylerin genlerinde rastgele değişiklikler meydana getiren mutasyon operatörleri kullanılır [61]. Mutasyonun yanı sıra, özellikle durağan-durum evrim şemasında, yavrunun popülasyondan hangi bireyin yerine geçeceği de çeşitliliği etkileyen faktörlerdendir [62]. Genetik algoritmaların yerel arama algoritmaları ile birlikte kullanılan halleri, hibrit genetik algoritma ya da memetik algoritma olarak adlandırılmaktadır [63].

Kutulama problemi için geliştirilen genetik algoritmalar: Kutulama problemi için ilk genetik algoritma Falkenauer ve Delchambre tarafından geliştirilmiştir [49]. Bu çalışmada daha sonra Gruplayıcı Genetik Algoritma (GGA) olarak da adlandırılacak yeni bir genetik algoritma varyantı önerilmiştir. Falkenauer ve Delchambre, genetik algoritmada kullanılan bir eşyanın hangi kutuya konduğunu gösteren (klasik) çözüm temsillerinin Kutulama problemi için iyi çalışmadığını göstermiş ve aynı grupta bulunan eşyaların bir gen olarak temsil edildiği ve bireylerin farklı sayıda genden oluştuğu bir çözüm temsili önermiştir. Ayrıca, optimize edilecek amaç fonksiyonu olarak kullanılan kutu sayısı yerine, kutu verimliliğinin karelerinin ortalamasının kullanılmasının daha efektif olduğunu belirtmişlerdir. Bu sayede, aynı sayıda kutu kullanan çözümler kendi içlerinde sıralanabilmektedir. Bu fonksiyon, neredeyse dolu ya da neredeyse boş kutuları yarı yarıya dolu kutulara tercih etmektedir. Bu sayede

az dolu kutulardaki eşyalar daha kolay bir şekilde başka bir kutuya taşınabilir. Önerdiği çözüm temsili için yeni çaprazlama ve mutasyon operatörleri önermiştir. Çaprazlama operatörleri çok noktalı çaprazlama operatörü gibi çalışmakta fakat direkt olarak kutular yavruya aktarıldığından bir eşya birden fazla kutuya atanması ya da seçilen genlerde bazı eşyalara dair bilgilerin olmaması gibi problemlerle karşılaşılabilmektedir. Birden fazla atanan eşyalar için sadece ilk atama göz önünde bulundurulur, hiç atanmamış eşyalar ise İSA algoritması kullanılarak atanır. Klasik mutasyon operatörleri genellikle genlerde rastgele değişimler yaparlar, fakat, bunun Kutulama problemi için çok yıkıcı olduğu belirtilmiş ve önceki bölümde anlatılan kutu eleme fikrini bir mutasyon operatörü olarak kullanmışlardır. İlk jenerasyon İS algoritması ile yaratılmıştır. Falkenauer başka bir çalışmasında önceki GGA algoritmasındaki çaprazlama ve mutasyon operatörüne değiştirme prosedürünü eklemiştir [50]. Reeves, Kutulama problemi için Falkenauer önerdiği yöntemlerde İSA algoritması yerine başka sezgisel algoritmalar kullanarak birçok deney yapmıştır [64]. Sonraki Sığan, İS ve En İyi Sığan algoritmalarını denemiş ve en iyi çalışanların İS ve En İyi Sığan algoritmaları olduğun belirtmiştir. Bu iki algoritmanın performansları birbirine çok yakın çıkmış ve farklı deneylerde farklı algoritmalar daha iyi çıkmıştır. Hussain ve Sastry Kutulama problemi için klasik genetik algoritmanın performansını İSA ve En İyi Sığan Azalan algoritmalarıyla karşılaştırmıştır ve genetik algoritmanın daha iyi sonuç verdiğini belirtmiştir [65]. Iima ve Yakawa, Kutulama problemi için Falkenauer'in önerdiğine benzer bir çözüm yapısını kullanan fakat farklı çaprazlama operatörü kullanan bir genetik algoritma geliştirmiştir [66]. Önerilen çaprazlama operatöründe iki ebeveynden rastgele seçilmiş bazı kutuları yavruya aktırılır ve değişim prosedürü uygulanır. Sonra atanmamış kalan eşyalar İS algoritması ile atanır. Mutasyon operatörü olarak [49]'deki mutasyon operatörü kullanılmıştır. Geliştirilen algoritma öncekilerle karşılaştırılmamıştır. Son olarak, Quiroz vd. literatürdeki güzel fikirleri birleştiren bir GGA önermiştir. Bu GGA [49, 50] çalışmalarında kullanılan çözüm temsilini kullanmıştır. İlk jenerasyon oluşturulurken İS algoritmasının, ilk olarak kutu kapasitesinin yarısından büyük eşyaları atayıp sonra kalanları atayan bir versiyonu kullanmıştır. Önerdikleri çaprazlama operatörü, iki ebeveyndeki kutuları doluluklarına göre büyükten küçüğe sıralar. Daha sonra, her adımda iki ebeveynin en büyük kutularını karşılaştırır ve kutuları daha dolu olan önce gelecek şekilde yavruya aktarır. Birden fazla atama olması halinde, ikinci atama göz ardı edilir. Mutasyon operatörü olarak [50] çalışmasındakine benzer bir mutasyon operatörü kullanmış fakat elenecek kutu sayısı dinamik olarak belirlenmiştir.

Şimdi de GFVGP problemi için geliştirmiş olduğumuz Memetik Algoritmaları anlatıyoruz.

GFVGP problemi için Memetik Algoritmalar: Memetik algoritmalarımız, her jenerasyonda tek bir yavrunun oluşturulduğu durağan-durum evrim şemasını [67] kullanmaktadır. Memetik algoritmalarımız, GFVGP problemi için spesifik olarak geliştirilmiş 2 çaprazlama ve 2 mutasyon operatörü içermektedir. Bu operatörler farklı modeller için temelde benzer bir şekilde çalışsalar da farklılaştıkları noktalar mevcuttur. Çaprazlama ile elde edilen yavruların popülasyondaki hangi bireyin yerine geçeceği, popülasyonun çeşitliliğini korumak için uygunluğa ek olarak bireyler arasındaki benzerliği de dikkate alan bir *ayrılma kuralı* ile seçilmektedir. İlk jenerasyon önceki bölümlerde anlatılan kurucu sezgisel algoritmalar ile oluşturulup yerel arama algoritmalarıyla iyileştirilmektedir.

Çözüm Temsili: [49, 50, 54] çalışmalarındakilere benzer bir şekilde gruplayıcı bir çözüm temsili kullanıyoruz. Bu temsilde her gen, bir partiye ve bu partiye atanan veritabanlarına karşılık gelmektedir. Dolayısıyla, her birey, kullanılan parti sayısı kadar genden oluşmaktadır.

Uygunluk Fonksiyonu: [49, 50, 54] çalışmalarındakilere kullanılan uygunluk fonksiyonunun GFVGP probleminde bulunan zamansal kısıtları göz önünde alan bir versiyonunu kullanıyoruz. L_{maks} en büyük parti kapasitesini göstersin. Bir p bireyinin uygunluğu

$$U(p) = \frac{1}{AFD} \sum_{i \in KP} \left(\frac{A_i}{L_{maks}} \right)^2 - AFD - KI - 1000 \cdot (ZI)$$

şeklinde hesaplanır. Burada, A_i, parti i'deki veritabanlarının toplam boyutunu gösterir. KP, p'de kullanılan partilerin kümesidir. AFD, p'nin amaç fonksiyonu değerini göstermektedir. KI toplam kapasite ihlalini, ZI ise ihlal edilen zamansal kısıt sayısını göstermektedir. Uygunluk fonksiyon değeri yüksek olan bireyler daha iyi kabul edilmektedir.

Bireyler arası benzerlik: İki bireyin benzerliği aynı partiye atanan veritabanı sayısı ile ölçülmektedir. Dolayısıyla, iki bireyin benzerliği 0 ile n arasında bir tamsayıya karşılık gelmektedir.

İlk neslin oluşturulması: Algoritmaların bu kısmı, modeller arasında değişiklik göstermektedir. P popülasyonu temsil etsin ve popülasyondaki birey sayısını $|P|$ ile ifade edelim.

$\langle kyf \mid kyf \mid \phi \mid kul \rangle$ ve $\langle kyf \mid kyf \mid ksn \mid kul \rangle$ notasyonları ile ifade edilen GFVGP modelleri için, $|P|-2$ birey İSPK algoritmasıyla, bir birey İSA algoritmasıyla, bir birey ise İSAPA algoritmasıyla oluşturulmaktadır.

$\langle kyf \mid kyf \mid \phi \mid son \rangle$ ve $\langle kyf \mid kyf \mid ksn \mid son \rangle$ notasyonları ile ifade edilen GFVGP modelleri için, $|P|-1$ birey İS algoritmasıyla, bir birey ise İSA algoritmasıyla oluşturulmaktadır.

$\langle kyf \mid kyf \mid grc \mid son \rangle$ notasyonu ile ifade edilen GFVGP örnekleri için, $|P|-1$ birey TİS algoritmasıyla, bir birey ise İSA algoritmasıyla oluşturulmaktadır.

$\langle kyf \mid kyf \mid grc \mid kul \rangle$ notasyonu ile ifade edilen GFVGP modelleri için, ilk olarak bir birey TİS algoritmasıyla oluşturulur. o, bu bireydeki kullanılan parti sayısını göstersin. Daha sonra, $|P|-3$ birey, TİS algoritmasının sadece rastgele seçilmiş $o+5$ partinin kullanıldığı modifiye edilmiş TİS algoritmasıyla oluşturulmaktadır. Kalan bireylerden biri İSA algoritmasıyla, diğer birey ise İSAPA algoritmasıyla oluşturulmaktadır.

Oluşturulan bireyler, amaç fonksiyonu *kul* ise RKP yerel arama algoritması ile, amaç fonksiyonu *son* ise ASP yerel arama algoritması ile iyileştirilir.

Fark edileceği üzere ilk jenerasyondaki birkaç birey açgözlü algoritmalar ile kalan bireyler ise rastgele algoritmalar ile oluşturulmaktadır. Göreceli zamansal kısıtların bulunduğu modellerde ise İS veya İSPK algoritmaları olurlu çözümler bulmakta yetersiz kaldıkları için TİS algoritması kullanılmıştır.

Çaprazlama Operatörleri: Algoritmalarımız çok-ebeveynli bir açgözlü c_1 ve bir rastgele c_2 olmak üzere iki çaprazlama operatörü kullanmaktadır. Bu operatörler birbirlerine benzer bir şekilde çalışmakta ve sadece ebeveynlerin partilerini farklı sıralarda göz önünde bulundurmaktadırlar. Ebeveynler ikili turnuva yöntemiyle seçilmektedir. c_1 ebeveynlerin partilerini anahtarları partilerin doluluk miktarları olmak üzere bir öncelikli kuyruğa koymaktadır. c_2 ise ebeveynlerin partilerini rastgele olarak bir kuyruğa koymaktadır. Buradan sonra iki operatör aynı şekilde çalışmaktadır. o en iyi ebeveynin amaç fonksiyonu değerini göstersin. Yavruya o parti aktarılana kadar ya da kuyrukta hiç parti kalmayana kadar, kuyruktan bir parti seçilir ve *göz ardı edilme* durumlarından biri sağlanmıyorsa $1 - \frac{1}{o}$ olasılıkla yavruya aktarılır.
Göz ardı edilme durumları:

- Seçilen partinin kapasitesi aşılmışsa,
- Seçilen parti zamansal kısıtı ihlal edilen bir veritabanı içeriyorsa,
- Parti daha önce yavruya aktarıldıysa,
- Partideki herhangi bir veritabanı daha önce yavruya aktarıldıysa,

Bu işlem tamamlandıktan sonra, hala atanmamış veritabanları kalabilir. Atanmamış veritabanları amaç fonksiyonu *kul* ise, İSAPA ve İSA algoritmalarıyla, amaç fonksiyonu *son* ise, İSA ve İS algoritmalarıyla atanarak iki farklı yavru elde edilir. Bunlardan daha fit olan yavru amaç fonksiyonuna göre AKP veya ASP yerel arama algorimalarıyla iyileştirilerek yeni nesile aktarılır.

Ayrılma kuralı: Bir yavru meydana geldiğinde bu yavrunun popülasyonda yerine geçeceği birey şu şekilde seçilmektedir. Eğer popülasyonda olursuz çözümler varsa bunlardan biri seçilir. Eğer popülasyondaki her birey olurluysa ve yavrudan daha fitse, yavru göz ardı edilir. Aksi taktirde, popülasyondaki bireylerden yavrudan daha az ya da aynı fitliğe sahip bireyler arasından, yavruya en çok benzeyen birey seçilir. Fark edileceği üzere, eğer yavrunun popülasyondaki bir bireyle aynı olması durumunda, popülasyona tekrar eklenmeyecektir.

Mutasyon Operatörleri: Memetik algoritmalarımız m_1 ve m_2 olarak adlandırdığımız iki mutasyon operatörü kullanmaktadır. İki operatörde göz önündeki bulundurulan GFVGP modeline göre, bir ya da iki yerel arama algoritması kullanmaktadır.

Amaç fonksiyonu *kul* ise, m_1 sadece RKP rastgele yerel arama algoritmasını kullanmaktadır. Amaç fonksiyonu *son* ise, m_1 sırasıyla RKP ve RSP algoritmalarını kullanmaktadır.

Amaç fonksiyonunun *kul* olduğu ve göreceli zamansal kısıtların olmadığı modeller için m_2 sırasıyla, PK ve RKP algoritmalarını, göreceli zaman kısıtlarının olduğu modeller içinse, sırasıyla AYA ve RKP algoritmalarını kullanır. Amaç fonksiyonunun *son* olduğu ve göreceli zamansal kısıtların olmadığı modeller için m_2 sırasıyla, PK ve RSP algoritmalarını, göreceli zaman kısıtlarının olduğu modeller içinse, sırasıyla AYA ve RSP algoritmalarını kullanır. Çizelge 3.2 mutasyon operatörleri tarafından kullanılan yerel arama algoritmalarını göstermektedir.

Çizelge 3.2: Mutasyon operatörleri tarafından kullanılan yerel arama algoritmaları.

Model	m_1	m_2
$\langle kyf \mid kyf \mid \phi \mid kul \rangle$ ve $\langle kyf \mid kyf \mid ksn \mid kul \rangle$	RKP	PK, RKP
$\langle kyf \mid kyf \mid \phi \mid son \rangle$ ve $\langle kyf \mid kyf \mid ksn \mid son \rangle$	RKP, RSP	PK, RSP
$\langle kyf \mid kyf \mid grc \mid kul \rangle$	RKP	AYA, RKP
$\langle kyf \mid kyf \mid grc \mid son \rangle$	RKP, RSP	AYA, RSP

Memetik algoritmalarda, mutasyon ve çaprazlama operatörlerinin seçilme olasılıkları ve çaprazlama operatörünün ebeveyn sayısı deneysel olarak seçilmiştir. Seçim sürecinin deneylerini bir sonraki bölümde anlatıyoruz.

4. DENEYSEL SONUÇLAR

Bu bölümde, ele alınan farklı modellerin başarımının ölçülmesinde kullanılan sentetik örneklerin nasıl üretildiği anlatılacak, geliştirilen memetik algoritmaların parametreleri seçilecek ve başarımları ölçülecektir. Sentetik örnekler üretilirken, Kutulama ve Basit Montaj Hattı Dengeleme problemleri için sentetik veri üretilen çalışmalardakilere benzer bir yöntem kullanılmaktadır. Örnekler üretildikten sonra, algoritmanın farklı parametre seçeneklerinin başarımları incelenecek ve en başarılı parametre seçenekleri belirlenecektir. Son olarak, belirlenen parametre seçenekleri için algoritmanın başarımı daha büyük bir örnek kümesinde incelenecektir.

4.1 GFVGP problemi örneklerinin oluşturulması

Tezde ele alınan problem daha önce çalışılmadığından ve veritabanı verileri firmalar için hassas bilgi olarak değerlendirilip paylaşılmadığından sentetik örnekler üretilmiştir.

Zamansal kısıtların olmadığı modeller için, 180 tane örnek literatürde Kutulama problemi için üretilen örneklerin üretilişine benzer [68] bir şekilde üretilmiştir. Burada veritabanı boyutları ve parti kapasiteleri parametreleri için üçer farklı dağılımdan sayılar ürettik. Bu 9 farklı dağılım kombinasyonu için $n = 50$ ve $n = 100$ olan onar örnek ürettik. Dolayısıyla, her iki amaç fonksiyonu içinde 180 örnek olmak üzere toplam 360 üretmiş olduk. Veritabanı boyutlarının ve parti kapasiteleri aşağıdaki dağılımlardan rastgele seçilmiştir:

- *Veritabanı boyutları:* $w_i \in [25,40],[20,120],[40,100]$.
- *Parti kapasiteleri:* $l_i \in [34,166],[50,150],[67,133]$.

Sadece kesin zamansal kısıtların olduğu modellerin örneklerini üretmek için ise, zamansal kısıtların olmadığı modeller için üretilmiş 360 örnekteki veritabanlarına sürülme ve son teslim zamanları aşağıda anlatıldığı gibi oluşturuldu. Amaç fonksiyonu *kul* olan modeller için, her veritabanı B_i için, ilk olarak 1 ile n arasında bir sürülme zamanı seçilip, daha sonra bu sürülme zamanı ile n arasından son teslim zamanı seçildi. Amaç fonksiyonu *son* olan modeller için bu şekilde örnek üretilirse, bir veritabanı için bile seçilen sürülme zamanı yüksek olması halinde örnek kolayca çözülebilir. Dolayısıyla, amaç fonksiyonu *son* olan örnekler için, ilk olarak bir alt sınır şu şekilde hesaplanıyor. j, ilk j partinin kapasiteleri toplamının, veritabanı boyutları toplamından büyük veya eşit olduğu en küçük tamsayı olsun. j, örnek için bir alt sınır olacaktır çünkü optimal çözümde kesinlikle ilk j veya daha fazla parti kullanılmak zorundadır. Bu alt sınır hesaplandıktan sonra, her veritabanı B_i için, 1 ile j arasında bir sürülme zamanı seçildi ve daha sonra bu sürülme zamanı ile n arasından son teslim zamanı

seçildi. Bu şekilde, sadece kesin zamansal kısıtlarının olduğu iki modelin her biri için 180 olmak üzere toplam 360 örnek oluşturuldu.

Göreceli zamansal kısıtların olduğu modeller için örnekleri, Basit Montaj Hattı Dengeleme (BMHD) problemi için üretilmiş olan örnekler şu şekilde değiştirilerek oluşturuldu:

- BMHD örneklerindeki öncelik ilişkileri, eşit olasılıkla sıkı veya gevşek göreceli zamansal kısıtlara dönüştürüldü.
- BMHD örneklerindeki iş süreleri veritabanı boyutları olarak kullanıldı.
- BMHD örneklerinde çevrim süreleri eşit ve 1.000'e eşittir. GFVGP problemi örneklerinde parti kapasiteleri şu üç dağılımdan rastgele olarak seçilmiştir: $[340, 1660], [500, 1500], [670, 1330]$.
- Her veritabanına $0,16$ olasılıkla ya bir sürülme zamanı ya bir son teslim zamanı, 0.04 olasılıkla ise hem bir sürülme zamanı hem de bir son teslim zamanı, modelin amaç fonksiyonuna göre yukarıda anlatıldığı gibi seçilmiştir.

Algoritmaların parametrelerinin seçilmesinde kullanılan örnekler şu şekilde oluşturulmuştur: her model için yukarıdaki gibi veritabanı sayısının 50 olduğu 45 örnek oluşturulmuştur. Bu seti parametre seçim kümesi olarak adlandırıyoruz.

4.2 Deney Düzeneği

Bütün algoritmalar Java programlama dilinde yazıldı ve JDK versiyonu 11.01 ile derlendi. Çalıştırma esnasında ise $8GB$ yığıt boyutu verildi. Tamsayılı lineer program CPLEX Java arayüzü ile yazıldı, $8GB$ bellek limiti ve 8 saat zaman limiti ile 12 iş parçacığı kullanacak şekilde çalıştırılmıştır. Bütün deneyler, Windows 10 Education yüklü 3.7Ghz (4.3Ghz Turbo) 64-bit AMD Ryzen 7 2600X 6-çekirdekli 12-iş parçacıklı bir işlemciye ve 16GB DDR4 3200MHz RAM'e sahip tek bir makinede gerçekleştirildi. CPLEX versiyonu olarak 12.62 kullanıldı.

4.3 Algoritmaların Parametrelerinin Seçilmesi

Bu bölümde algoritmaların farklı parametre seçenekleri altındaki performansları incelenecektir. Bunun için yukarıda anlatılan her model için 45 örnekten oluşan örnek seti kullanılmıştır. Bütün parametre seçeneklerinde, $0,2$ olasılıkla bir mutasyon operasyonu, $0,8$ olasılıkla ise bir çaprazlama operasyonu seçilmiştir.

Çaprazlama operatörlerinde kullanılacak ebeveyn sayısı parametresine kısaca ebeveyn sayısı diyoruz ve p ile gösteriyoruz. Mutasyon operasyonu seçildiğinde m_1 ve m_2'nin seçim olasılıklarını o_{m_1} ve o_{m_2} ile ifade diyoruz. Benzer bir şekilde, çaprazlama operasyonu seçildiğinde c_1 ve c_2'nin seçim olasılıklarını o_{c_1} ve o_{c_2} ile ifade diyoruz. Fark edileceği üzere $o_{m_2} = 1 - o_{m_1}$'dir. Benzer bir şekilde $o_{c_2} = 1 - o_{c_1}$'dir.

p, o_{m_1} ve o_{c_1} parametrelerinin her biri için 3 farklı deney, dolayısıyla toplam 27 deney yapıldı. Ebeveyn sayısı için, $p = 2$, $p = 3$ ve p'nin $\{2,3\}$ kümesinden eşit ihtimalle rastgele seçildiği parametre seçeneklerini test edildi. Mutasyon operatörlerinin seçim olasılıkları için o_{m_1}'in $1/3, 1/2, 2/3$ değerlerini aldığı parametre seçenekleri, çaprazlama operatörlerinin seçim olasılıkları için o_{c_1}'in $1/3, 1/2, 2/3$ değerlerini aldığı parametre seçeneklerini test edildi.

İlk olarak, parametre seçim kümesindeki örnekler için tamsayılı lineer programı CPLEX ile çalıştırılarak optimal çözümü ya da optimal çözülemediyse bir alt sınır elde edildi. Her model için oluşturulan 45 örnekten, CPLEX, $\langle kyf \mid kyf \mid \phi \mid kul \rangle$ ve $\langle kyf \mid kyf \mid grc \mid kul \rangle$ notasyonları ile ifade edilen modellerin sırasıyla 1 ve 8 örneğini optimal olarak çözemedi. Diğer kalan bütün örnekler optimal olarak çözüldü. Bütün örnekler için, memetik algoritmaların performansları, CPLEX tarafından bulunan alt sınır ile karşılaştırılarak ölçüldü. Bu alt sınır optimal çözülen örnekler için amaç fonksiyonu değerini eşittir.

Bütün deneylere, her parametre seçeneği için memetik algoritmalar 45 örneğin her biri için on farklı rastgele çekirdek ile 1.000 jenerasyon boyunca çalıştırıldı. Alt sınırdan ortalama yüzde uzaklığı ve optimal çözülen örnek sayısını bildirildi.

Sonuçlar Çizelge 4.1, Çizelge 4.2 ve Çizelge 4.3'de görülmektedir. Her çizelge, p parametresinin farklı bir değeri için farklı operatör seçim olasılıklarının sonuçlarını göstermektedir. Örneğin Çizelge 4.1, $p = 2$ için 9 farklı operatör seçim olasılığı için elde edilen sonuçları göstermektedir. Çizelgelerin her hücresinin üst kısmında, memetik algoritmalarca bulunan çözümlerin alt sınırdan ortalamada yüzdece ne kadar uzak olduğu, hücrenin alt kısmında parantez içinde ise bulunan çözümlerin kaçının optimal olduğu belirtilmektedir. Bir çözümün, alt sınıra yüzde uzaklığını şu şekilde hesaplanır: $100 * \left(1 - \frac{\textit{Bulunan Çözümün Amaç Fonksiyonu Değeri}}{\textit{Alt Sınır}}\right)$. Ortalamalar alınırken geometrik ortalama kullanılmıştır. Hatırlayacağınız üzere, her parametre seçeneği, 45 örneğin her biri için 10 kere çalıştırıldığı için, parantez içindeki değerler en fazla 450 olabilir. En iyi sonucu veren parametre seçenekleri kalın yazılmıştır.

Çizelgelerden görülebileceği üzere, ebeveyn sayısının $\{2,3\}$ kümesinden rastgele seçildiği parametre seçenekleriyle, 4 model için en iyi sonuç elde edilmiştir. Aynı zamanda, ebeveyn sayısının 3 olduğu parametre seçenekleriyle, 4 model için en iyi sonuç elde edilmiştir. Fakat ebeveyn sayısının 2 olduğu parametre seçenekleriyle, sadece 2 model için en iyi sonuç elde edilmiştir. Bu gözlem, geliştirmiş olduğumuz çok ebeveynli çaprazlama operatörlerinin diğer çalışmalarda kullanılan iki ebeveynli operatörlere göre daha efektif olduğunu göstermektedir.

$\langle kyf \mid kyf \mid \phi \mid kul \rangle$ notasyonu ile ifade edilen modeller için, bütün parametre seçenekleri neredeyse bütün örnekleri optimal çözmüştür ve 6 parametre seçeneği aynı ortalama uzaklığa sahiptir. Bu model için, ebeveyn sayısının $\{2,3\}$ kümesinden rastgele seçildiği parametre seçeneklerinden birini tercih ediyoruz çünkü en iyi sonucun elde edildiği 6 parametre seçeneğinin 3 tanesinde ebeveyn sayısı bu parametre değerine sahiptir. Bunların içinde ise, mutasyon ve çaprazlama operatörlerinin eşit olasılıkla seçildiği parametre seçeneğini tercih ediyoruz çünkü bu parametre seçeneğiyle diğerler parametre seçeneklerine göre daha fazla optimal çözüm bulunmuştur.

Çizelge 4.1: Çaprazlama operatörlerinde 2 ebeveyn kullanıldığında, operatör seçim olasılıklarının model başarımına etkisi.

operatör seçim olasılıkları				kul			son		
o_{m_1}	o_{m_2}	o_{c_1}	o_{c_2}	ϕ	*ksn*	*grc*	ϕ	*ksn*	*grc*
2/3	1/3	1/2	1/2	0,33 (426)	3,24 (159)	12,98 (121)	0,66 (347)	1,75 (264)	1,16 (367)
1/2	1/2	1/2	1/2	0,33 (426)	3,330 (167)	12,78 (112)	0,72 (340)	1,56 (277)	1,21 (366)
1/3	2/3	1/2	1/2	0,34 (425)	3,27 (162)	13,37 (114)	0,78 (334)	1,60 (268)	1,16 (370)
2/3	1/3	1/3	2/3	**0,31 (428)**	3,35 (174)	**12,24 (116)**	0,65 (350)	1,50 (276)	1,18 (368)
1/2	1/2	1/3	2/3	0,34 (424)	3,24 (170)	12,27 (120)	0,65 (349)	1,46 (287)	1,23 (365)
1/3	2/3	1/3	2/3	**0,31 (428)**	3,42 (168)	13,04 (114)	0,71 (342)	1,49 (282)	1,18 (366)
2/3	1/3	2/3	1/3	0,34 (425)	3,84 (152)	13,06 (112)	0,63 (351)	2,00 (247)	1,13 (370)
1/2	1/2	2/3	1/3	0,34 (424)	4,15 (146)	13,38 (111)	0,69 (341)	1,85 (256)	1,15 (367)
1/3	2/3	2/3	1/3	0,35 (424)	3,91 (143)	14,29 (97)	0,68 (343)	1,82 (253)	1,19 (367)

En iyi sonucun birden fazla parametre seçeneği tarafından elde edildiği diğer bir model ise, $\langle kyf \mid kyf \mid grc \mid son \rangle$ notasyonuyla ifade edilen modeldir. Bu model için, ebeveyn sayısının $\{2,3\}$ kümesinden rastgele seçildiği parametre seçeneğini tercih ediyoruz çünkü bu parametre seçeneğiyle diğer parametre seçeneklerine göre daha fazla kere optimal çözüm bulunmuştur.

Kalan modeller için, en iyi sonuç sadece bir parametre seçeneğiyle elde edildiği için kalan modeller için en iyi sonuçların elde edildiği parametre seçeneklerini tercih ediyoruz.

Çaprazlama operatörü seçim olasılıklarına göre sonuçlar incelendiğinde, c_2 operatörünün daha yüksek olasılıkla seçildiği parametre seçeneklerinin, diğerlerine göre genelde daha iyi sonuçlar verdiği görülmektedir.

Mutasyon operatörünü seçim olasılıklarına göre sonuçlar incelendiğinde, m_2 operatörünün daha yüksek olasılıkla seçildiği parametre seçeneklerinin, özellikle zamansal kısıtların olduğu modellerde diğerlerine göre daha iyi sonuçlar verdiği görülmektedir. Bu gözlem, karıştırma algoritmasının ve ardışık yerel arama algoritmalarının yararlı olduğunu göstermektedir.

4.4 Memetik Algoritmaların Başarımının Ölçülmesi

Memetik algoritmalarımızın seçilen parametre seçenekleri için performanslarını her model için 180 örnekten oluşan daha büyük bir örnek kümesinde ölçüyoruz.

Çizelge 4.2: Çaprazlama operatörlerinde 3 ebeveyn kullanıldığında, operatör seçim olasılıklarının model başarımına etkisi.

operatör seçim olasılıkları				kul			son		
o_{m_1}	o_{m_2}	o_{c_1}	o_{c_2}	ϕ	*ksn*	*grc*	ϕ	*ksn*	*grc*
2/3	1/3	1/2	1/2	0,34 (425)	3,59 (146)	13,33 (116)	0,69 (344)	1,76 (275)	1,13 (372)
1/2	1/2	1/2	1/2	0,32 (427)	3,56 (161)	13,21 (109)	0,67 (346)	1,79 (275)	1,13 (369)
1/3	2/3	1/2	1/2	0,35 (423)	3,64 (152)	13,72 (113)	0,67 (346)	1,54 (272)	1,14 (371)
2/3	1/3	1/3	2/3	0,32 (428)	3,03 (175)	12,96 (123)	0,70 (342)	1,46 (284)	1,19 (368)
1/2	1/2	1/3	2/3	0,33 (426)	3,29 (167)	13,95 (114)	0,67 (348)	1,48 (285)	1,14 (370)
1/3	2/3	1/3	2/3	**0,31 (428)**	**2,74 (180)**	13,37 (113)	0,72 (340)	**1,38 (289)**	1,16 (368)
2/3	1/3	2/3	1/3	0,32 (427)	3,53 (152)	13,61 (107)	0,70 (342)	2,12 (245)	1,09 (372)
1/2	1/2	2/3	1/3	**0,31 (428)**	4,43 (139)	13,77 (104)	0,66 (346)	1,86 (250)	1,14 (370)
1/3	2/3	2/3	1/3	0,34 (425)	4,11 (141)	14,03 (105)	0,71 (339)	1,93 (255)	**1,06 (372)**

Önceki bölümdekine benzer bir şekilde bu örneklerin hepsi için tamsayılı lineer modeli CPLEX ile çözerek bir alt sınır elde ediyoruz. Algoritmalarımıza, GFVGP problemi örneğinin yanı sıra, bu alt sınırı da girdi olarak veriyoruz ve alt sınıra eşit amaç fonksiyonu değerine sahip olurlu bir çözüm elde edince algoritmayı erken sonlandırıyoruz. Algoritmalarımızı, her örnek için 10 kere farklı rastgele çekirdeklerle, farklı jenerasyon sayısı ile çalıştırıyoruz. Bu 10 çalıştırma sonucunda elde edilen en iyi çözümleri ele alıyoruz. Her örnek için elde edilen sonuçların alt sınırdan ortalama yüzdece uzaklıklarının geometrik ortalamasını alarak bir modelin ortalama başarımını elde ediyoruz. Bu sayede, her örnek başarıma aynı oranda katkı sağlamaktadır. Alt sınırdan ortalama yüzdece uzaklığa ek olarak, kaç örnekte optimal çözümün bulunduğunu ve kaç örnekte hiç olurlu çözüm bulunamadığını da inceliyoruz.

İlk olarak, CPLEX'in optimal çözüm bulamadığı örnek sayılarını belirtiyoruz. $\langle kyf \mid kyf \mid \phi \mid kul \rangle$, $\langle kyf \mid kyf \mid ksn \mid kul \rangle$, $\langle kyf \mid kyf \mid grc \mid kul \rangle$, $\langle kyf \mid kyf \mid \phi \mid son \rangle$, $\langle kyf \mid kyf \mid ksn \mid son \rangle$, ve $\langle kyf \mid kyf \mid grc \mid son \rangle$ notasyonları ile ifade edilen modeller için, sırasıyla $25, 1, 58, 21, 0$ ve 10 örnek için optimal çözümü bulamamıştır. Bunların dışındaki bütün örnekler için optimal çözüm bulunmuştur.

Çizelge 4.4 bütün modeller için memetik algoritmalarımızın farklı jenerasyon sayıları ile elde edilen çözümlerin alt sınırdan ortalama yüzde uzaklığını göstermektedir. Buna ek olarak, olurlu çözüm bulunamayan örnek sayı parantez içerisinde yanda belirtilmiştir. Eğer bütün örnekler için olurlu çözüm bulunduysa, bu değer verilmemiştir. Jenerasyon sayısının 0 olduğu satırlar, 10 çalıştırmanın ilk jenerasyonlarında bulunan en iyi çözüme karşılık gelmektedir. Yani bu satırlardaki

Çizelge 4.3: Çaprazlama operatörlerinde rastgele olarak 2 veya 3 ebeveyn kullanıldığında, operatör seçim olasılıklarının model başarımına etkisi.

operatör seçim olasılıkları				kul			son		
o_{m_1}	o_{m_2}	o_{c_1}	o_{c_2}	ϕ	*ksn*	*grc*	ϕ	*ksn*	*grc*
2/3	1/3	1/2	1/2	0,35 (423)	3,41 (166)	12,84 (114)	0,69 (344)	1,71 (259)	1,09 (371)
1/2	1/2	1/2	1/2	**0,31 (429)**	3,38 (157)	12,99 (109)	0,67 (344)	1,63 (273)	**1,06 (374)**
1/3	2/3	1/2	1/2	0,32 (427)	3,26 (156)	13,46 (113)	0,69 (344)	1,56 (282)	1,11 (370)
2/3	1/3	1/3	2/3	0,32 (427)	3,20 (184)	12,56 (121)	**0,62 (352)**	1,61 (271)	1,18 (368)
1/2	1/2	1/3	2/3	**0,31 (428)**	2,96 (182)	12,88 (121)	0,71 (341)	1,42 (284)	1,21 (367)
1/3	2/3	1/3	2/3	0,35 (423)	3,54 (165)	12,65 (114)	0,76 (336)	1,41 (288)	1,16 (369)
2/3	1/3	2/3	1/3	0,33 (426)	3,86 (153)	14,05 (106)	0,65 (347)	1,85 (262)	1,19 (366)
1/2	1/2	2/3	1/3	**0,31 (428)**	3,79 (147)	13,59 (105)	0,69 (342)	1,81 (259)	1,15 (366)
1/3	2/3	2/3	1/3	0,32 (427)	3,71 (150)	13,53 (104)	0,70 (342)	1,75 (258)	1,22 (366)

değerler aslında, kurucu sezgisel algoritmalar tarafından bulunan çözümlerin yerel arama algoritmalarıyla iyileştirildikten sonra elde edilmiş en iyi çözümün alt sınırdan ortalama uzaklığını göstermektedir. Eğer başlangıç çözümü olurlu değilse, yerel arama algoritmalarının tamir algoritmalarını kullandıklarını hatırlayın.

İlk olarak geliştirilen yöntemlerin olurlu çözüm bulma başarılarını ele alacağız. Kullanılan tamir algoritmalarına rağmen, zamansal kısıtların olduğu 720 örnekten 91 tanesinde, ilk jenerasyonda olurlu bir çözüm bulunamamıştır. Bunlardan 89 tanesine memetik algoritmanın ileriki jenerasyonlarında bir olurlu çözüm bulunmuştur. Fakat 2 örnek için 100.000 jenerasyon sonunda bile olurlu çözüm bulunamamıştır.

Zamansal kısıtların olmadığı modeller için, her örnek için olurlu bir çözüm bulunmuştur. Fark edileceği üzere, bazı veritabanlarının boyutları bazı parti kapasitelerinden fazla olabildiği için ve parti sayısı sınırlı olduğu için bu modellerde bile olurlu çözüm bulmak Kutulama problemi kadar kolay değildir.

Sadece kesin zamansal kısıtların olduğu modeller, ilk jenerasyonda en çok olurlu çözüm bulunamayan problem modelidir. Fakat, sadece 1.000 jenerasyon içerisinde bile, bütün örnekler için olurlu bir çözüm bulunmuştur. Bu, kurucu sezgisel algoritmaların ve tamir algoritmalarının iyileştirilebileceğini göstermektedir. Fakat aynı zamanda, tasarlanan mutasyon ve çaprazlama operatörlerinin ilk jenerasyonun tamamen olursuz çözümlerden oluştuğu zamanlarda bile efektif bir şekilde çalıştığını göstermektedir.

Çizelge 4.4: Farklı jenerasyon sayıları için memetik algoritmaların başarımları.

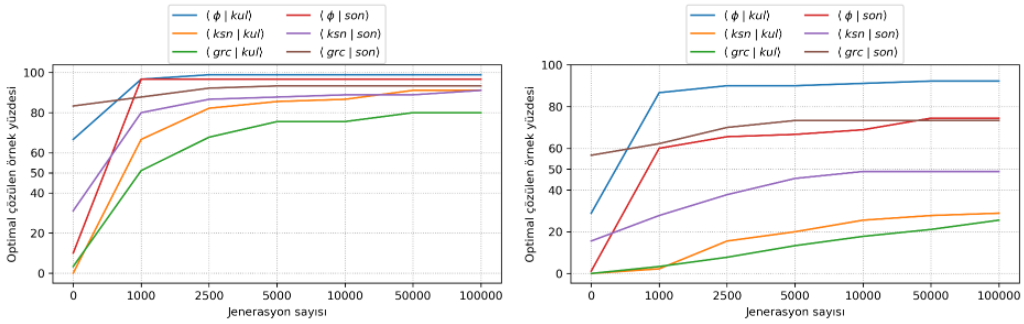
jenerasyon	kul			son		
sayısı	ϕ	*ksn*	*grc*	ϕ	*ksn*	*grc*
Veritabanı sayısı 50						
0	1,13	17,82(13)	32,71	5,71	4,70(20)	0,94
1.000	0,11	1,47	6,97	0,10	0,60	0,46
2.500	0,04	0,83	4,34	0,10	0,41	0,30
5.000	0,04	0,67	3,35	0,10	0,38	0,25
10.000	0,04	0,58	3,35	0,10	0,30	0,25
50.000	0,04	0,35	2,91	0,10	0,30	0,25
100.000	0,04	0,35	2,91	0,10	0,24	0,25
Veritabanı sayısı 100						
0	2,33	24,05(30)	58,16(3)	6,50	7,86(25)	2,12
1.000	0,25	4,78	24,08(2)	0,62	2,16	1,52
2.500	0,19	3,04	17,16(2)	0,54	1,34	1,12
5.000	0,19	2,39	14,31(2)	0,53	1,09	0,95
10.000	0,17	2,08	12,98(2)	0,48	1,04	0,94
50.000	0,15	1,76	11,42(2)	0,40	1,03	0,92
100.000	0,15	1,69	10,98(2)	0,40	1,03	0,90
Veritabanı sayısı $\in \{50, 100\}$						
0	1,73	20,51(43)	44,66(3)	6,10	6,21(45)	1,53
1.000	0,18	3,12	15,11(2)	0,36	1,38	0,99
2.500	0,11	1,93	10,49(2)	0,32	0,87	0,71
5.000	0,11	1,52	8,63(2)	0,31	0,73	0,60
10.000	0,10	1,33	8,00(2)	0,29	0,67	0,60
50.000	0,10	1,05	7,03(2)	0,25	0,66	0,59
100.000	0,10	1,02	6,82(2)	0,25	0,63	0,58

Hem kesin hem göreceli zamansal kısıtların bulunduğu modellerin örneklerinin sadece kesin zamansal kısıtların bulunduğu modellerin örneklerinden daha az sayıda kesin zamansal kısıt içerdiğini hatırlayın. İlk modeller için ikinci modellere göre ilk jenerasyonda daha az örnekte olurlu çözüm bulunamamıştır. Bu, geliştirilen kurucu sezgisel ve tamir algoritmalarının, göreceli zamansal kısıtlarla daha iyi başa çıktığını göstermektedir. Fakat, 2 örnek için binlerce jenerasyondan sonra bile olurlu bir çözüm bulamamıştır.

Şimdi, memetik algoritmaların ortalama performanslarını ve optimal çözüm buldukları örnek sayılarını inceleyeceğiz. Şekil 4.1'in sol tarafında veritabanı sayısının 50 olduğu örneklerin, sağ tarafındaysa veritabanı sayısının 100 olduğu örneklerin yüzde kaçında optimal çözüm bulunduğunun jenerasyon sayısına göre değişimi görülmektedir.

4.5 Zamansal Kısıtların Olmadığı GFVGP Modelleri

Çizelge 4.4'ten görüldüğü üzere, zamansal kısıtların olmadığı modellerde, memetik algoritmalarımız örneklerin çok büyük bir kısmını optimal çözmektedir. Optimal



Şekil 4.1: Her model için farklı jenerasyon sayıları için örneklerin yüzde kaçının optimal çözüldüğü gösterilmektir.

çözülemeyen örnekler içinse optimale çok yakın çözümler bulmaktadırlar.

Amaç fonksiyonu *kul* olan modeller için, memetik algoritmamız 2.500 ve 50.000 jenerasyonda, 180 örneğin sırasıyla 170 ve 172 tanesini optimal çözmektedir. Bu optimal çözümlerin 86 tanesi ilk jenerasyonda bulunmaktadır. İlk jenerasyonda bulunan en iyi çözümler, ortalamada alt sınırdan $\%1,73$ uzaklıktayken, sadece 1.000 jenerasyon sonunda bu ortalama uzaklık $\%0,18$'e düşmektedir. CPLEX'in bu örneklerden 25 tanesini 12 iş parçacığı ile 8 saatte bile optimal çözemediği unutulmamalıdır. Yani, algoritmamız CPLEX'in optimal çözemediği 17 örneği optimal çözmektedir. Algoritmamız sadece 1.000 jenerasyonda bile örneklerin büyük bir kısmını optimal çözdüğü için, jenerasyon sayısının artırılması algoritmanın performansını pek etkilememektedir.

Amaç fonksiyonu *son* olan modeller için, memetik algoritmamız 1.000 ve 100.000 jenerasyonda, 180 örneğin sırasıyla 141 ve 154 tanesini optimal çözmektedir. Optimal çözümlerin sadece 10 tanesi ilk jenerasyonda bulunmaktadır. İlk jenerasyonda bulunan en iyi çözümler, ortalamada alt sınırdan $\%6,1$ uzaklıktayken, sadece 1.000 jenerasyon sonunda ortalama uzaklık $\%0,36$'ya düşmektedir. Jenerasyon sayısının artırılması, veritabanı sayısının 100 olduğu örneklerde başarımı artırmaktadır.

4.6 Kesin Zamansal Kısıtların Olduğu GFVGP Modelleri

Amaç fonksiyonu *kul* olan modeller için, memetik algoritmamız 2.500 ve 50.000 jenerasyonda, 180 örneğin sırasıyla 88 ve 108 tanesini optimal çözmektedir. İlk jenerasyonda hiçbir örneğe optimal çözüm bulunamamıştır. Jenerasyon sayısı arttıkça algoritmanın başarımında kayda değer bir artış görülmektedir. İlk jenerasyonda bulunan en iyi çözümler, ortalamada alt sınırdan $\%20,51$ uzaklıktadır. Bu uzaklık, zamansal kısıtların olmadığı örneklere oranla oldukça yüksektir. Yine de, jenerasyon sayısı arttıkça uzaklık azalmakta 10.000 ve 100.000 jenerasyon sonunda, uzaklık sırasıyla $\%1,33$'e ve $\%1,02$'ye kadar düşmektedir.

Amaç fonksiyonu *son* olan modeller için, memetik algoritmamız 1.000 ve 100.000 jenerasyonda, 180 örneğin sırasıyla 97 ve 126 tanesini optimal çözmektedir. Bu optimal çözümlerin 42 tanesi ilk jenerasyonda bulunmuştur. Jenerasyon sayısı arttıkça algoritmanın başarımında kayda değer bir artış görülmektedir. İlk jenerasyonda

bulunan en iyi çözümler, ortalamada alt sınırdan %6,21 uzaklıktadır. Bu uzaklık, sadece zamansal kısıtların olduğu fakat amaç fonksiyonun *kul* olduğu örneklere göre azdır. Jenerasyon sayısı arttıkça uzaklık azalmakta ve 10.000 ve 100.000 jenerasyon sonunda, uzaklık sırasıyla %0,73'e ve %0,63'e kadar düşmektedir.

4.7 Göreceli Zamansal Kısıtların Olduğu GFVGP Modelleri

Amaç fonksiyonu *son* olan modeller için, memetik algoritmamız 1.000 ve 5.000 jenerasyonda, 180 örneğin sırasıyla 135 ve 150 tanesini optimal çözmektedir. Bu optimal çözümlerin 126 tanesi ilk jenerasyonda bulunmuştur. İlk jenerasyonda bulunan en iyi çözümler, ortalamada alt sınırdan %1,53 uzaklıktadır. Jenerasyon sayısı arttıkça uzaklık azalmakta ve 10.000 ve 100.000 jenerasyon sonunda, uzaklık sırasıyla %0,99 ve %0,58'e kadar düşmektedir. Bu uzaklıklar, sadece kesin zamansal kısıtların olduğu fakat amaç fonksiyonun *son* olduğu örneklerden bile azdır. Bunu iki sebeple açıklanabilir. İlk olarak, daha önce de belirttiğimiz gibi hem kesin hem göreceli zamansal kısıtların bulunduğu modellerin örnekleri sadece kesin zamansal kısıtların bulunduğu modellerin örneklerinden daha az sayıda kesin zamansal kısıt içermektedir. İkinci olarak, hem kesin hem göreceli zamansal kısıtların bulunduğu modellerin örnekleri BMHD örnekleri kullanılarak yaratıldı. Bu örnekler, diğer modeller için yarattığımız örneklerden farklı bir veritabanı boyutu parti kapasitesi oranına sahiptir. Bu yüzden de, çözümlerde neredeyse yarı sayıda parti kullanılmaktadır.

Amaç fonksiyonu *kul* olan modeller için, memetik algoritmamız 2.500 ve 100.000 jenerasyonda, 180 örneğin sırasıyla 68 ve 95 tanesini optimal çözmektedir. Bu optimal çözümlerin sadece 3 tanesi ilk jenerasyonda bulunmuştur. Fakat bu örnekler için algoritmamız 10.000 jenerasyon sonunda bile örneklerin çoğunu optimal çözememiştir. İlk jenerasyonda bulunan en iyi çözümler, ortalamada alt sınırdan %44,46 uzaklıktadır. Jenerasyon sayısı arttıkça uzaklık azalmakta 1.000, 2.500 ve 100.000 jenerasyon sonunda, uzaklık sırasıyla %15,11'e, %10,49'a ve %6,82'ye kadar düşmektedir. Uzaklıkların bu kadar yüksek olmasının bir sebebi, bu model için üretilen 180 örnekten 58 tanesinin CPLEX tarafından optimal çözülememesidir.

Fark edileceği üzere, genellikle memetik algoritmalarımız amaç fonksiyonunun *son* olduğu örneklerde amaç fonksiyonunun *kul* olduğu örneklere göre daha başarılırdır. Bu amaç fonksiyonu *son* olan örneklerde, hangi partilerin kullanılacağı aşikarken, aynı durumun amaç fonksiyonu *kul* için geçerli olmamasından kaynaklanmaktadır.

Zamansal kısıtların olmadığı örnekler için yukarıdakinin tam tersi bir durum geçerlidir ve amaç fonksiyonu *kul* olan modeller için daha iyi sonuçlar elde edilmektedir. Bu, zamansal kısıtların olmadığı ve amaç fonksiyonunun *kul* olduğu örneklerde, kapasitesi en yüksek partilerin kullanılacağının kesin olarak bilinmesiyle açıklanabilir.

Genel durumda, geliştirilen algoritmalar, veritabanı sayısının 50 olduğu 540 örnekten 486 tanesini, veritabanı sayısının 100 olduğu 540 örnekten 293 tanesini, yani bütün örneklerin %72,1'ini 10.000 jenerasyon sonunda optimal çözmüştür. Aynı zamanda, algoritmalar veritabanı sayısının 50 olduğu 540 örnekten 496 tanesini, veritabanı sayısının 100 olduğu 540 örnekten 309 tanesini, yani bütün örneklerin %74,5'ini 100.000 jenerasyon sonunda optimal çözmüştür. Bunlara ek olarak, örneklerin

%95'inde, algoritmalarımızın 10.000 ve 100.000 jenerasyon sonunda buldukları sonuçlar optimalden en fazla sırasıyla %10 ve %7'dir.

Algoritmalarımız optimal çözümü bulduklarında çalışmayı durdurdukları için ortalama çalışma süresini vermekten kaçınıyoruz ve bunun yerine en uzun çalıştıkları örneklerin sürelerini belirtiyoruz. Bütün modeller için optimal çözülemeyen örnek olduğu için, bu süre algoritmalarımızın çalışma süreleri için bir üst sınır olarak düşünülebilir. Algoritmaların 10.000 ve 100.000 jenerasyon için maksimum çalışma süreleri sırasıyla, 8 ve 82 saniyeden kısadır.

5. SONUÇLAR VE ÖNERİLER

Tez kapsamında, Veritabanı Göçü Planlaması probleminin bir varyantı olan Güvenlik Farkındalıklı Veritabanı Göçü Planlaması problemi literatüre kazandırıldı. Veritabanı Göçü Planlanması problemi test maliyetlerinin azaltılması açısından ele alınmış ve bu amaç fonksiyonu için hem teorik hem de deneysel bir çok sonuç ortaya konulmuştur [27, 24, 25]. Bu çalışmada, kullanılan parti sayısını ve kullanılan en büyük parti numarasını minimize etme açından iki farklı amaç fonksiyonu ele alındı. Bu amaç fonksiyonları veritabanı göçü esnasında güvenlik riskini azaltma bağlamında ortaya çıkmaktadır. Problem, önerilen problem çatısın altında 24 farklı modelde ele alındı ve bu modellerin 23 tanesinin hesaplama karmaşıklığı tespit edildi. Ayrıca birçok model için yakınsanabilirlik ve yakınsanamazlık sonuçları elde edildi. Problem çatısındaki **NP-zor** olan modeller için memetik algoritmalar, kurucu sezgisel algoritmalar ve yerel arama algoritmaları geliştirildi. Geliştirilen memetik algoritmalar 8 saniyeden az bir sürede, örneklerin %95'inde optimalden en fazla %10 uzaklıkta çözümler bulmakta ve örneklerin %72.1'ini optimal çözmektedir. Ayrıca, memetik algoritmalar yaklaşık 10 kat daha uzun çalıştırıldıklarında, örneklerin %95'inde optimalden en fazla %7 uzaklıkta çözümler bulmakta ve örneklerin %74.5'ini optimal çözmektedir.

Hesaplama karmaşıklığı belirlenemeyen $\langle sbt \mid kyf \mid ksn \mid kul \rangle$ notasyonuyla ifade edilen GFVGP probleminin hesaplama karmaşıklığının belirlenmesi ve geliştirilen sezgisel algoritmaların başarımlarının daha da iyileştirilmesi ileriye dönük çalışmalara bırakılmıştır.

KAYNAKLAR

[1] **Labrinidis**, **A.** and **Jagadish**, **H. V.** (2012). Challenges and opportunities with big data. In: *Proceedings of the VLDB Endowment* 5.12, pp. 2032–2033.

[2] **Zanella**, **A.**, **Bui**, **N.**, **Castellani**, **A.**, **Vangelista**, **L.**, and **Zorzi**, **M.** (2014). Internet of things for smart cities. In: *IEEE Internet of Things journal* 1.1, pp. 22–32.

[3] **Hayes**, **J.** (2015). Multimedia big data: Content analysis and retrieval. In: *Big-Data Analytics and Cloud Computing*, pp. 37–51.

[4] **Hashem**, **I. A. T.**, **Yaqoob**, **I.**, **Anuar**, **N. B.**, **Mokhtar**, **S.**, **Gani**, **A.**, and **Khan**, **S. U.** (2015). The rise of "big data" on cloud computing: Review and open research issues. In: *Information systems* 47, pp. 98–115.

[5] **Drumm**, **C.**, **Schmitt**, **M.**, **Do**, **H.-H.**, and **Rahm**, **E.** (2007). Quickmig: automatic schema matching for data migration projects. In: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 107–116.

[6] **Gandhi**, **R.**, **Halldórsson**, **M. M.**, **Kortsarz**, **G.**, and **Shachnai**, **H.** (2004). Improved Results for Data Migration and Open Shop Scheduling. In: *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, pp. 658–669.

[7] **McBrien**, **P.** and **Poulovassilis**, **A.** (1999). Automatic Migration and Wrapping of Database Applications - A Schema Transformation Approach. In: *Conceptual Modeling - ER '99, 18th International Conference on Conceptual Modeling, Paris, France, November, 15-18, 1999, Proceedings*, pp. 96–113.

[8] **Meier**, **A.** (1995). Providing Database Migration Tools - A Practicioner's Approach. In: *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland*, pp. 635–641.

[9] **Brodal**, **G. S.**, **Frigioni**, **D.**, and **Marchetti-Spaccamela**, **A.**, eds. (2001). *Algorithm Engineering, 5th International Workshop, WAE 2001 Aarhus, Denmark, August 28-31, 2001, Proceedings*. Vol. 2141. Lecture Notes in Computer Science. Springer.

[10] **Goldman**, **R.**, **McHugh**, **J.**, and **Widom**, **J.** (1999). From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. In: *ACM SIGMOD Workshop on The Web and Databases, WebDB 1999, Philadelphia, Pennsylvania, USA, June 3-4, 1999. Informal Proceedings*, pp. 25–30.

[11] **Hall**, **J.**, **Hartline**, **J.**, **Karlin**, **A. R.**, **Saia**, **J.**, and **Wilkes**, **J.** (2001). On algorithms for efficient data migration. In: *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pp. 620–629.

[12] **Gandhi**, **R.** and **Mestre**, **J.** (2009). Combinatorial Algorithms for Data Migration to Minimize Average Completion Time. In: *Algorithmica* 54.1, pp. 54–71.

[13] **Seo**, **B.** and **Zimmermann**, **R.** (2005). Efficient disk replacement and data migration algorithms for large disk subsystems. In: *TOS* 1.3, pp. 316–345.

[14] **Myllymaki**, **J.** (2001). Effective Web data extraction with standard XML technologies. In: *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pp. 689–696.

[15] **Vassiliadis**, **P.** and **Simitsis**, **A.** (2009). Extraction, Transformation, and Loading. In: *Encyclopedia of Database Systems* 10.

[16] **Behm**, **A.**, **Geppert**, **A.**, and **Dittrich**, **K. R.** (1997). *On the Migration of Relational Schemas and Data to Object-Oriented Database Systems*. Tech. rep. University of Zurich.

[17] **Wang**, **J.** and **Lochovsky**, **F. H.** (2003). Data extraction and label assignment for web databases. In: *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, Budapest, Hungary, May 20-24, 2003*, pp. 187–196.

[18] **Khuller**, **S.**, **Kim**, **Y. A.**, and **Wan**, **Y. J.** (2003). Algorithms for data migration with cloning. In: *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pp. 27–36.

[19] **Chatterjee**, **A.** and **Segev**, **A.** (1991). Data Manipulation in Heterogeneous Databases. In: *SIGMOD Record* 20.4, pp. 64–68.

[20] **Narayanan**, **D.**, **Thereska**, **E.**, **Donnelly**, **A.**, **Elnikety**, **S.**, and **Rowstron**, **A. I. T.** (2009). Migrating server storage to SSDs: analysis of tradeoffs. In: *Proceedings of the 2009 EuroSys Conference, Nuremberg, Germany, April 1-3, 2009*, pp. 145–158.

[21] **Hirofuchi**, **T.**, **Ogawa**, **H.**, **Nakada**, **H.**, **Itoh**, **S.**, and **Sekiguchi**, **S.** (2009). A Live Storage Migration Mechanism over WAN for Relocatable Virtual Machine Services on Clouds. In: *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGrid 2009, Shanghai, China, 18-21 May 2009*, pp. 460–465.

[22] **Chon**, **H. D.**, **Agrawal**, **D.**, and **El Abbadi**, **A.** (2002). Data Management for Moving Objects. In: *IEEE Data Eng. Bull.* 25.2, pp. 41–47.

[23] **Ferrandina**, **F.**, **Meyer**, **T.**, **Zicari**, **R.**, **Ferran**, **G.**, and **Madec**, **J.** (1995). Schema and Database Evolution in the O2 Object Database System. In: *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland*, pp. 170–181.

[24] **Subramani**, **K.**, **Caskurlu**, **B.**, and **Velasquez**, **A.** (2018). Minimization of Testing Costs in Capacity-Constrained Database Migration. In:

Algorithmic Aspects of Cloud Computing - 4th International Symposium, ALGOCLOUD 2018, Helsinki, Finland, August 20-21, 2018, Revised Selected Papers, pp. 1–12.

[25] **Acikalin**, **U. U.**, **Caskurlu**, **B.**, **Wojciechowski**, **P.**, and **Subramani**, **K** (2021). New Results on Test-Cost Minimization in Database Migration. In: *International Symposium on Algorithmic Aspects of Cloud Computing*. Springer, pp. 38–55.

[26] **Jensen**, **M.**, **Schwenk**, **J.**, **Gruschka**, **N.**, and **Iacono**, **L. L.** (2009). On Technical Security Issues in Cloud Computing. In: *IEEE International Conference on Cloud Computing, CLOUD 2009, Bangalore, India, 21-25 September, 2009*, pp. 109–116.

[27] **Patil**, **S.**, **Roy**, **S.**, **Augustine**, **J.**, **Redlich**, **A.**, **Lodha**, **S.**, **Vin**, **H. M.**, **Deshpande**, **A.**, **Gharote**, **M.**, and **Mehrotra**, **A.** (2010). Minimizing Testing Overheads in Database Migration Lifecycle. In: *COMAD*, p. 191.

[28] **Impagliazzo**, **R.** and **Paturi**, **R.** (2001). On the complexity of k-SAT. In: *Journal of Computer and System Sciences* 62.2, pp. 367–375.

[29] **Jr**, **E. C. man**, **Garey**, **M.**, and **Johnson**, **D.** (1996). Approximation algorithms for bin packing: A survey. In: *Approximation algorithms for NP-hard problems*, pp. 46–93.

[30] **Coffman**, **E. G.**, **Galambos**, **G.**, **Martello**, **S.**, and **Vigo**, **D.** (1999). Bin packing approximation algorithms: Combinatorial analysis. In: *Handbook of combinatorial optimization*. Springer, pp. 151–207.

[31] **De Carvalho**, **J. V.** (2002). LP models for bin packing and cutting stock problems. In: *European Journal of Operational Research* 141.2, pp. 253–273.

[32] **Clautiaux**, **F.**, **Alves**, **C.**, and **Carvalho**, **J. Valério de** (2010). A survey of dual-feasible and superadditive functions. In: *Annals of Operations Research* 179.1, pp. 317–342.

[33] **Delorme**, **M.**, **Iori**, **M.**, and **Martello**, **S.** (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. In: *European Journal of Operational Research* 255.1, pp. 1–20.

[34] **Friesen**, **D. K.** and **Langston**, **M. A.** (1986). Variable sized bin packing. In: *SIAM journal on computing* 15.1, pp. 222–230.

[35] **Murgolo**, **F. D.** (1987). An efficient approximation scheme for variable-sized bin packing. In: *SIAM Journal on Computing* 16.1, pp. 149–161.

[36] **Epstein**, **L.** and **Levin**, **A.** (2008a). An APTAS for generalized cost variable-sized bin packing. In: *SIAM Journal on Computing* 38.1, pp. 411–428.

[37] **Epstein**, **L.**, **Favrholdt**, **L. M.**, and **Levin**, **A.** (2011). Online variable-sized bin packing with conflicts. In: *Discrete Optimization* 8.2, pp. 333–343.

[38] **Jansen**, **K.** (1999). An approximation scheme for bin packing with conflicts. In: *Journal of combinatorial optimization* 3.4, pp. 363–377.

[39] **Jensen**, **T. R.** and **Toft**, **B.** (2011). Graph coloring problems. John Wiley & Sons.

[40] **Zuckerman**, **D.** (2006). Linear degree extractors and the inapproximability of max clique and chromatic number. In: *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pp. 681–690.

[41] **Epstein**, **L.** and **Levin**, **A.** (2008b). On bin packing with conflicts. In: *SIAM Journal on Optimization* 19.3, pp. 1270–1298.

[42] **Liu**, **Q.**, **Cheng**, **H.**, **Tian**, **T.**, **Wang**, **Y.**, **Leng**, **J.**, **Zhao**, **R.**, **Zhang**, **H.**, and **Wei**, **L.** (2021). Algorithms for the variable-sized bin packing problem with time windows. In: *Computers & Industrial Engineering* 155, p. 107175.

[43] **Wee**, **T.** and **Magazine**, **M. J.** (1982). Assembly line balancing as generalized bin packing. In: *Operations Research Letters* 1.2, pp. 56–58.

[44] **Crescenzi**, **P.** (1997). A short guide to approximation preserving reductions. In: *Proceedings of Computational Complexity. Twelfth Annual IEEE Conference*. IEEE, pp. 262–273.

[45] **Cormen**, **T. H.**, **Leiserson**, **C. E.**, **Rivest**, **R. L.**, and **Stein**, **C.** (2022). Introduction to algorithms. MIT press.

[46] **Chang**, **J.**, **Gabow**, **H. N.**, and **Khuller**, **S.** (2014). A model for minimizing active processor time. In: *Algorithmica* 70.3, pp. 368–405.

[47] **Garey**, **M. R.** and **Johnson**, **D. S.** (1979). Computers and intractability. In: *A Guide to the*.

[48] **Xu**, **J.** and **Parnas**, **D. L.** (1990). Scheduling processes with release times, deadlines, precedence and exclusion relations. In: *IEEE Transactions on software engineering* 16.3, pp. 360–369.

[49] **Falkenauer**, **E.**, **Delchambre**, **A.**, et al. (1992). A genetic algorithm for bin packing and line balancing. In: *ICRA*, pp. 1186–1192.

[50] **Falkenauer**, **E.** (1996). A hybrid grouping genetic algorithm for bin packing. In: *Journal of heuristics* 2.1, pp. 5–30.

[51] **Singh**, **A.** and **Gupta**, **A. K.** (2007). Two heuristics for the one-dimensional bin-packing problem. In: *OR Spectrum* 29.4, pp. 765–781.

[52] **Rohlfshagen**, **P.** and **Bullinaria**, **J. A.** (2010). Nature inspired genetic algorithms for hard packing problems. In: *Annals of Operations Research* 179.1, pp. 393–419.

[53] **Yeşil**, **Ç.**, **Turkyılmaz**, **H.**, and **Korkmaz**, **E. E.** (2012). A new hybrid local search algorithm on bin packing problem. In: *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*. IEEE, pp. 161–166.

[54] **Quiroz-Castellanos**, **M.**, **Cruz-Reyes**, **L.**, **Torres-Jimenez**, **J.**, **Gómez**, **C.**, **Huacuja**, **H. J. F.**, and **Alvim**, **A. C.** (2015). A grouping genetic algorithm with controlled gene transmission for the bin packing problem. In: *Computers & Operations Research* 55, pp. 52–64.

[55] **Holland**, **J. H.** (1992). Genetic algorithms. In: *Scientific american* 267.1, pp. 66–73.

[56] **Burke**, **E. K.**, **Newall**, **J. P.**, and **Weare**, **R. F.** (1998). Initialization strategies and diversity in evolutionary timetabling. In: *Evolutionary computation* 6.1, pp. 81–103.

[57] **McCall**, **J.** (2005). Genetic algorithms for modelling and optimisation. In: *Journal of computational and Applied Mathematics* 184.1, pp. 205–222.

[58] **Goldberg**, **D. E.** and **Deb**, **K.** (1991). A comparative analysis of selection schemes used in genetic algorithms. In: *Foundations of genetic algorithms*. Vol. 1. Elsevier, pp. 69–93.

[59] **Srinivas**, **M.** and **Patnaik**, **L. M.** (1994). Genetic algorithms: A survey. In: *computer* 27.6, pp. 17–26.

[60] **Baker**, **J. E.** (1985). Adaptive selection methods for genetic algorithms. In: *Proceedings of an International Conference on Genetic Algorithms and their applications*. Vol. 1. Hillsdale, New Jersey.

[61] **Piszcz**, **A.** and **Soule**, **T.** (2006). A survey of mutation techniques in genetic programming. In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp. 951–952.

[62] **Lozano**, **M.**, **Herrera**, **F.**, and **Cano**, **J. R.** (2008). Replacement strategies to preserve useful diversity in steady-state genetic algorithms. In: *Information sciences* 178.23, pp. 4421–4433.

[63] **Neri**, **F.** and **Cotta**, **C.** (2012). Memetic algorithms and memetic computing optimization: A literature review. In: *Swarm and Evolutionary Computation* 2, pp. 1–14.

[64] **Reeves**, **C.** (1996). Hybrid genetic algorithms for bin-packing and related problems. In: *Annals of Operations Research* 63.3, pp. 371–396.

[65] **Hussain**, **S.** and **Sastry**, **V.** (1997). Application of genetic algorithm for bin packing. In: *International journal of computer mathematics* 63.3-4, pp. 203–214.

[66] **Iima**, **H.** and **Yakawa**, **T.** (2003). A new design of genetic algorithm for bin packing. In: *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.* Vol. 2. IEEE, pp. 1044–1049.

[67] **Syswerda**, **G.** (1991). A study of reproduction in generational and steady-state genetic algorithms. In: *Foundations of genetic algorithms*. Vol. 1. Elsevier, pp. 94–101.

[68] **Scholl**, **A.**, **Klein**, **R.**, and **Jürgens**, **C.** (1997). Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. In: *Computers & OR* 24.7, pp. 627–645.